SRAM TABANLI CİHAZLARDA 2-BOYUTLU EG-LDPC KODLARINDAN YARARLANILARAK UZAY'IN RADYASYON ORTAMINDAN KAYNAKLANAN GEÇİCİ HATALARA KARŞI KORUMA SAĞLANMASI

MUSTAFA DEMİRCİ

YÜKSEK LİSANS TEZİ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ

TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

MART 2013

ANKARA

Fen Bilimleri Enstitü onayı

_____

Prof. Dr. Ünver KAYNAK

Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Doç. Dr. Hamza KURT

Anabilim Dalı Başkanı

Mustafa DEMİRCİ tarafından hazırlanan SRAM TABANLI CİHAZLARDA 2-BOYUTLU EG-LDPC KODLARINDAN YARARLANILARAK UZAY'IN RADYASYON ORTAMINDAN KAYNAKLANAN GEÇİCİ HATALARA KARŞI KORUMA SAĞLANMASI adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

_____

Doç. Dr. Bülent TAVLI

Tez Danışmanı

Tez Jüri Üyeleri

Başkan : Yrd. Doç. Dr. Ali BOZBEY _____

Üye : Doç. Dr. Bülent TAVLI _____

Üye : Yrd. Doç. Dr. Tolga GİRİCİ _____

Üye : Yrd. Doç. Dr. Enver ÇAVUŞ _____

Üye : Doç. Dr. Kemal BIÇAKCI _____

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Mustafa Demirci

Üniversitesi : **TOBB Ekonomi ve Teknoloji Üniversitesi**
Enstitüsü : **Fen Bilimleri**
Anabilim Dalı : **Elektrik ve Elektronik Mühendisliği**
Tez Danışmanı : **Doç. Dr. Bülent TAVLI**
Tez Türü ve Tarihi : **Yüksek Lisans – Mart 2013**

Mustafa DEMİRCİ

SRAM TABANLI CİHAZLARDA 2-BOYUTLU EG-LDPC KODLARINDAN YARARLANILARAK UZAY'IN RADYASYON ORTAMINDAN KAYNAKLANAN GEÇİCİ HATALARA KARŞI KORUMA SAĞLANMASI

ÖZET

Bu tez çalışmasında uzay sistemleri elektronik donanımlarında yer alan SRAM tabanlı belleklerde uzayın radyasyon ortamı nedeniyle görülebilecek geçici hatalara karşı korunma sağlayan yeni yöntemler önerilmektedir. Öncelikle hataların SRAM bellek yongaları ve SRAM tabanlı FPGA'ler üzerindeki etkilerini, hata özellik ve yapılarını gösterilecektir. Daha sonra hata özellikleri göz önünde bulundurularak hata etkilerini düzeltecek etkin yöntemler sunulacaktır. Önerilen çözüm, EG-LDPC kodları temel alınarak 2-boyutlu bir kod yapısı oluşturulması ve verinin kodlanarak SRAM belleklerde saklanmasıdır. Bahsi geçen 2-boyutlu kod yapıları sayesinde SRAM belleklerde saklanan veri veya FPGA yapılandırma belleğinin içeriklerinde görülen hataların etkileri ortadan kaldırılmıştır. Benzer amaçla kullanılan kod yapılarıyla karşılaştırıldığında; bu çalışmada önerilen kod yapısının daha yüksek hata tespit ve düzeltme kapasitesine sahip olduğu görülmüştür. Aynı zamanda, elde edilen kod çözücüler, EG-LDPC kodlarının donanımla gerçeklenmeye çok uygun olması nedeniyle düşük gecikmeye sahiptir ve daha az karmaşıktır.

Anahtar Kelimeler: Hata dayanıklılık, EG-LDPC kodları, Geçici hatalar, blok kodlar.

University : **TOBB Economics and Technology University**
Institute : **Institute of Natural and Applied Sciences**
Science Programme : **Electrical and Electronics Engineering**
Supervisor : **Asc. Prof. Bülent TAVLI**
Degree Awarded and Date : **M.Sc. – March 2013**

Mustafa DEMİRCİ

2-DIMENSIONAL EG-LDPC CODES FOR ACHIEVING FAULT TOLERANCE IN SRAM BASED DEVICES

ABSTRACT

In this thesis, two dimensional EG-LDPC codes based methods are explored to migitate effects of radiation induced soft errors on SRAM memory chips and SRAM based FPGAs in space environment. First, impacts of errors, error mechanisms and features of errors are shown. Then, solutions are proposed by employing 2-dimensional (2-D) error correction codes (ECC) composed of EG-LDPC codes for data to be stored at SRAM memory chips or at configuration memory of SRAM based FPGAs. Using 2-D ECC scheme proposed in this thesis it is shown that effects of soft errors at SRAM memories can be mitigated and error detection and correction capabilities are enhanced compared to 2-dimensional schemes suggested at previous works.

Keywords: Fault tolerance, EG-LDPC codes, Soft errors, Block codes.

TEŞEKKÜR

Bu çalışmayı gerçekleştirmemde emeği olan Yrd. Dr. Enver Çavuş'a, Doç. Dr. Kemal Bıçakcı ve Doç. Dr. Bülent Tavlı'ya ve beni her zaman destekleyen eşime teşekkür ederim.

İÇİNDEKİLER

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

KISALTMALAR

Kısaltma	Kısaltma (İng)	Açıklama	İngilizce
TOE	SEE	Tek Olay Etkisi	Single Event Effect
TOH	SEU	Tek Olay Hatası	Single Event Upset
ÇBH	MBU	Çoklu Bit Hatası	Multiple Bit Upset
TOK	SEL	Tek Olay Kilitlenmesi	Single Event Latch-up
TOGE	SET	Tek Olay Geçici Etkisi	Single Event Transient
TOFB	SEFI	Tek Olay Fonksiyonel Bozunması	Single Event Functional Interrupt
	SRAM		Static Random Access Memory
HTİO	FDIR	Hata Tespit İzolasyon ve Onarma	Fault Detection Isolation and Recovery
	COTS	Ticari Kullanıma Hazır	Commercial – off- the-Shelf
ÜMY	TMR	Üçlü Modüler Yedeklilik	Triple Modular Redundancy
APKD	FPGA	Alan Programlanabilir Kapı Dizisi	Field Programmable Gate Array
	ASIC	Uygulamaya Özgü Tümleşik Devre	Application Specific Integrated Circuit
TİD	TID	Toplam İyonizasyon Dozu	Total Ionization Dose
	CMOS		
BRAM	BRAM	Blok RAM	Block RAM
BRAMB	BRAMi	Blok RAM Bağlantı	Block RAM interconnect
	IOB	Giriş/çıkış Bloğu	Input/Output Block

SSİ	DSP	Sayısal Sinyal İşleme	Digital Signal Processing
	LUT	Başvuru Çizelgesi	Look-up Table
	RAM	Rastgele Erişimli Bellek	Random Access Memory
	FRDO	Çerçeve Yazmaç Verisi Çıkışı	Frame Register Data Output
	FRDI	Çerçeve Yazmaç Verisi Girişi	Frame Register Data Input
	FDAR	Çerçeve Verisi Adres Yazmacı	Frame Data Address Register
THD/ÇHT	SEC/DED	Tek Hata Düzelten / Çift Hata Tespit Eden	Single Error Corrector / Double Error Detector
PBN	PIP	Programlanabilir Bağlantı Noktası	Programmable Interconnect Point
HDK	ECC	Hata Düzeltme Kodu	Error Correction Code
ÖG	EG	Öklid Geometrisi	Euclidian Geometry
	EG-LDPC	Öklid Geometrisi Düşük Yoğunluklu Eşlik Denetimi	Euclidian Geometry Low Density Parity Check
GA	GF	Galois Alanı	Galois Field
TBÇM	OSML	Tek Basamaklı Çoğunluk Mantığı	One Step Majority Logic

SEMBOL LİSTESİ

Bu çalışmada kullanılmış olan simgeler açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	**Açıklama**
ρ	Radyasyona maruz kalan hedef maddenin özkütlesi(mg/cm^3)
E	Parçacık enerjisi(MeV)
r	Enerji yüklü parçacığın hedef kütle içinde kat ettiği mesafe(cm)
u, i	Bilgi bit vektörü
v, c	Kod sözcüğü
n	Kod sözcüğü genişliği
k	Bilgi vektörü uzunluğu
G	Üreteç Matris
H	Eşlik Denetim Matrisi
e	Hataya maruz kalmış vektör
s	Sendrom Vektörü
d(v,w)	İki vektör arası uzaklık
d_{min}	Bir blok kodun minimum uzaklığı
Γ	Eşlik denetim matrisi sütun ağırlığı
P	Eşlik denetim matrisi satır ağırlığı
σ_{cihaz}	Cihaz kesit alanı
σ_{bit}	Bit kesit alanı
θ	Parçacık huzmesi etki açısı
v_L	Etki vektörü
Q_c	Kritik Enerji

1. GİRİŞ

Uzay uygulamalarında kullanılan elektronik donanımlarda görülebilecek radyasyon etkilerine karşı önlem almak için tasarımcılar tarafından izlenebilecek iki yol haritası bulunmaktadır:

- Tasarımda uzay koşullarına uygunluğu kanıtlanmış radyasyona karşı güçlendirilmiş (Radiation-Hardened, Rad-Hard) bileşenlerin kullanılması: Önleyici yöntemler yonga tasarımı sırasında işlevsel bloklara entegre edilir. Uzay koşullarına uygunluk ise bileşenin önceki uzay görevlerindeki görev başarısı ile doğrulanır.
- Hata Tespit, İzolasyon ve Onarım (FDIR) tasarım teknikleri kullanılarak bozukluğa dayanıklı sistemlerin oluşturulması: Ticari kullanıma hazır (COTS) bileşenler ile Hata Tespit ve Düzeltme (EDAC) yöntemler bir arada kullanılır.

Güvenilirlik, performans, maliyet ve ulaşılabilirlik gibi kısıtlar dikkate alındığında her iki yol haritası da ödünleşimlere sahiptir.

COTS olarak üretilmiş SRAM bellek yongaları ve SRAM tabanlı FPGA'ler, radyasyona karşı güçlendirilmiş donanımlarla karşılaştırıldığında birçok avantaj sunmaktadır. Bellek erişim prosedürlerinin az karmaşıklığa sahip olması ve düşük gecikme ile erişim sağlaması, SRAM belleklerin başlıca avantajlarındandır. Bu nedenle mikrodenetleyici veya FPGA üzerinde koşan uygulamaların dinamik bellek ihtiyaçları için tercih edilirler. SRAM tabanlı FPGA'ler duruma özel uygulamaların gerçeklenmesi için geleneksel mikroişlemci uygulamalarından daha hızlı ve ASIC çözümlerinden daha düşük maliyetlidir [1]. Ayrıca, SRAM tabanlı FPGA'ler yeniden programlanabilme özelliği sayesinde, var olan uygulamalar işlevselliğini ve görevini sürdürürken yeni uygulamalar veya uygulamaların yeni gerçeklemeleri için

cihazların yeniden yapılandırılabilmesi esnekliğini sunar. Bunların yanı sıra SRAM tabanlı FPGA'ler ticari kullanıma hazır malzemeler olduğundan ve FPGA'lerin yer aldığı tasarımlar olgunlaşmış tasarım araçlarıyla ortaya çıkarıldığından, uzay sistemlerinde maliyetin düşmesine yardımcı olur. Bu avantajlarıyla uçucu programlama belleğine sahip FPGA'ler uzay tabanlı işleme görevlerinde son yıllarda sıkça kullanılmaktadır [1].

Buna karşın bahsedilen FPGA'lerde ASIC'lerden farklı olarak mantık devreleri ve bağlantıları SRAM tabanlı devreler aracılığıyla gerçeklendiğinden FPGA'ler üzerinde yer alan tasarımlarda Tek Olay Hataları (Single Event Upset, SEU) ve Çoklu Bit Hataları (Multiple Bit Upset , MBU) sonucu hata kipleri gözlemlenebilir. Radyasyon etkilerinin sonuçları FPGA'lerde SEU'lar ve MBU'lar etkilerini işlenmekte olan verinin ve / veya verinin işlendiği fonksiyonun değişmesi şeklinde gözlemlenebilir. Başka bir etki sonucu ise FPGA içindeki bloklar ve temel yapıların oluşturduğu sayısal devreler arasındaki bağlantı ağlarının bozulmasıdır. Bu tür hatalar sonucunda uydularda görev kaybı veya işlevsellikte azalma ile karşılaşılabilir [2]. SRAM tabanlı FPGA'ler yukarda anlatıldığı şekilde hatalara karşı hassas olduğundan, tasarımda yer alan FPGA'lerin uzay ortamındaki radyasyon etkilerini azaltıcı önlemlerle desteklenmesi gerekmektedir.

Ticari kullanıma hazır elektronik malzemeler uzay ortamının radyasyon etkilerine karşı hassas iken radyasyona karşı güçlendirilmiş elektronik bileşenler güvenilirlik açısından üstündür. Radyasyona karşı güçlendirilmiş elektronik bileşenler daha yüksek fiyat ve daha uzun tedarik süresine sahiptir. Ayrıca bu bileşenler askeri ve kritik malzemeler sınıfında bulunduğundan regülasyonlara [3] tabidirler ve bu nedenle ulaşılabilirlikleri kısıtlıdır. Ticari kullanıma hazır bileşenler daha ucuzdur ve elde edilmesi daha kolaydır. Bunların yanı sıra ticari kullanıma hazır bileşenler ve radyasyona karşı güçlendirilmiş bileşenlerin üretim sırasında kullanılan işlem teknolojisi isterleri birbirinden farklıdır. Radyasyona karşı güçlendirilmiş bir bileşen tasarlanırken radyasyon etkilerine karşı gürbüz bir mimari ve malzeme seçilir. Bu seçim nedeniyle hız ve performans özelliklerinde ticari kullanıma hazır bileşenlere

göre dezavantajlıdır. Ayrıca fiziksel kaynakların bir kısmı da güvenilirliği iyileştirme için kullanılır. Örnek olarak, dahili Üçlü Modüler Yedeklilik'e (Triple Modular Redundancy, TMR) sahip FPGA'ler ve bellek sistemleri tasarımın/bellek ihtiyacının yaklaşık 3 katı kadar fiziksel kaynağa ihtiyaç duyar ve daha fazla güç tüketir. Ticari kullanıma hazır elektronik bileşenler radyasyona karşı korumalı bileşenlere kıyasla, düşük güç tüketimli yüksek kapasiteye sahip daha yoğun fiziksel kaynak, daha yüksek hız ve çalışma frekansı sunar. Uzay sistemlerinde veri işlemenin yörüngede gerçekleştirilmesinin daha çok tercih edilmeye başlanması, artan görev çeşitlilikleri ve karmaşıklıkları nedeniyle bileşenlerin performansı, güç tüketimi, fiziksel kaynak miktarı ve hız daha önemli hale gelmektedir. Bu kriterler açısından avantajları bünyesinde barındıran COTS ürünler için istenilen güvenilirlik değeri Hata Düzeltme Kodları (HDK) gibi yöntemlerle istenilen seviyeye getirebilir.

Öklid Geometrisi Düşük Yoğunluklu Eşlik Denetim (EG-LDPC) kodu eşlik denetim matrisinin yapısı nedeniyle kod çözücü donanımla gerçeklenmek için çok uygundur ve kod çözücü devreleri mantık devreleriyle gerçeklenebilir. Kod çözümü için tek basamaklı çoğunluk mantığı yapısı tercih edildiğinde kod çözücü devresi düşük karmaşıklığa sahiptir ve gecikmeler düşüktür. Bu avantajlarının yanı sıra EG-LDPC kullanılarak oluşturan iki boyutlu kod yapıları, çoklu bit hatalarını düzeltmek için kullanılan 1- ve 2- boyutlu HDK'lere oranla daha iyi hata tespit ve düzeltme yeteneklerine sahiptir.

Bu çalışmada SRAM bellek yongalarında depolanan verilerin ve SRAM tabanlı FPGA'lerin yapılandırma verilerinin maruz kaldığı uzay radyasyon etkileri sonucu oluşan Tek Olay Hataları (SEU) ve Çoklu Bit Hatalarının (MBU) etkilerinin azaltması için EG-LDPC kodu tabanlı 2-boyutlu HDK'ler önerilmiştir. Bu HDK'ler FPGA üzerinde gerçeklenerek, yapılan simülasyonlarla bu kodların hata tespit ve düzeltme yetenekleri ölçülmüş ve elde edilen sonuçlar daha önceki çalışmalarda farklı kod yapılarıyla karşılaştırılmıştır.

Bu tezin ana hatları şu şekildedir: 2. bölümde bu tez kapsamında kullanılan temel bilgilere yer verilecektir. Bu bölümde uzayın radyasyon etkilerine, etki türleri ve etkilerinin oluşumlarına, hata etkilerine maruz kalan bellek ve FPGA yapılarına, hata düzeltme kodları ve bu çalışmada temel olarak kullanılacak EG-LDPC kodlarına değinilmiştir. İlişkili Çalışmalar bölümünde radyasyon etkileri sonucu oluşan hataların özelliklerini ortaya koyan deneysel ve teorik çalışmalar ile önceki çalışmalarda bu hatalara karşı önlem olarak sunulan yöntemlerden bahsedilmiştir. Dördüncü kısımda bu tez kapsamında gerçeklenen hata düzeltme yöntemlerine yer verilirken beşinci kısımda bu çalışmaların performansının değerlendirilmesinde ve önceki çalışmalarla karşılaştırılmasında kullanılan deneysel sonuçlar verilmiştir. Beşinci kısımda bu çalışmada sunulan yöntemlerin avantajları / dezavantajları ve önceki çalışmalarla performans karşılaştırılmaları yapılarak tez sonlandırılmıştır.

2. TEMEL KONULAR

Bu bölümde yapılan tez çalışması kapsamında anlaşılması faydalı olacak bilgilere yer verilmiştir.

2.1 Uzay'ın Radyasyon Etkileri

Modern elektronik bileşenler uzaydaki ışınımsal dış etkenlere karşı çok hassastırlar. Yüksek enerjili ve/veya iyonlaşan proton, ağır iyon ve elektron gibi parçacıklar birçok gözlemlenebilir etkiye neden olur. Bu etkiler performansın düşmesinden işlevsel bozulmaya kadar birçok şekilde ortaya çıkabilir. Uyduların görev ömürlerinde kısalmaya ve uydularda önemli hatalarla neden olabilirler [2].

Uzay radyasyonu çeşitli enerji ve çok yönlü akılarla tetiklenen güneşsel olaylardan güçlü bir şekilde etkilenen tekdüze olmayan ve dinamik bir ortam oluşturur. Farklı yükseklikteki yörüngeler için radyasyon olayları farklılıklar gösterir.

Ticari kullanıma hazır bileşenlerin kullanımı görev risklerini azaltmak adına daha titiz test ve etkilere karşı güçlendirme yöntemlerinin gerekliliğini ortaya çıkarır. Modern bileşenlerin karmaşıkları arttıkça radyasyon tepkileri çeşitlenmekte ve yeni hata kipleri ortaya çıkmaktadır[4].

Elektronik cihazlarda gözlemlenen bozuklukların kaynağı iyonlaşan uzay çevresel koşullarında bulunan yüksek enerjili parçacıklardır. Kaynaklarına bağlı olarak 3 sınıfa ayrılabilirler:

1) Radyasyon Kemerleri: Elektronlar (30MeV'e kadar) ve protonlar (500MeV'e kadar) gibi enerji yüklü parçacıklar yeryüzünün manyetik alanı tarafından hapsedilirler. Yeryüzüne yakın ve sakınılması gereken bir bölge oluştururlar.

2) Güneşsel Parlamalar: Protonlar (500MeV'e kadar) ve ağır iyonlar(10 MeV'e kadar) güneşsel olaylar sırasında yayılırlar. Güneşsel döngüden etkilenirler, "güneşsel maksimum" denen periyotlarda olaylar daha sık görülür.
3) Kozmik Işımalar: Kozmik ışımalar enerji seviyeleri çok yüksek olabilen iyonların dayanağıdır. Kozmik ışımaların kaynakları güneş sistemi dışındandır.

Yarı iletken cihazların bahsi geçen parçacıklarla etkileşimi sonucu çeşitli etkiler görülebilir:

1) Total İyonlaşma Dozu (TİD): İyonlaşma dozunun iletken içerisinde uzun süre zarfında depolanması ile etkileri görülür. Cihazın elektriksel performansında düşüşe neden olur.
2) Yer Değiştirme Hasarı: İyonlaşmayan protonlar ya da yüksek enerjili elektronlar hedefte birikerek yer değiştirme hasarıyla sonuçlanan kafes yapısında bozukluklara neden olur. Elektriksel performansı düşürdüğü ve arka plan gürültülerini arttırdığı için sensor ve yükseltici gibi cihazlar için kritiktir.
3) Tek Olay Etkileri(TOE): Bir parçacıktan kaynaklanan yüksek enerji iyonlaşma dozunun birikmesi sonucu TOE'ler görülür. Proton ve ağır iyonlardan kaynaklanır. Transistorların durumlarında değişikliğe yol açar.

2.1.1 TOE'lerin fiziksel oluşumu

Bu çalışmada TOE'lerin SRAM tabanlı FPGA ve bellek yongaları üzerindeki etkileri incelenecektir. Bu nedenle radyasyon etkilerini temel SRAM hücre yapısının maruz kaldığı etkilerden başlayarak yonga seviyesinde görülen etkileri açıklamak faydalı olacaktır.

TOE'lerin fiziksel oluşumu hassas bölge içerisinde ağır iyonların etkileşimi sonucu indüklenen yük aktarımı ve bunu takiben yükün çıkış düğümünde toplanmasıyla açıklanabilir. Hassas bölgeler dahili elektrik alanının yük depolamasına izin verdiği ters ön-beslemeli düğümlerdir.

Ağır iyonlar enerjiyi birincil olarak iyonlaşmayla aktarırlar. Etkileşim iki basamakta tanımlanır:

1) δ ışınlarının açığa çıkması: dışarıdan gelen parçacıkların ortamdaki elektronlarla etkileşimi sonucu ikincil elektronlar meydana gelir.
2) Elektron/boşluk çiftinin oluşumu: ikincil elektronlar ortamla etkileşime girerek elektron/boşluk çiftini meydana getirir. Bu yüklerin dairesel dağılımı iyon güzergahını oluşturur.

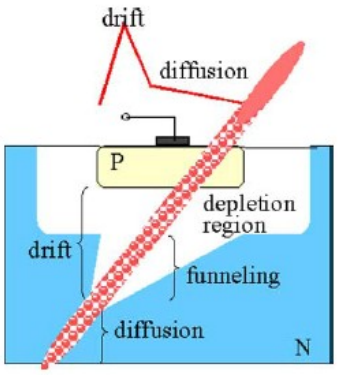
Enerji yüklü parçacık ile etkileşime giren hedef arasındaki yük aktarımı doğrusal enerji transferi(LET) kullanılarak modellenir. LET hedefte kat edilen birim mesafe için iyonlaşma ile aktarılan enerji miktarıdır.

$$\mathbf{LET} = \frac{\mathbf{1}}{\boldsymbol{\rho}}\frac{\mathbf{dE}}{\mathbf{dx}}(\mathbf{x})\left(\frac{\mathbf{MeV}}{\mathbf{mg.\,cm^2}}\right) \tag{2.1}$$

ρ hedef özkütlesi (mg/cm^3), E parçacık enerjisi(MeV) ve r de parçacığın menzilidir(cm). Aktarılan enerji miktarı TOE'ler için cihazlardaki ilk göz önünde bulundurulması gereken niceliktir. LET parametresinin kullanımı bilinen boyutlara sahip bir hacimde depolanan enerjinin hesaplanabilmesine olanak sağlar.

Enerji yüklü parçacığın çarpması sonucu yarı iletken içerisinden 3 temel mekanizma yer alır(Şekil 2.1):

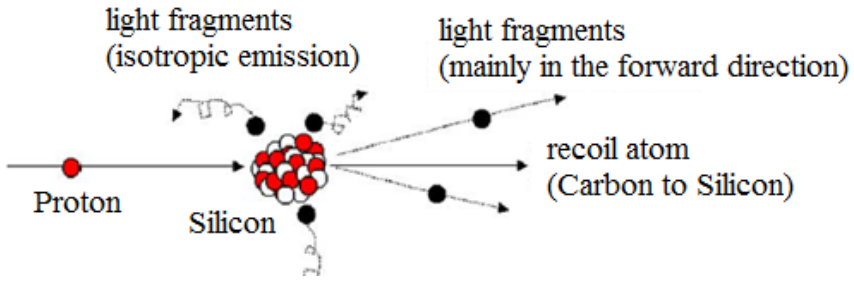
- Enerji boşalmasının meydana geldiği bölgede elektron boşluk çifti yerel potansiyel farkı ya da dışarıdan uygulanan alan nedeniyle ayrışır, daha sonra elektron ve boşluk akımları yarı iletkende gözlemlenir.
- Güzergahta yer alan yüklerin yüksek yoğunluğu nedeniyle, üstleçtehunibiçiminde oluşan güzergah boyunca gerilimde düşüş gözlenir.
- Ek yükler difüzyon yoluyla düğüme kayacaktır.



Şekil 2.1 Yük toplanma mekanizması[4]

Hassas düğümde depolanan enerji, bir cihaz parametresi olan kritik enerjiden(Q_c) yüksek olursa bir TOE meydana gelir. Kritik enerji; transistor geometrisi, üretim teknolojisi ve malzemeye bağımlıdır.

Enerji yüklü iyonların aksine protonların LET'leri direkt olarak TOE'ye sebebiyet vermek için çok düşüktür. Ancak protonların hedefle elastik veya elastik olmayan etkileşimleri sonucu protonların enerjilerinin bir kısmını geri tepme atomlarına aktarır. Geri tepme atomlarının kütlesi etkin protonlardan daha yüksek olduğundan LET'leri de yüksektir. Protonlar açısından TOE, ikincil parçacıkların enerjilerini aktarmasıyla doğrudan gerçekleşmeyen bir mekanizmadır(Şekil 2.2).

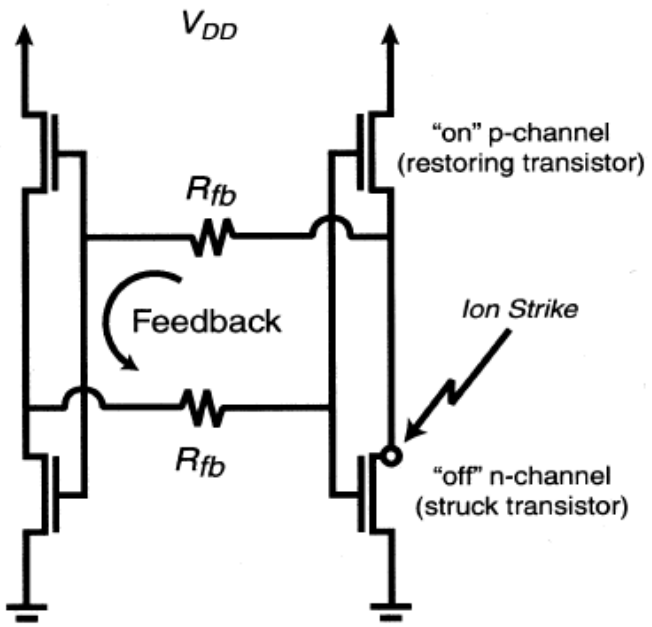


Şekil 2.2 Bir protonun bir silikon atomuyla elastik olmayan etkileşimi[4]

Transistor boyutunda fiziksel olgularla tetiklenen TOE'lerin oluşum mekanizması [5]'te verilmiştir. SRAM'lerdeki hata mekanizması, tipik SRAM bellek hücresi yapısını oluşturan karşılıklı bağlaşımlı evirici çiftindeki aktif geri beslemeden kaynaklanır. SRAM hücresini oluşturan 4 transistorun savakları(drain) parçacık çarpmalarına karşı hassastır. Enerji yüklü parçacık genellikle ters ön-beslemeli savak düğümü olan hassas bölgeye çarptığında transistor "kapalı" konumuna geçer, düğümde toplanan yük parçacığın çarptığı transistorda kısa sürekli bir akım indükler. Evirici çiftteki "açık" konumunda olan transistor parçacık tarafından indüklenen akımı dengelemek için bir akım üretir. Onarıcı olarak adlandırılan "açık" konumdaki transistordan akan akım transistorun sonlu akım sürme değeri ve kanal iletkenliğine sahip olması nedeniyle savakta gerilim düşer. Gerilim düşmesiyle birlikte evirici çiftin çıkışında normal işleyişte olması gerekenden farklı bir mantık değeri görülür. Eviricinin çıkışındaki mantık değerinin değişmesi ve bunun sonucunda SRAM bellek hücresinin içeriğinin de değişmesi sonucu SEU meydana gelmiş olur. Bu mekanizmanın gösterimi Şekil 2.3'de verilmiştir.

Bir SRAM'in SEU ve MBU'lara karşı hassasiyeti SRAM hücresinin fiziksel büyüklüğüyle yani daha önce değinildiği gibi işlem teknolojisinin yetenekleriyle doğrudan ilişkilidir. Fiziksel büyüklük ve hatalara karşı hassasiyet arasındaki ilişkiyi açıklamak için hücre geri besleme zamanı olgusunun tanımını yapmakta fayda vardır. Hücre geri besleme zamanı parçacık çarpmasından etkilenen düğümlerin,

evirici üzerinden gerilimi geri beslemesi için gerekli süre yani cihazın parçacık çarpmasından yanlış mantık değerini çıktı olarak vermesine kadar geçen süredir. Hücre geri besleme zamanı temel olarak eviricideki *RC* gecikme zamanıyla benzeşir. *RC* gecikmesi ne kadar küçükse hücre gerilim değişimlerine o kadar çok çabuk cevap verir, yani değişimlerden kolay etkilenir ve sonuç olarak SEU'lara karşı daha hassastır. Bunun yanında enerji yüklü parçacığın LET'i, çarpmanın gerçekleştiği nokta, işlem teknolojisi açısından ise onarıcı transistorun akım sürme yeteneği ve azınlık taşıyıcılar, SEU'ların oluşumunu etkileyen diğer fiziksel faktörlerdir. Özetlemek gerekirse, yarı iletken teknolojisinin gelişimiyle transistor boyutları gün geçtikçe küçülmekte ve transistorlar hızlanmaktadır. Bu değişim de hatalara karşı daha hassas SRAM yapılarını beraberinde getirmektedir.



Şekil 2.3 SRAM hücresinde TOE mekanizması[5]

2.1.2 TOE Türleri

Doğal olaylar sonucu meydana gelen uzay radyasyon ortamı katı-hal elektronik cihazların ve tümleşik devrelerin elektriksel özelliklerinde kalıcı veya geçici değişimlere neden olur. Radyasyon bozukluklarına dayanıklı elektronik cihaz geliştirmesinde elektronik malzemelerde oluşan temel etkilerle ilgili bilgi sahibi olmak önemli bir noktadır[6]. Bu etkilerin tanımları [7]'de verilmiştir.

Tek Olay Etkileri(Single Event Effect, SEE), bir mikroelektronik analog veya sayısal cihazın, bileşenin, alt sistemin yada sistemin durum veya performansında tek bir enerji yüklü parçacık çarpması nedeniyle oluşan gözlemlenebilir ve ölçülebilir değişim şeklinde tanımlanır.TOE'ler tek bir enerji yüklü parçacığın uzay ortamında görevini sürdüren elektronik donanıma çarpması ve enerjisini aktarması üzerine hataya neden olması sonucunda görülür. SRAM tabanlı bellek ve FPGA'ler ile alakalı olan TOE kategorileri Tek Olay Kilitlenmesi (Single Event Latchup, SEL), Tek Olay Geçici Etkileri (Single Event Transient, SET), Tek Olay Hataları(Single Event Upset, SEU) Çoklu Bit Hatası (Multiple Bit Upset, MBU) ve Tek Olay Fonksiyonel Bozulmalarıdır(Single Event Functional Interrupt, SEFI). Her bir durum farklı gözlemlenebilirliğe ve sonuçlara sahiptir(Çizelge 2.1).

Çizelge 2.1 Tek Olay Etkileri

Tek Olay Etkileri (TOE)	**Sonuç**	**Hedef**
Tek Olay Kilitlenmesi (SEL)	Bileşenin yüksek akım çekmesi	CMOS/BiCMOS cihazlar
Tek Olay Geçici Etkileri (SET)	Genliği ve süre belirli dürtü tepkisi	Analog / karışık sinyal devreleri
Tek Olay Hatası (SEU)	Bellek elemanı içerik verilerinin bozulması	Bellekler
Çoklu Bit Hatası (MBU)	Birden fazla bellek hücresinin tek bir parçacık çarpmasıyla bozunması	Bellekler
Tek Olay İşleyiş Bozukluğu (SEFI)	Normal işleyiş kaybı	Dahili kontrol ve durum devreleri

Tek Olay Kilitlenmesi(SEL), tek bir enerji yüklü parçacığın cihaz yapısının hassas bölgelerinden geçmesi sonucu oluşan ve işlev kaybıyla sonlanan, cihazın yüksek akımlı durumudur. TOK cihazda kalıcı hatalara neden olabilir. Cihaz kalıcı olarak hasar görmediyse cihaz güç döngüsüne sokularak(önce kapatılıp sonra açılarak) normal işlevine geri dönüş yapabilir. CMOS cihazlardaki parazitik transistorlerin radyasyon tarafından tetiklenen pozitif geri besleme şartlarından etkilenmesi sonucu görülür. Örnek olarak tek bir enerji yüklü parçacığın güç ve toprak arasında tetiklediği parazitik bipolar kısa devre nedeniyle görülebilir.

Tek Olay Geçici Hataları(SET), enerji yüklü parçacık çarpması sonucu kısa sürede cihaz boyunca yayılanani voltaj yükselmelerine transistörlerin çıkışlarında görülen geçici değişikliklere neden olur.

Tek Olay Hataları(SEU) enerji yüklü parçacığın tetiklediği anlık, sinyallerin sonucunda görülen geçici hatalar olarak tanımlanır. SEU'lar bellek elemanların içeriklerinin "0" mantıksal seviyesinden "1"e ya da "1" mantıksal değerinden "0"a dönüşmesi şeklinde görülür ve ancak orijinal değerin bellek hücresine yazılmasıyla ancak bu olayın etkisi ortadan kaldırılabilir. SRAM bitleri bu FPGA'lerde mantık fonksiyonlarını ve rotalamayı gerçeklemekte ve denetlemekte kullanıldığı için burada bahsedilen bozulmalar SRAM tabanlı FPGA'ler için geçerli olan problemlerdir. Çoklu Bit Hataları(MBU) ise tek bir parçacık çarpması sonucu, SEU etkilerinin birden fazla bellek hücresinde aynı anda görülmesidir.

Tek Olay Geçici Hataları(SET) CMOS elemanlarda birleşimsel mantık elemanlarınınters önbeslenmiş kesişimlerineyüksek enerjili parçacıkların çarpması sonucu ortaya çıkar. Örneğin eviricilerde SET, ters önbeslenmiş veya kapalı transistörü açmaya yetecek yükü yükleyebilecek ve o mantık kapısının çıkışını geçici olarak değiştirip çevresindeki devrelerde ilerleyebilecek sinyali oluşturabilmektedir. En ciddi sonuçlar oluşan geçişlerin kritik sinyallerde örneğin saat, asenkron reset gibi sinyallerde oluşması ile kapan devrelerinin yanlış veri yüklemesi sonucunda ortaya çıkacaktır.

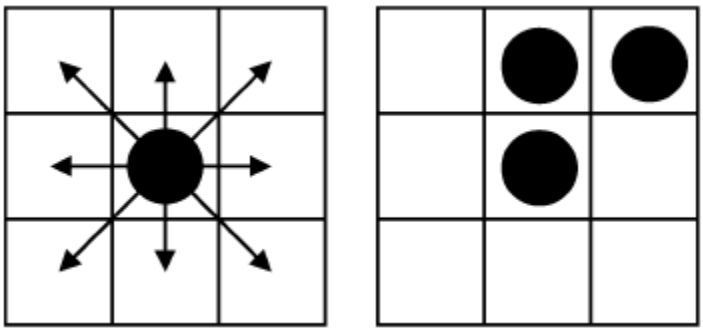
SRAM tabanlı aygıtlarda en çok dikkat edilmesi gereken ve en çok görülen TOE olgusu SEU ve MBU'lardır. SEU, SET'in bellek hücrelerinin iç devrelerinde oluşması sonucu anlık olarak yanlış verinin yakalanması durumu olarak görülebilir. Örneğin yeterli uzunlukta ve büyüklükte SET, SRAM bitlerini oluşturan çapraz bağlı mantık kapılarından birini etkileyerek diğer kapının durum değiştirmesine ve saklanan bitin terslenmesi SEU'lere sebebiyet verir. SEU'lar yıkıcı hatalar değildir ve etkilerini onarmak mümkündür.

SEU'nun tespit edilebilmesi ve sonuçları, hatanın nerede oluştuğuna bağlı bir durumdur. Kullanıcı tasarımına karşılık gelmeyen bir SEU çıkışları etkilemeyecektir.

2.1.3 Çoklu Bit Hataları(MBU)

Uzay tasarımları daha güçlü, hafif, küçük hacim ve daha az maliyetli, aynı zamanda daha işlevsel donanımlara ihtiyaç duydukça TOE'lere karşı hassas ticari teknolojilerinin kullanımı ön plana çıkmaktadır. Bu teknolojiler yüksek hızlı CMOS elektronik ve fiberoptik cihazları kapsar. Bu teknolojileri kullanan tümleşik devreler, kompleks mikrodenetleyiciler, yüksek kapasiteli FPGA'ler, yüksek yoğunluklu SRAM'ler gibi bir çok farklı ailede yer almaktadır[8].

İşlem teknolojisi zamanla geliştikçe, düşük maliyet, düşük kaynak gerilimi ve düşük gürültü marjı özellikleri ile karakterize edilen küçük fiziksel boyutlara sahip, yüksek performanslı tümleşik devreler tasarlanmakta ve üretilmektedir. Bu değişimlerin sonucu olarak radyasyon tarafından tetiklenen geçici hatalar bellek yongalarının güvenilirliğinde giderek artan bir öneme sahip olmuştur. MBU'lar tek bir parçacık çarpması sonucu aynı anda birden fazla bellek hücresinde SEU etkisi görülmesidir ve MBU'lar aynı sözcük içinde iki ya da daha çok bitte görülen çok hücre hatası olarak tanımlanır. MBU fiziksel olgusunu açıklayabilmek için komşu hücreler tanımlanır[8]. Bir hücre için komşu hücre, o hücreyi çevreleyen ve ona temas eden 8 hücre olarak tanımlanmıştır ve şekilde gösterilmiştir. Enerji yüklü parçacık çarpması sonucu gözlemlenen TOE'nin sonucunun MBU olarak nitelendirebilmesi için hataya maruz kalan tüm bitlerin en az bir başka bitle komşu olması gereklidir. Örnek olarak bir MBU'dan etkilenen 3-bit Şekil 2.4'de verilmiştir.



Şekil 2.4 Komşu bit kavramı[8]

Geçmişte SEU etkilerinin azaltılması en çok ilgi gösterilen ve en çok çaba harcanan olguydu. Ancak SRAM bellekleri, temel yapıtaşlarının fiziksel özelliklerinin değişimiyle [9-12]'de belirtildiği üzere enerji yüklü parçacıkların tetiklediği Çoklu Bit Hatası(MBU) şeklinde görülen etkilere karşı daha hassas hale gelmektedir. MBU'lar fonksiyonel başarısızlık, görev kaybı ya da görev verilerinin kaybına neden olabileceğinden, gerekli önlemler alınmadığında MBU'lara karşı hassas bileşenlerin uzay sistemlerinde kullanımı ciddi sonuçlara yol açabilir[2].

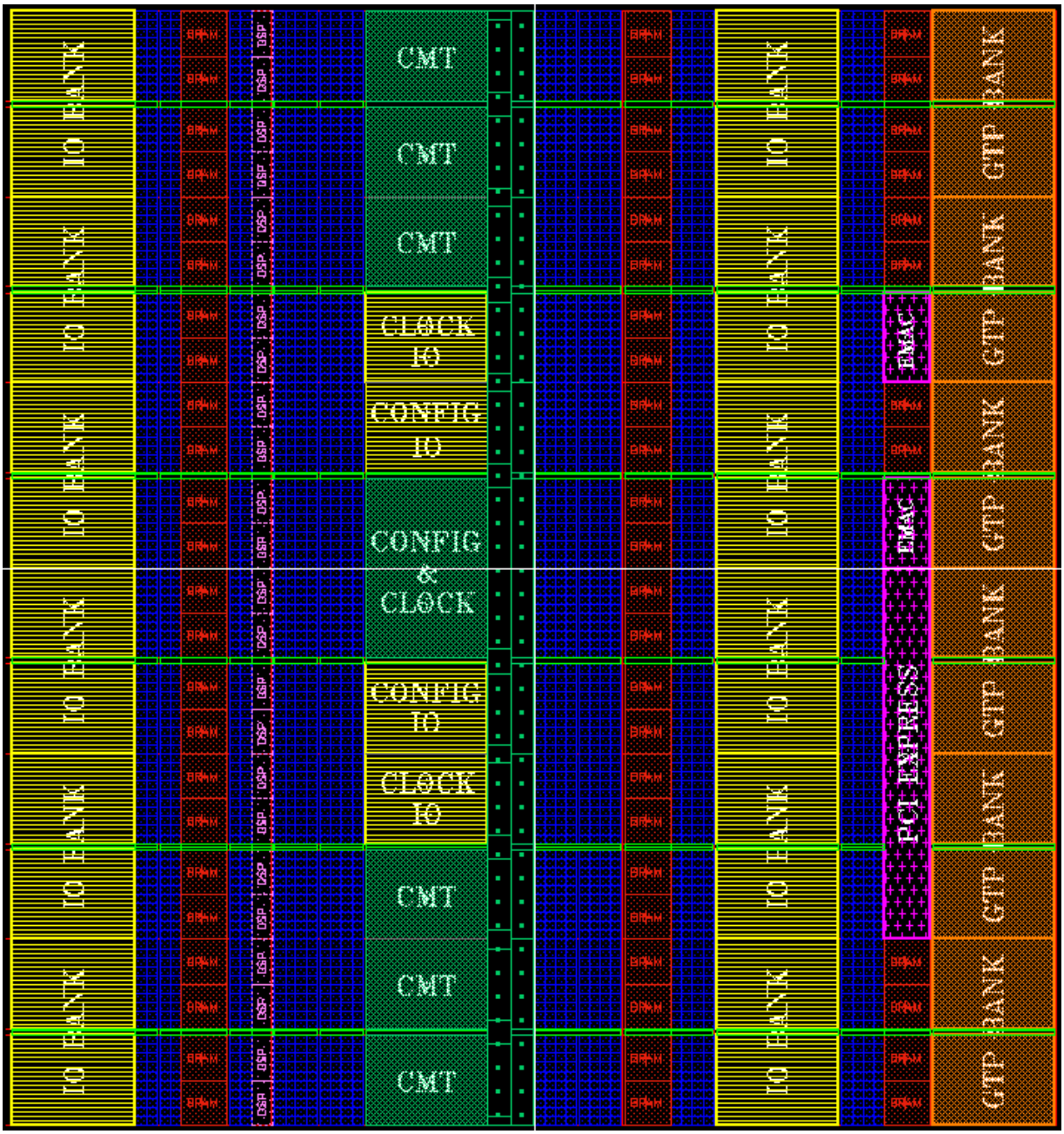
MBU'lar dört mekanizma ile ilişkilidir;

i) Iyonların hareketi boyunca yükün yayılması ve komşu SEU'ya hassas düğümler tarafından toplanması[13,14],
ii) Yüzeyin hemen altından geçen yüklü iyonun rotası üzerinde yer alan bellek hücreleri tarafından depolanan yük [15],
iii) Kontrol devresine iyon çarpması[16],
iv) Protonlar tarafından tetiklenen geri tepme ve tepki ürünleri.

2.2 FPGA mimarisi ve Yapılandırma

Uzayın radyasyon etkilerine karşı gerekli önlemler alınarak, tasarımlarda SRAM tabanlı FPGA'lerin Bölüm 1'de bahsedilen avantajlarından yararlanılması adına Virtex-5 FPGA'lar bu çalışmada önerilen yöntemler ile uzay sistemlerinde kullanıma elverişli hale getirilecektir. SRAM tabanlı FPGA'lerin radyasyon etkilerine hassasiyetlerini ve hata durumlarını açıklayabilmek ve hata etkilerini azaltıcı tespit ve düzeltme yöntemleri önerebilmek için öncelikle FPGA mimarisi ve fiziksel yapısının açıklanması faydalı olacaktır. Bu tezde donanım mimarisinde de anlatıldığı üzere Xilinx firması tarafından üretilmekte olan Virtex-5 serisi FPGA'e tasarımda yer verilmiştir. İşlem teknolojisinin gelişimiyle birlikte FPGA serimlerinde de ana hatlar genel olarak korunsa da yeni FPGA tasarımlarında farklılıklar gözlemlenebilmektedir.

Şekil 2.5'da verilen Virtex-5 serisinin fiziksel serimi, bu farklılıklar göz önüne alındığında da Virtex, Virtex-II ve Virtex-4 cihazları için de geçerli olan mimarinin gösterimi olarak kullanılabilir[1].



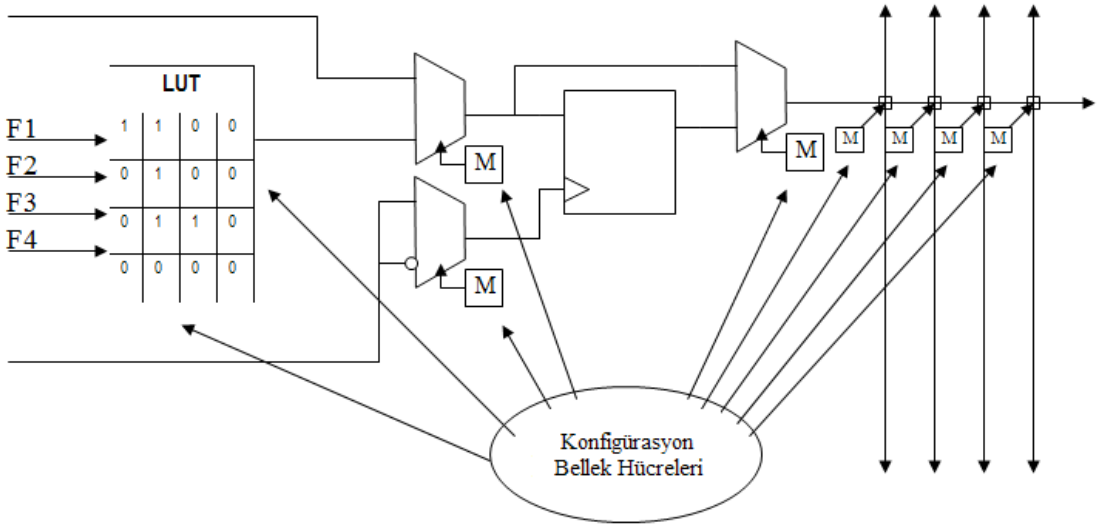
Şekil 2.5 Virtex-5 fiziksel serimi [1]

FPGA iç yapısı, fiziksel kaynaklar ve FPGA'in yapılandırılmasına [17]'de detaylıca değinilmiştir. Tüm Virtex cihazları için beş adet ana kaynak vardır: yapılandırılabilir mantık bloğu(CLB), Blok RAM(BRAM), BlokRAM bağlantı(BRAMi), giriş/çıkış bloğu(IOBs) ve saat devresi. Virtex-4 ve Virtex-5 bunlara ek olarak sayısal sinyal işleme(DSP) bloklarına sahiptir.

Şekil 2.5'da da görüldüğü üzere cihazlar sütunsal bir yapıya sahiptir ve her sütun bir adet ana kaynağa tahsis edilmiştir. Virtex-4 ve Virtex-5 serilerinde en sağ, en sol ve orta sütunlar IOB'lere ayrılırken, daha eski cihazlarda ise IOB'ler cihazların çevre sınırlarına dizilmiştir. İçte kalan sütunların çoğunda mantık işlemlerinin gerçeklendiği CLB'ler yer alır. Geri kalan sütunlar ise BRAM, BRAM bağlantı ve DSP blokları içindir. CLB'ler ardışık ve birleşimsel devrelerin gerçeklenmesi için kullanılan temel mantık kaynaklarıdır.

CLB elemanı, mantık ve aritmetik devrelerinin gerçeklendiği başvuru çizelgeleri(LUT), kullanıcı kapan devreleri, çoğullayıcılar(multiplexer) ve elde devrelerini(carry chain) barındırır. CLB'nin işleyişi yapılandırma verisi ile şekillendirilir.

Virtex-5 FPGA'leri uygulamaya yönelik yapılandırma verisinin FPGA içerisinde dağılmış olarak yer alan kurulum belleğine yazılması ile programlanır. Şekil 2.6'te CLB dilimi ve anahtarlama matrisi üzerinde bahsedilen yapılandırma belleği yerleşimi ile yapılandırma verisi ve FPGA bileşenlerinin yapılandırılması arasındaki ilişki gösterilmektedir. Bu bellekler en temel yapıda çerçeveleri oluşturur ve Virtex-5 FPGA'lerinde çerçeveler en küçük adreslenebilen yapılardır. Şekil 2.6'te de görüldüğü üzere yapılandırma verisi LUT'ların gerçeklediği fonksiyonları, çoğullayıcıları ve bağlantı matrisleri tarafından elemanlar arası bağlantıları ile direkt ilişkilidir ve bu elemanları düzenler.



Şekil 2.6 Konfigürasyon belleği - CLB ilişkisi

FPGA üzerindeki erişilebilir en küçük yapılandırma verisine çerçeve denir. Çerçeveler FPGA'in bir satırının tüm yüksekliğini kapsayan 1312-bitlik bir dik istif olarak düşünülebilir. 1312 bitlik bir çerçeve için 41 adet 32-bitlik sözcük gereklidir. FPGA içindeki tüm satırlar için bu cümle – çerçeve ilişkisi geçerlidir. Bir satır FPGA'in temel bileşenlerinin istiflenmesi sonucu meydana gelir.

Her sütun belirli bir sayıda minör adres aracılığıyla erişilen çerçeveler(frame) içerir. Sütunda yer alan çerçeve sayısı blok tipine bağlıdır. Virtex-5 serisi için yapılandırma çerçevelerini, fonksiyonları ve erişim türleri temel alınarak sınıflandırıldığında kullanılan yapılandırma adres uzayında aşağıdaki blok tipleri vardır:

- Ara Bağlantı ve Blok Yapılandırma: Yapılandırma adres uzayının tüm ara bağlantı ve CLB, DSP, IOB gibi blokların yapılandırma çerçevelerini içerir. Ayrıca BRAM'lerin yapılandırma parametreleri de bu kısımdadır (Ör: giriş-çıkış genişlikleri). BRAM'lerin içeriklerini kapsamaz.
- BRAM içeriği: BRAM'lerin güncel bellek içerikleri yer alır. BRAM'lerin içerikleri için yapılandırma çerçevelerine erişim olağan yapılandırma çerçevelerinden farklı yapılmaktadır. Yapılandırma verisinin boyutunu makul

seviyelerde tutabilmek adına BRAM içerikleri yapılandırma verilerini gerekli değilse dikkate alınmayabilir. Bu iki nedenden dolayı BRAM içerikleri için yapılandırma adres uzayında ayrı bir kısım ayrılmıştır.

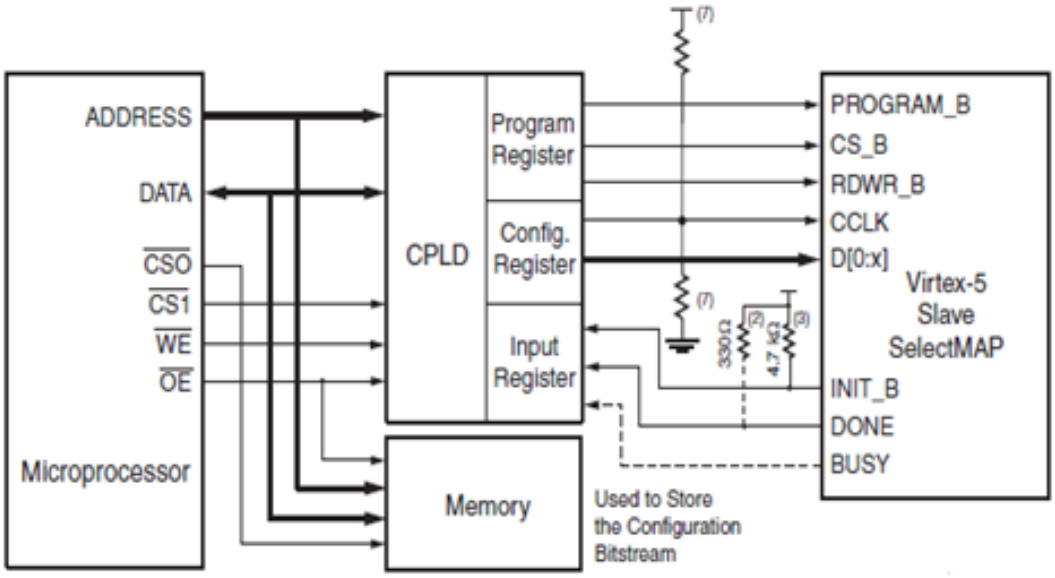
- Ara Bağlantı ve Blok Özel Çerçeveleri: kısmi yeniden yapılandırma için gerekli yapılandırma verilerini içerir.

Virtex-5 FX130T modeli toplam 37520 yapılandırma çerçevesine sahiptir. Cihazın toplam çerçeve sayısı ise 38390'dır. Her çerçeve toplamda 32-bitlik 41 sözcükten oluşmaktadır. Bu da yapılandırma dizisinin boyutunun 1538320 sözcük olduğu anlamına gelir.

Virtex-5 elemanları içerisinde bulunan programlanabilen belleklere erişim, sadece malzeme içerisinde yer alan yapılandırma kaydedicilere erişim ile sağlanmaktadır. Kullanıcı uygulaması tasarım aracı tarafından yapılandırma verisine dönüştürülür ve yukarıda bahsi geçen çerçevelere FPGA yapılandırılırken yazılır. Çerçevelere yazma ve okuma işlemleri konfigürasyon kaydedicilere gönderilen çeşitli komutlar ve her bir çerçevelerin adresi belirtilerek yapılmaktadır. Tüm bu komut, adres ve yapılandırma çerçevelerinin içerikleri yapılandırma veri dosyasını oluşturur. Çerçeve Veri Yazmaç Girişi (FRDI) çerçevelere yazma sırasında her bir 32-bitlik verinin kaydedildiği ve FPGA içersindeki mantık devreleri ile çerçevelere yazmanın yapıldığı kaydedicidir. Çerçeve Veri Yazmacı Çıkışı (FRDO) FPGA konfigürasyonu okuma sırasında okunan ve değişiklik olsun olmasın (SEU etkisi ile vs.) konfigürasyon verisini FPGA'in dışına taşıyan kaydedicidir. Çerçeve Veri Adres Yazmacı(FDRA) çerçevelere erişimi sağlayan adres kaydedicisidir. Tasarım aracı tarafından oluşturulan yapılandırma dosyası, yapılandırmanın FPGA'e yazılması işlemi için FDRI ve FDAR'ye yazılacak içerikler ile yazma komutlarını içerir. Yapılandırmanın okunması için yapılandırma dosyası FDRA'ya yazılacak içerik ve okuma komutlarını içerir.

Yapılandırma ve yapılandırma verisini okuma işlemlerini gerçekleştirmek için FDRI, FDRO ve FDRA yazmaçlarına dışarıdan farklı arayüzler ile erişmek mümkündür. Bu

arayüzlerden en yüksek hızda veri aktarımını sağlayanı SelectMap'tir. SelectMap 8-, 16- ve 32-bit veri yolu genişlikleri sunar. Arayüz veri hattı ve kontrol sinyallerinden oluşur ve SRAM arayüzüne benzer yapıdadır. SelectMap arayüzünün blok şeması Şekil 2.7'te verilmiştir.



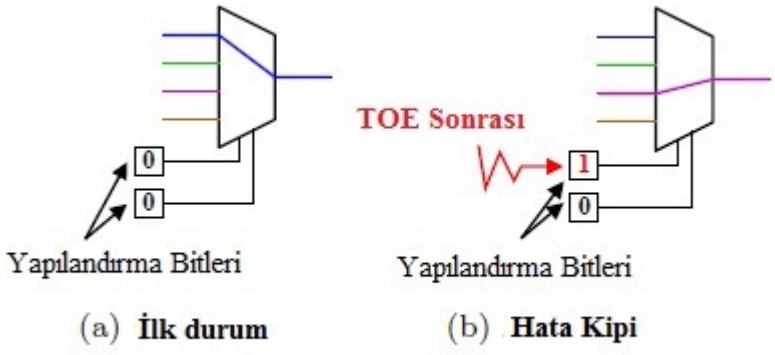
Şekil 2.7 SelectMAP blok şeması[17]

2.3 FPGA'lerde SEU ve SEFI kaynaklı hata kipleri

FPGA'ler normal ASIC devrelerinin maruz kalmadığı birçok SEU kaynaklı hata kiplerine sahiptirler. Hata kipleri genel olarak çoğullayıcılarda, iletim hatlarında, tampon devrelerinde, başvuru çizelgelerinde, denetim bitlerinde ve sabit mantık değerlerini sağlayan kilitleme devrelerinde görülen hataları içerir[7]. Bu hata kipleri yapılandırma verisinin bozunmasının sonuçları olarak ortaya çıkar.

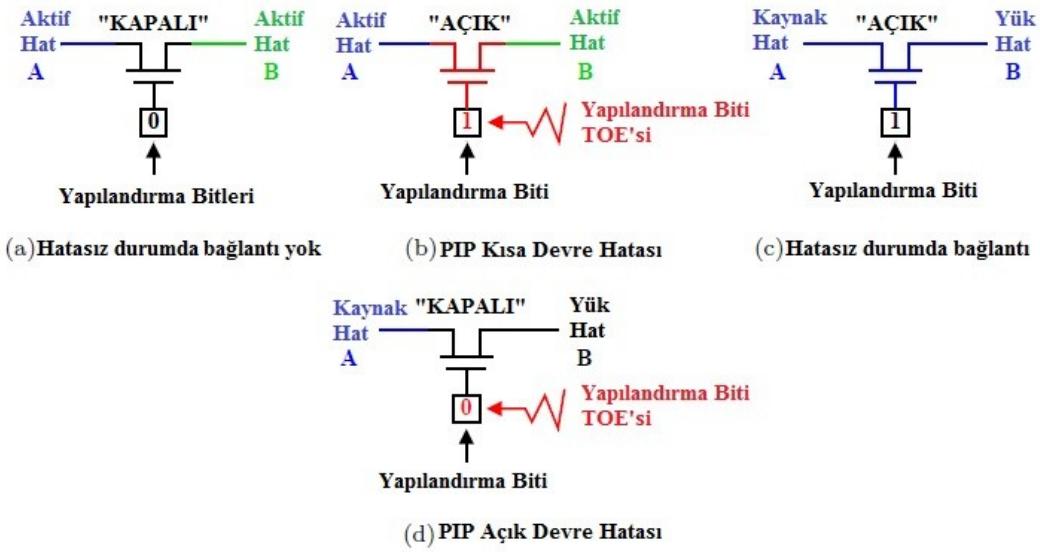
Çoğullayıcı Hata Kipi: Çoğullayıcıları denetleyen yapılandırma bitlerinin maruz kalacağı TOE sonucu farklı bir iletim yapılandırması ortaya çıkar. İstenenden farklı bir giriş çıkışa rotalanarak işlevsel bozunuma neden olur. Şekil 2.8'da bu hata kipine

bir örnek sunulmuştur. Normal işleyişte üstten birinci girişin çıkışa rotalanması hedeflenmiştir ve yapılandırma bitleri buna uygun olarak çoğullayıcıyı denetlemektedir. Yapılandırma bitlerinden biri TOE maruz kalarak istenmeyen girişin çıkışa rotalanmasına neden olur.



Şekil 2.8 Çoğullayıcı Hata Kipi[7]

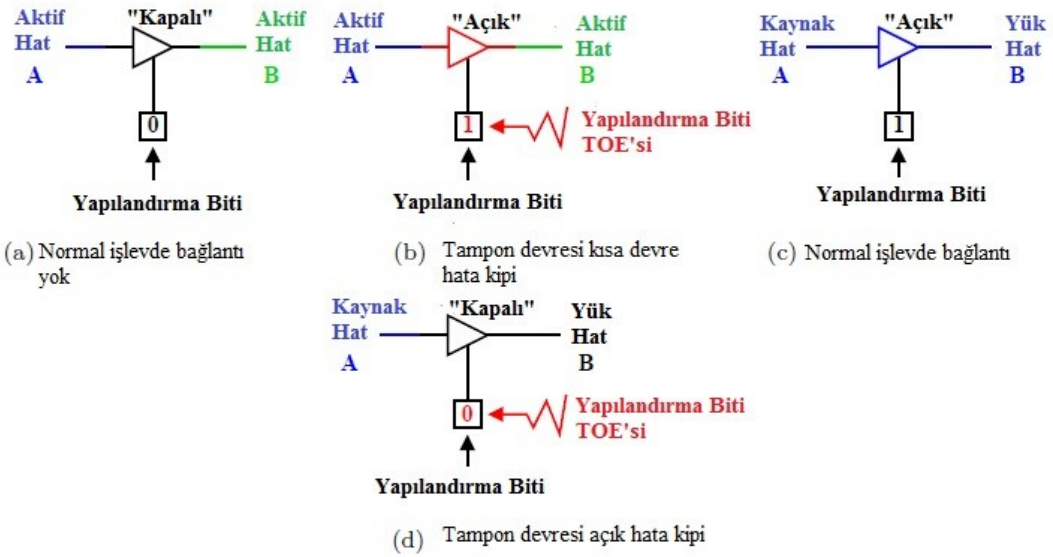
İletim Ağları Hata Kipi: Bloklar arası iletim ağları, Programlanabilir Bağlantı Noktaları (PIP) üzerinden bağlantıları sağlar. PIP'ler iki hat arasında yer alan ve açılıp kapanabilen geçiş transistorundan oluşmaktadır. Bu transistorların denetimlerini yapılandırma bitleri üstlenir. Şekil 2.9'de de görüldüğü üzere yapılandırma bitlerinde hata neden olan TOE'ler sonucu geçiş transistorlarının açılıp kapanması sonucu istenmeyen bağlantılar yapılabilir veya işlevini yürütmekte olan bağlantılar kopabilir.



Şekil 2.9 İletim Ağı Hata Kipleri[7]

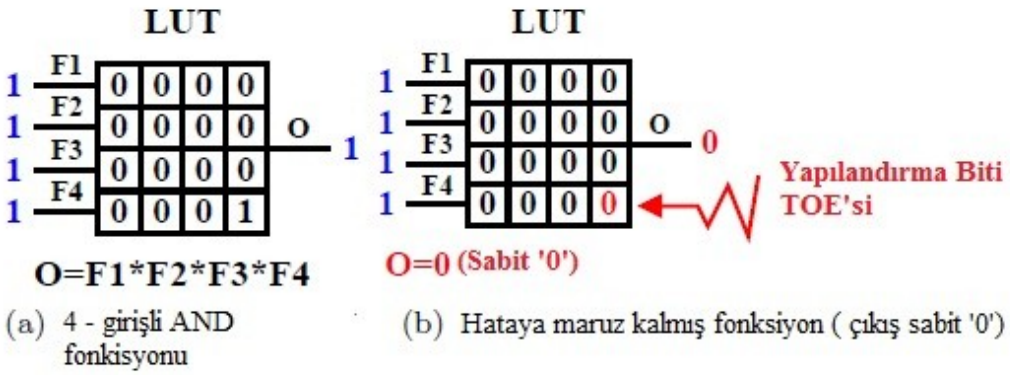
Tampon Devresi Hata Kipi: Tampon devrelerinde görülen hatalar PIP hatalarına çok benzemektedirler. Temel farkları şudur: PIP teki geçiş transistoru yerine hata aktif bir sürücü tarafından oluşmaktadır ve tek yönlüdür. PIP hata kiplerinde yapılandırma verisinde oluşan hatalar hem giriş hem çıkış tarafını etkilerken tampon devresi hata kipinde sadece çıkış tarafı etkilenmektedir. Tampon devresi hata kipleri gösterimi Şekil 2.10'de verilmiştir.

Mantık hataları: İki tip mantık hatası vardır: Başvuru çizelgesi değerlerinin değişmesi ve denetim biti değişimi.



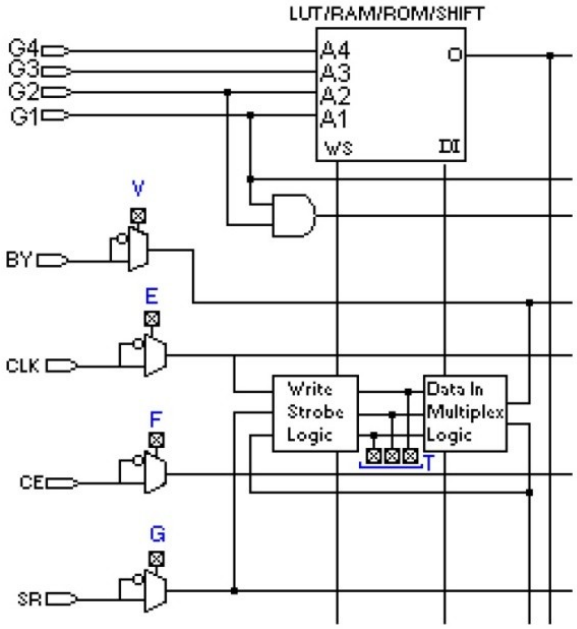
Şekil 2.10 Tampon devresi hata kipleri[7]

Virtex FPGA ailesi çoğu mantık fonksiyonlarını üretmek için başvuru çizelgelerini kullanmaktadır. Başvuru çizelgeleri FPGA içerisinde SRAM tabanlı devreler ile gerçeklendiğinden TOE'lere karşı hassastır. Başvuru çizelgeleri içerik değerlerinin hatalara maruz kalarak değişmesi sonucu gerçeklenen fonksiyonlardan beklenmeyen sonuçlar elde edilir. Örnek olarak bir çarpma fonksiyonu ele alınmıştır. Başvuru çizelgesinin içeriğinin değişmesi sonucu çarpma devresi, sabit '0' veren bir mantık devresine dönüşmüştür.



Şekil 2.11 LUT Hata Kipi[7]

Denetim Bitleri Hata Kipleri: Virtex mimarisini oluşturan temel birimlerden olan CLB ve IOB birimleri çeşitli görevleri oluşturmak için yapılandırma verisi tarafından ayarlanan denetim bitlerini kullanmaktadır. Şekil 2.12'de verilen örnekte V, E, F, G ile gösterilen bitler yapılandırma verisi ile programlanabilen evirici bitleridir. Burada oluşabilecek bir bit bozunması (SEU) ile yanlış veri seçilmiş olacaktır. Aynı şekilde T ile gösterilen ve LUT' un LUT, 16x1 çift portlu RAM, 32x1 RAM veya kaydırmalı kaydediciden biri olarak görev yapmasını belirleyen bitlerde oluşan bozunma yanlış kipli işlevselliğin seçilmesine sebebiyet verecektir.

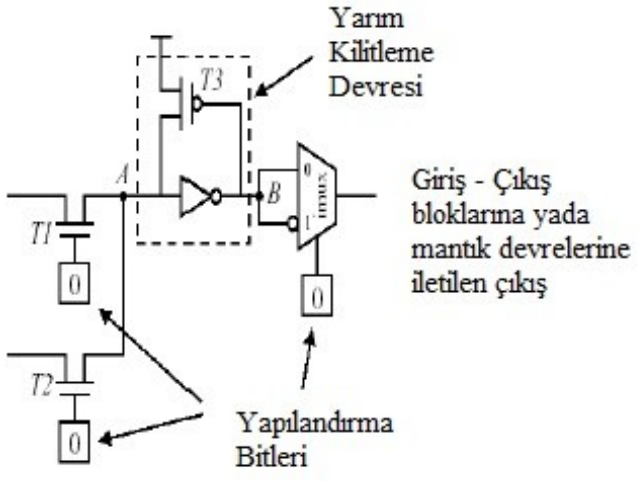


Şekil 2.12 Denetim Bitleri Hata Kipleri[7]

Kullanıcı Bellek Hata Kipi: Kullanıcı hafızası olarak kullanılan (BRAM), LUT tabanlı RAM'ler, CLB içinde yer alan kapan devreleri ve I/O blok kapan devreleri (I/OB-FF) TOE'lere açıktır. Bu kaynaklarda meydana gelen bozukluklar, FPGA'in programlanma verisini inceleyerek kolaylıkla tespit edilemez ve düzeltilemez. Bozukluğun tespit edilebilmesi için doğru değerlerin bilinmesi gerekir

Mantık Sabitleri Hata Kipleri: Xilinx mimarisinde sabit '0' ve '1' mantık değerlerini sağlamak için toprak ve V_{cc} sinyallerine erişim sağlanmamaktadır. Logic sabitleri genellikle geçici olarak sağlanmaktadır.

Xilinx FPGA'lerinde sabit mantık değerlerini elde etmek için iki yöntem izlenir. İki yöntemde TOE'lere açıktır. Birinci yöntemde FPGA içerisinde sabit "0" ve "1" değerleri yarım kilitleme devresi adı verilen yapılar tarafından sağlanabilir. Yarım kilitleme devreleri TOE'lere karşı hassastır. Bu devreler tasarımcı tarafından doğrudan tasarıma yerleştirilmemektedir ve programlama verisi tarafından kontrol edilmemektedir. Bu özellikler yarım kilitleme devrelerin gözlemlenebilirliğini ve düzenlenmesini kısıtlar. Devreler sadece yeniden yapılandırma sırasında düzeltilebilir, yapılandırma ve veri sürtmesiyle(scrubbing) bu mümkün değildir. Yarım kilitleme devrelerinde oluşan bozunmalar sonucu sabit değerlerde meydana gelebilecek değişikler sonucu işlev kaybı veya bozukluğu muhtemeldir.



Şekil 2.13 Xilinx Yarım Kilitleme Devresi[7]

Sabit mantık değerleri sağlamak için kullanılan yaklaşımlardan biri de LUT'lar kullanarak bu değerleri oluşturmak ve FPGA içinde ihtiyaç olan kısımlara yöneltmektir. Bu LUT'lerde bitlerin değer değiştirmesi sonucu bu tarz hatalar

meydana gelir. Ancak, LUT'ler için alınan önlemlerle bu tarz hataların etkileri azaltılabilir.

*Tek Olay Fonksiyonel Bozuklukları:*Yapılandırma devresi veya reset denetleyicisi gibi FPGA'in kontrol elemanlarında meydana gelen bozukluklar sonucu oluşur.

FPGA içinde TOE'lerden etkilenebilecek birçok konfigürasyon ve kontrol kaydedici bulunmaktadır. Bu kaydedicilerde meydana gelebilecek TOE'ler cihazın işlevlerini düzgün olarak yerine getirememesine neden olur. Bu TOE'ler Tek Olay Fonksiyonel Bozuklukları(TFOB) olarak sınıflandırılır. Virtex ailesini etkileyen TFOB'lere örnek olarak JTAG TAP denetleyici bozuklukları, SelectMAP denetleyici bozuklukları ve açılış denetleme devresi bozuklukları verilebilir.

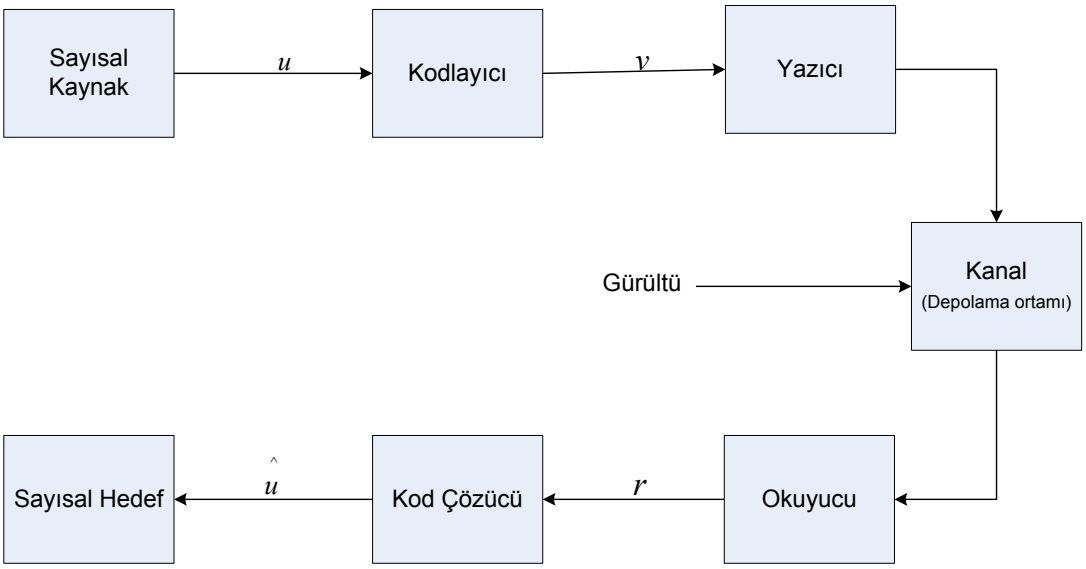
2.4 Hata Düzeltme Kodları

Bu bölümde hata etkilerini azaltıcı önlem olarak önerilen Hata Düzeltme Kodlarının(HDK) genel özelliklerine değinilecektir. HDK'lerle ilgili detaylı bilgiler [18]'da verilmiştir.

Verinin depolandığı veya iletildiği ortam çevresel etmenler, çevresel girişim ve ortamın fiziksel özellikleri nedeniyle rastgele bit hatalarına açıktır. Hata kodlama bu hataları tespit etmeyi ve düzeltmeyi kapsayan, verinin kaynaktan hedefe kadar bozulmamış şekilde iletilmesini sağlayan bir yöntemdir. Hata kodlama, bilgisayar uygulamalarında hata korumalı hesaplama, manyetik, optik ve katı-hal veri depolama ortamlarında, uydu ve derin uzay haberleşmesi alanlarında sıkça kullanılmaktadır.

Ticari ve askeri uygulamalarda veri transferi, verinin işlenmesi, veri depolamanın daha büyük ölçekte ve daha yüksek hızlarda gerçekleştirilmesi sonucu artan hata oranları nedeniyle güvenilir ve verimli veri depolama sistemleri ön plana çıkmaktadır..

Shannon, veri oranı kanal kapasitesinden düşük olduğu sürece gürültüye maruz kalan kanal veya depolama ortamında veride meydana gelen hataların istenen seviyeye düşürülebileceğini [19]'da açıklamıştır. Yöntem olarak ise gürültülü kanallarda kullanılacak hata denetimi için hata düzeltme kodlarını işaret etmiştir. Hata denetimi için kodlama kullanımı modern haberleşme ve sayısal veri depolama sistemlerinin temel bileşenlerinden biri olarak tasarımlarda yer almaktadır. Şekil 2.14'de günümüzde geçerli olan veri depolama/iletim modeli verilmiştir.



Şekil 2.14 Veri iletimi / Veri depolama sistemi blok şeması

Veri depolama ve veri iletimi birçok ortak özelliğe sahiptir. İki süreçte de veri bir kaynaktan bir hedefe taşınır. Tipik bir veri depolama / iletim sistemi Şekil 2.14'de görüldüğü şekliyle temsil edilebilir. Sayısal kaynak bir insan veya bir makine(ör: sayısal bilgisayar, tasarımda bir fonksiyon, süreç) olabilir. Kodlayıcı, u veri dizisini kod sözcüğü(codeword) adı verilen v kodlanmış veri dizisine dönüştürür. *Yazıcı* kodlayıcının çıktılarını kanalda veri aktarımı / depolanması için uygun dalga biçimi yapısında düzenler. Dalga biçimi kanala/depolama ortamına girer ve gürültü nedeniyle bozunmalara uğrar. Tipik aktarım kanalları telefon hatları, mobil iletişim ağları, fiber optik ağlar, HF telsizler, telemetri, mikrodalga ve uydu linkleridir.

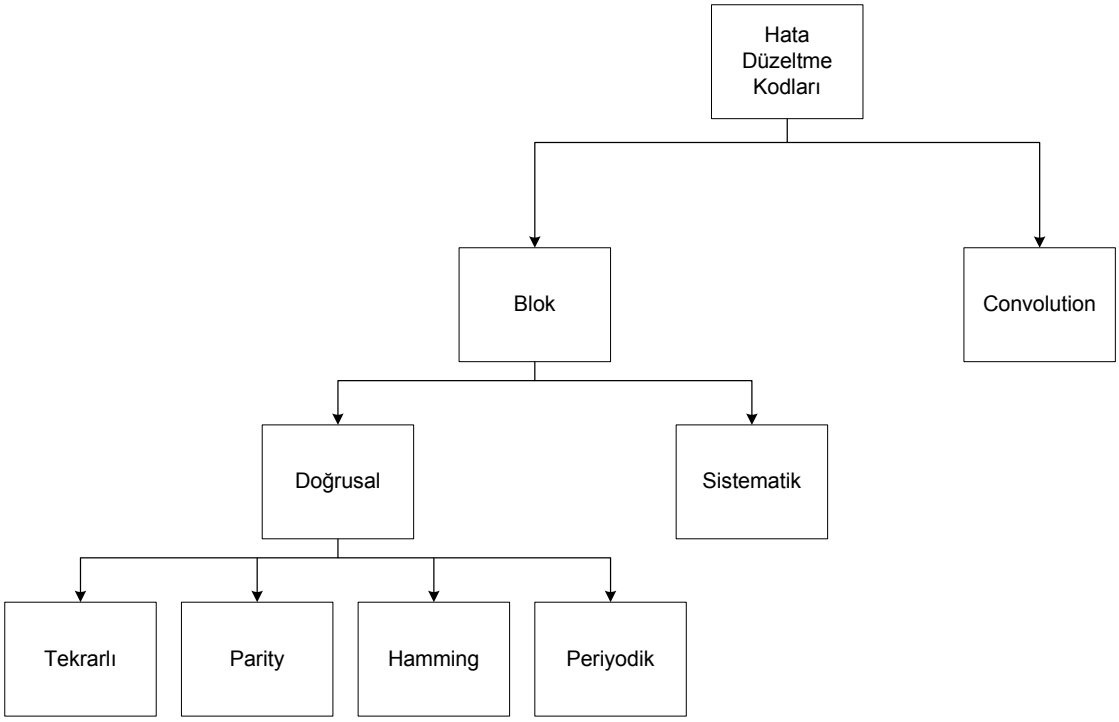
Depolama ortamı ise katı hal yarı iletken hafızalar, manyetik depolama birimleri, harddisk, kompakt disk, optik veri depolama birimleri vb. içerir. Her bir örnek için farklı tür ve/veya türlerde etki gösteren gürültü kaynağı bulunmaktadır. Okuyucu kanaldan / depolama ortamında veriyi T zaman aralığı için toplar ve kod çözücüde kullanılabilir yapıda düzenleyerek(r) kod çözücüye iletir. Kod çözücüalınmış veri dizisi r'yi kestirilmiş veri dizisi $\hat{u}$'ya dönüştürür. Kod çözme işlemi kodlayıcı ve kanalda gözlemlenen gürültü temel alınarak oluşturulan kurallara göre gerçekleştirilir. İdeal koşullar altında $\hat{u}$ ile u'nun birebir aynı olması beklenir, ancak gürültü nedeniyle kod çözme hataları gözlemlenebilir.

Veri kodlama teorisi şu özelliklere sahip kodlayıcı ve kod çözücülerin tasarlanması üzerine odaklanır[18]:

1) Verinin gürültülü ortamda mümkün olduğunca hızlı ve yoğun şekilde depolanması/aktarılması,
2) Verinin kod çözücü çıktısından güvenilir olarak yeniden oluşturulabilmesi,
3) Kodlayıcı ve kod çözücünün tasarım maliyetinin kabul edilebilir sınırlar içinde yer alması,
4) Kod çözme hatası olasılığının minimum olması.

2.4.1 Kod Türleri

Günümüzde bahsi geçen uygulama alanlarında sıklıkla kullanılan HDK'ler Şekil 2.15'te verilmiştir. Bu çalışmada gerçeklenen EG-LDPC kodların yer aldığı Blok Kodların genel özelliklerine bu bölümde değinilecektir.



Şekil 2.15 Hata Düzeltme Kodları'nın Türleri

2.4.2 Blok Kodlar

Kodlayıcı veri dizisini k sayısında veri bitinden (sembol) oluşan bloklara böler. Bir mesaj bloğu mesaj adı verilen $u = (u_0, u_1, \dots, u_{k-1})$ ikili k-çokuzlusuyla(binary n-tuple) ifade edilir. Birbirinden bağımsız 2^k tane mesaj oluşturmak mümkündür. Kodlayıcı her u mesajını bir $v = (v_0, v_1, \dots, v_{n-1})$ ikili n-çokuzlusuna dönüştürür. Dönüşüm sonucu elde edilen v kod sözcüğü (codeword) olarak adlandırılır. 2^kelde edilebilir birbirinden farklı mesaj için 2^k farklı kod sözcüğü üretilebilmektedir. Bu n uzunluğundaki 2^k kod cümlesinin oluşturduğu kümeye (n, k) blok kodu denir. Kod oranı her bir oluşturulan kod sözcüğü için kodlayıcıya giren veri biti sayısı olarak tanımlanır ve şöyle ifade edilir:

$$R = \frac{k}{n} \tag{2.2}$$

Kodlayıcının çıktısı olan n-bitli kod sözcüğü sadece ve sadece karşılık gelen k-bitli mesaja bağımlıdır. Her mesaj bloğu diğer bloklardan bağımsız olarak kodlanır. Kodlayıcı hafızasızdır ve birleşimselmantık devreleriyle gerçeklenebilirler.

Bir kodun işe yarar olması, her bir mesaja karşılık bir kod cümlesi üretilebilir olması için, $r \leq n$ ya da $R \leq 1$ olmalıdır. $k < n$olduğunda, mesaja *n-k* artıklık (redundant) biti eklenerek kod sözcüğü oluşturulur. Bu *n-k* artıklık biti, koda gürültülü kanalda hatalara karşı mücadele yeteneği kazandırır. Veriyi gürültülü kanal üzerinde güvenilir şekilde depolamak/aktarmak için artıklık bitlerinin mesaja ne şekilde ekleneceği kodlayıcı tasarımının ele aldığı temel konudur.

2.4.2.1 Doğrusal Blok Kodlar

n uzunluğunda 2^k adet kod sözcüğüne sahip bir blok kod, sadece ve sadece 2^k kod cümlesi GF(2) üzerinde tanımlı n-çokuzluların oluşturduğu bir vektör uzayının k-boyutlu bir altkümesini oluşturursa *(n,k) doğrusal kodu* olarak adlandırılır. Eğer iki kod sözcüğünün 2 tabanında toplamı yine bir kod sözcüğü ise o blok kod doğrusaldır.

(n,k) doğrusal kodu C, tüm ikili n-çokuzluların oluşturduğu V_n vektör uzayının k-boyutlu altkümesi olduğu için, $0 \leq i < k$ için $u_i = 0\ veya\ u_i = 1$ koşulunu ve aşağıdaki koşulu sağlayan *k* tane $g_0, g_1, \dots, g_{k-1}$ doğrusal bağımsız kod sözcüklerini bulmak mümkündür;

$$\boldsymbol{v = u_0 g_0 + u_1 g_1 + \cdots + u_{k-1} g_{k-1}} \tag{2.3}$$

k tane doğrusal kod cümlesi $k\ x\ n$ matrisin satırları olarak düzenlenir:

$$G = \begin{bmatrix} g_0 \\ g_1 \\ . \\ . \\ . \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} g_{00}g_{01}g_{02} & \cdots & g_{0,n-1} \\ g_{10}g_{11}g_{02} & \cdots & g_{1,n-1} \\ & . & \\ & . & \\ & . & \\ g_{k,0}g_{k,1}g_{k,2} & \cdots & g_{k,n-1} \end{bmatrix} \tag{2.4}$$

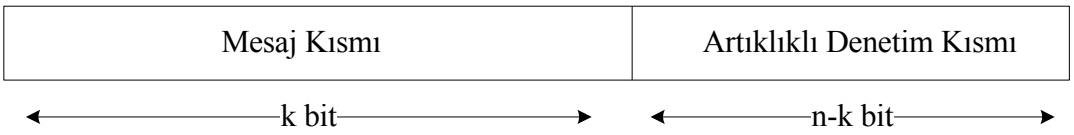
$u = (u_0, u_1, \dots, u_{k-1})$ kodlanacak mesaj ise, karşılık gelen kod cümlesi şöyledir:

$$\begin{aligned} v &= u.G \\ &= (u_0, u_1, \dots, u_{k-1}) \cdot \begin{bmatrix} g_0 \\ g_1 \\ . \\ . \\ . \\ g_{k-1} \end{bmatrix} \\ &= u_0 g_0 + u_1 g_1 + \cdots + u_{k-1} g_{k-1} \end{aligned} \tag{2.5}$$

G 'nin satırları *(n,k) C* doğrusal kodunu üretir. Bu nedenle *G* matrisi, *C* doğrusal kodunun üreteç matrisi olarak adlandırılır. *(n,k)* doğrusal kodunun herhangi *k* sayıda doğrusal bağımsız kod cümlesi kodun üreteç matrisini oluşturmak için kullanılabilir. Bir *(n,k)* doğrusal kod üreteç matris *G* 'nin *k* sayıdaki satırıyla tamamen belirtilebilir. Doğrusal blok kodlar üreteçve eşlik denetimimatrisleriyle tanımlanır.

Sistematik yapıdaki doğrusal blok kodlarda kod cümlesi mesaj kısmı ve artıklıklı doğrulama kısmı olarak Şekil 2.16'te gösterildiği üzere doğrudan ikiye parçaya ayrılabilir. Sistematik kodların üreteç devreleri sistematik olmayan kodların üreteç devrelerine göre daha karmaşık olmayan donanım bileşenleriyle gerçeklenebilir. Bu

nedenle sistematik yapı kodlarda bulunması tercih edilen bir özelliktir. Mesaj kısmı *k* değiştirilmemiş veri bitinden, artıklıklı doğrulama kısmı ise veri bitlerinin doğrusal toplamları sonucu elde edilen *n-k* eşlik denetimi bitlerinden oluşur.



Şekil 2.16 Sistematik Kod Sözcüğü

Doğrusal sistematik (n, k) kodu aşağıdaki yapıdaki $k\ x\ n$ boyutlarındaki G matrisiyle tümüyle belirtilebilir:

$$G = \begin{bmatrix} g_0 \\ g_1 \\ . \\ . \\ . \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} 1\ 0\ 0 & \dots & 0\ p_{00}p_{01}p_{02} & \dots & p_{0,n-1} \\ 0\ 1\ 0 & \dots & 0\ p_{10}p_{11}p_{12} & \dots & p_{1,n-1} \\ 0\ 0\ 1 & \dots & 0\ p_{20}p_{21}p_{22} & \dots & p_{2,n-1} \\ & & . & & \\ & & . & & \\ 0\ 0\ 0 & \dots & 1\ p_{k,0}p_{k,1}p_{k,2} & \dots & p_{k,n-1} \end{bmatrix} \tag{2.6}$$

$$G = [I_{k\ x\ k} \mid P_{k\ x\ n}]$$

$u = (u_0, u_1, \dots, u_{k-1})$ kodlanacak mesaj ve karşılık gelen kod cümlesi:

$$\begin{aligned} v &= \left(v_0, v_1, v_{2,}\ \dots, v_{n-1}\right) \\ &= (u_0, u_1, \dots, u_{k-1})\ .\ G \end{aligned} \tag{2.7}$$

v'nin bileşenleri:

$$v_i = u_i, \qquad 0 \leq i < k \quad (2.8)$$

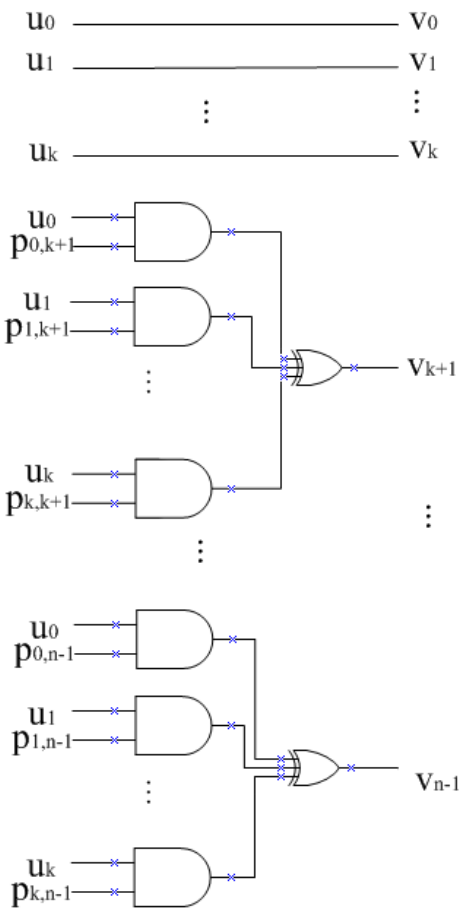
$$v_j = u_0 p_{0j} + u_1 p_{1j} + \ \ldots + u_{k-1} p_{k-1,j}, \qquad n - k + 1 \leq j < n \quad (2.9)$$

Üstteki denklemlerden anlaşılacağı üzere *v* kod cümlesinde en solda kalan *k* sayıdaki bit kodlanan $u_0, u_1, \ldots, u_{k-1}$ veri bitleriyle aynı, kalan *n-k* artıklık biti ise veri bitlerinin doğrusal toplamlarında oluşmaktadır. *b* denkleminde verilen *n-k* tane denklem eşlik denetim denklemleri olarak tanımlanır.

Doğrusal sistematik kodların kodlayıcı devresi temel mantıksal kapılar kullanılarak kolayca tasarlanabilir. Devre tasarımında 2 tabanında toplayıcılar ve AND mantıksal kapıları kullanılabilir. 2 tabanında toplayıcıların donanımsal karşılığı olarak XOR mantık kapıları kullanılmaktadır. Şekil 2.17'te örnek bir kodlayıcı devresi verilmiştir.

2.4.3 Sendrom ve Hata Tespiti

Üreteç matris *G* ve eşlik denetim matrisi *H* ile temsil edilen bir (n, k) doğrusal koduyla elde edilen ve gürültülü bir kanaldan geçirilen $v = (v_0, v_1, v_{2,} \ldots, v_{n-1})$ kod sözcüğünü ele alalım. Kanaldaki gürültü nedeniyle elde edilen $r = (r_0, r_1, r_{2,} \ldots, r_{n-1})$ vektörü *v*'den farklı olabilir.



Şekil 2.17 Kodlayıcı Devresi

$$e = r + v$$
$$= (e_0, e_1, e_2, \ldots, e_{n-1}) \tag{2.10}$$

e vektör toplamı;

- $r_i \neq v_i$ise$e_i = 1$,
- $r_i = v_i$ise$e_i = 0$ koşullarını sağlayan bir n-çokuzlusu oluşturur ve hata vektörü olarak adlandırılır.

Hata vektörü gönderilen kod sözcüğüyle alınan vektör arasındaki bit farklılıklarının pozisyonları bilgisini içerir. *e* vektöründeki her '1' biti kanaldaki gürültü nedeniyle oluşan hataları temsil eder. Alınan vektörün, kod sözcüğü ve hata vektörünün vektör toplamı olduğunu (2.10) formülünden çıkarabiliriz;

$$r = v + e \tag{2.11}$$

r vektörü alındığında, alıcı *v* ve *e*'yi ayrı ayrı bilemez. *r* alındığında ilk olarak hatanın mevcut olup olmadığı tespit edilmelidir. Hatanın varlığı tespit edildiğinde ya hataların kod çözücü tarafından düzeltilmesi gerekir ya da göndericiden *v*'nin yeniden gönderilmesi talep edilir.

v alıcıya ulaştığı zaman kod çözücü sendromu hesaplar:

$$\begin{aligned} s &= r \,.\, H^T \\ &= \left(s_0, s_1, s_2, \ \ldots, s_{n-k-1}\right) \end{aligned} \tag{2.12}$$

Sadece ve sadece *r* kod cümlesiyle aynı ise $s = 0$'dir ve *r* kod cümlesi değil ise $s \neq 0$'dir. Böylece $s \neq 0$ olduğunda *r*'nin kod cümlesi olmadığı anlaşılır ve *r*'de bit hatalarının olduğu tespit edilir. $s = 0$ olduğunda ise alıcı r kod cümlesi olarak kabul eder.

Bazı durumlarda belirli hata vektörlerindeki hataların tespit edilmesi mümkün olmayabilir (ör: r'de hatalar mevcut ancak $s = r \,.\, H^T = 0$). Bu durum hata örüntüsünün sıfır olmayan bir kod cümlesiyle aynı yapıda olduğu durumlarda gözlemlenir. Bu yapıdaki hata örüntülerine tespit edilemeyen hata örüntüsü adı verilir. 2^{k-1}tane sıfır olmayan kod cümlesi olduğu için 2^{k-1} tane tespit edilemeyen

hata örüntüsü mevcuttur. Tespit edilemeyen bir hata örüntüsü oluştuğunda, bu kod çözücüde kod çözme hatasına neden olur.

H eşlik denetim matrisinin satırları kodun eşlik denetim denklemlerinin katsayıları ile oluşturulur ve aşağıdaki biçimde gösterilebilir. *H*'yi üretme yöntemlerinden daha sonra bahsedilecektir.

$$H = \begin{bmatrix} p_{00}p_{01} & \cdots & p_{0,n-1} \\ p_{10} & p_{11} & \cdots & p_{1,n-1} \\ & & . & \\ & & . & \\ & & . & \\ p_{n-k-1,0}p_{n-k-1,1} & \cdots & p_{n-k-1,n-1} \end{bmatrix} \tag{2.13}$$

Olduğunda, sendrom bitleri şu şekilde ifade edilir:

$$\begin{aligned} s_0 &= r_0p_{00} + r_1p_{01} + \ \ldots + r_{n-1}p_{0,n-1} \\ s_1 &= r_0p_{10} + r_1p_{11} + \ \ldots + r_{n-1}p_{1,n-1} \\ &. \\ &. \\ &. \\ s_{n-1} &= r_0p_{n-k-1,0} + r_1p_{n-k-1,1} + \ \ldots + r_{n-1}p_{n-k-1,n-1} \end{aligned} \tag{2.14}$$

Yukarıdaki formüllerden de anlaşılacağı üzere sendrom bitleri alınan vektör *r* ve veri vektörlerinden elde edilen eşlik denetim bitlerinin doğrusal toplamalarıyla elde edilir. Kodlayıcı devresiyle benzer yapıdadır ve kodlayıcı devresinde de olduğu gibi temel mantık kapıları (XOR ve AND) kullanılarak gerçeklenebilir.

2.4.4 Kod Özellikleri

Kodların hata tespit etme ve düzeltme yetenekleriyle ilgili fikir sahibi olmak ve kodları birbiriyle karşılaştırma esnasında kullanılabilecek iki temek özellikten bahsedilir:

- Blok kodun minimum uzaklığı,
- Blok kodun hata tespit etme ve çözme kapasitesi.

2.4.4.1 Blok kodun Minimum Uzaklığı

Bu parametre bir kodun rastsal hataları tespit edebilme ve düzeltme kapasitesini belirler. *v*, $v = (v_0, v_1, v_2, \ldots, v_{n-1})$şeklinde bir n-çokuzlusu olarak tanımlandığında, *v*'nin Hamming ağırlığı, *v* vektöründeki sıfır olmayan bileşen sayısı olarak tanımlanmıştır ve *w(v)* olarak gösterilir. Örnek olarak $v = (1\ 0\ 1\ 1\ 0\ 1)$'in Hamming ağırlığı 4'tür. *v* ve *w* iki n-çokuzlusu arasındaki Hamming uzaklığı iki n-çokuzlu arasında farklı olan bit sayısı olarak tanımlanır ve *d(v,w)* olarak belirtilir. $v = (1\ 0\ 1\ 1\ 0\ 1)$ ve $w = (0\ 0\ 1\ 1\ 0\ 0)$ arasındaki Hamming uzaklığı, *d(v,w)*= 2'dir. Hamming uzaklığı üçgen eşitsizliğini sağlayan metrik bir fonksiyondur. v,w ve x n-çokuzluları için;

$$d(v,w) + d(w,x) \geq d(v,x) \tag{2.15}$$

Hamming uzaklığının tanımı ve 2 tabanında toplamanın tanımından yola çıkarak *v* ve *w* arasındaki Hamming uzaklığının toplamı, *v* ve *w*'nin toplamının Hamming ağırlığına eşittir.

$$d(v,w) = w(v+w) \tag{2.16}$$

Bir *C* blok kodu verildiğinde, *C*'nin minimum uzaklığı d_{min}şeklinde gösterilir ve şöyle tanımlıdır:

$$d_{min} \triangleq \min\{d(v,w): v,w \in C, v \neq w\} \tag{2.17}$$

Cdoğrusal bir blok kodu ise, iki kod sözcüğünün toplamı yine bir kod cümlesidir. (**2.8**)'e dayanarak C'deki iki kod cümlesinin arasındaki Hamming uzaklığı üçüncü bir kod sözcüğünün Hamming ağırlığına eşittir.

$$\begin{aligned} d_{min} &\triangleq \min\{w(v,w): v,w \in C, v \neq w\} \\ &= \min\{w(x): x \in C, x \neq 0\} \\ &\triangleq w_{min}. \end{aligned} \tag{2.18}$$

$w_{min} \triangleq \{w(x): x \in C, x \neq 0\}$parametresi C doğrusal kodunu minimum ağırlığı olarak adlandırılır.

Teorem 1: Bir doğrusal kodun minimum uzaklığı o kodun sıfıra eşit olmayan kod cümlelerinin minimum ağırlığına eşittir.

Teorem 2: *C* bir *(n,k)* doğrusal kod ve *H*, *C*'nin eşlik denetim matrisi olsun. Hamming ağırlığı l olan her kod cümlesi için, *H* matrisinin l tane kolonu vektör toplamlarının sıfır vektörüne eşit olma koşuluna sağlar. Diğer taraftan, eğer *H* matrisinin vektör toplamı sıfır olan l tane kolonu var ise, *C*'de Hamming ağırlığı l olan bir kod cümlesi mevcuttur.

Aşağıdaki sonuçları Teorem 2'den çıkarmak mümkündür:

Sonuç 2.1: *H* eşlik denetimine sahip *C* doğrusal kodu için, *H*'nin *d-1* veya daha az sayıda kolonunun toplamı *0* değilse, *C* kodu en az *d* minimum ağırlığına sahiptir.

Sonuç 2.2: *H* eşlik denetimine sahip *C* doğrusal kodu için, C'nin minimum ağırlığı H'nin toplamı sıfır olan en az sayıdaki kolonlarının sayısına eşittir.

Aşağıdaki (7,4) lük bir kodun eşlik denetim matrisini şöyle olsun;

$$H = \begin{bmatrix} 1\,0\,0\,1\,1\,0\,1 \\ 0\,1\,0\,0\,1\,1\,0 \\ 0\,0\,1\,1\,1\,1\,0 \end{bmatrix} \tag{2.19}$$

Yukarıdaki H matrisini incelendiğinde, H'nin tüm sütunlarının sıfıra eşit olmadığı görülmektedir. Bu nedenle hiçbir iki sütunun toplamı sıfır vermez. Sonuç 2.1'e göre bu kodun minimum ağırlığı en az 3'tür. Ancak, sıfırıncı, ikinci ve üçüncü kolonların toplamı sıfırdır. Bu kodun ekteki kod tablosuna bakıldığında minimum ağırlığının 3 olduğunu görürüz ve Teorem 1'den minimum uzaklığın 3 olduğu sonucuna ulaşırız.

Sonuç 2.1 ve Sonuç 2.2 genelde minimum uzaklığın belirlenmesinde ya da bir doğrusal kodda minimum uzaklık için alt sınır konmasında kullanılır.

2.4.4.2 Blok Kodun Hata Tespit Etme ve Çözme Kapasitesi

v kod cümlesi gürültülü bir kanaldan geçtiğinde; *l* sayıda hataya sahip bir hata örüntüsü, alınan *r* vektörünün yollanan kod cümlesinden *l* sayıdaki bit sayısının değişmesine neden olur. Kısaca, *d(v,r)=l*'dir. C blok kodunun minimum uzaklığı d_{min} ise, *C*'nin iki ayrık kod cümlesi en az d_{min} sayıdaki pozisyonda farklılık gösterir. Bu *C* kodu için, hiçbir $d_{min} - 1$ veya daha az sayıda hataya sahip hata örüntüsü bir kod cümlesini bir başkasına çeviremez. Bu nedenle $d_{min} - 1$ ya da daha az sayıda hata içeren bir örüntü C'de bir kod cümlesi olmayan bir r vektörüne neden olur. Alıcı alınan r vektörünün bir kod cümlesi olmadığını tespit ettiğinde, hataları

tespit etmiş olur. Sonuç olarak, d_{min} minimum uzaklığına sahip bir blok kod, $d_{min} - 1$ veya daha az sayıdaki hataların tüm kombinasyonlarını tespit edebilir. Ancak, en az bir çift kod cümlesi d_{min}sayıda pozisyonda değişik bitlere sahip olduğundan ve d_{min}sayıdaki oluşacak hata bir kod cümlesini başka bir kod cümlesine çevireceğinde d_{min} sayıdaki hatalar tespit edilemez. Aynı argüman d_{min}'den fazla sayıda hata için de geçerlidir. Sonuç olarak, minimum uzaklığı d_{min}olan bir kodun rastgele hata tespit kapasitesi $d_{min} - 1$'dir.

Minimum uzaklığı d_{min} olan bir kod, $d_{min} - 1$ ya da az sayıda hata içeren tüm hata örüntü kombinasyonlarını tespit edebileceğini garanti etse de, d_{min} ya da daha fazla sayıdaki hataların büyük bir oranını tespit edebilme kapasitesine sahiptir. *(n,k)* doğrusal kodu *n* uzunluğunda $2^n - 2^k$ sayısındaki hata örüntülerini tespit etme kabiliyetine sahiptir. $2^n - 1$olası sıfıra eşit olmayan hata örüntüsünden $2^k - 1$tanesi, $2^k - 1$ tane sıfıra eşit olmayan kod cümlesiyle aynıdır. Bu $2^k - 1$ tane hata örüntüsünden biri gözlemlenirse, alınan *v* kod cümlesi bir başka *w* kod cümlesine dönüşür. Bu durumda *w* alındığında sendrom sıfıra eşit olur ve kod çözücü *w*'yi aktarılan kod cümlesi olarak kabul eder. Sonuç olarak doğru olmayan bir kod çözümü gerçekleşmiş olur. Bu çıkarım sonucunda bu kod için $2^k - 1$ tane tespit edilemeyen hata örüntüsü olduğu sonucuna ulaşılır. Eğer bir hata örüntüsü sıfıra eşit olmayan bir kod cümlesiyle aynı ise, alınan *r* vektörü bir kod cümlesi olmayacaktır ve sendrom sıfıra eşit olmayacaktır. Bu durumda hata tespit edilecektir. $2^n - 2^k$tane hata örüntüsü (n,k) doğrusal kodunun kod cümleleriyle aynı değildir. Bu $2^n - 2^k$ hata örüntüsü tespit edilebilir hata örüntüleridir. Büyük n sayısı için $2^k - 1$, 2^n'den çok küçüktür. Bu nedenle sadece hata örüntülerinin çok küçük bir kısmı kod çözücüden tespit edilemeden geçecektir.

Minimum uzaklığı d_{min} olan bir blok kod, $t = \left\lfloor \frac{d_{min}-1}{2} \right\rfloor$ ya da az sayıdaki hatayı düzeltmeyi garanti eder. $\left\lfloor \frac{d_{min}-1}{2} \right\rfloor$ terimi $\frac{d_{min}-1}{2}$ 'den küçük olan en büyük tamsayıyı belirtir. $t = \left\lfloor \frac{d_{min}-1}{2} \right\rfloor$ parametresi kodun rastsal hata düzeltme kapasitesi olarak adlandırılır. Bu özelliğe sahip koddan t-hata-düzelten kod olarak bahsedilir.

2.5 EG-LDPC Kodları

Bu bölümde EG-LDPC kodların genel özellikleri ile Öklid Geometrisi'nde tanımlanan üreteç matris ve eşlenik denetim matrislerinin özelliklerine değinilecektir.

Koda adını veren Öklid Geometrisi (EG) sonlu bir geometridir ve şu kurallar tarafından tanımlanır [18]:

- Her doğru ρ noktadan oluşur,
- Herhangi iki noktada sadece ve sadece bir doğru geçer,
- Her nokta γ doğruyla kesişir,
- Herhangi iki doğru ya bir noktada kesişir ya da birbirine paraleldir.

EG- LDPC kodları, elemanları ikili düzende olan eşlik denetim matrisi H ile temsil edilebilir. H'nin satırları EG'de yer alan doğrulara, sütunlar ise noktalara karşılık gelir. H'nin satırları etki vektörü olarak adlandırılır, ρ ağırlığına sahiptir ve doğru üzerinde yer alan noktaları gösterir. Sütunlar ise kesişim vektörü olarak adlandırılır ve belirli bir noktada kesişen doğruları temsil eder, ağırlığı γ'dir. Eğer EG'nin, i'nci doğrusu j'inci noktasını içeriyor ya da j'inci noktası i'nci doğrusunun üzerinde yer alıyorsa H'nin (i, j) koordinatlarında yer alan eleman '1' değerine sahiptir, yani $h_{ij} = 1$'dir. Haynı zamanda EG'nin noktaları üzerindeki EG doğrularının etki matrisi olarak da adlandırılır. EG'de tanımlanan ve yukarıdaki özellikleri sağlayan bu matris LDPC matrislerinin tanımlarına da uyduğu için H'nin eşlik denetim matrisi olarak tanımlandığı kod aynı zamanda bir LDPC kodudur. Bu çalışmada tip-I 2-D EG-LPDC kodu FPGA ile gerçeklenecektir ve kullanılacaktır. Tip-I 2-D EG-LDPC [20]'de verildiği üzere $t \geq 2$ koşulu ile şu parametrelere sahiptir:

- bilgi bitleri, $k = 2^{2t} - 3^t$;
- kod sözcüğü uzunluğu, $n = 2^{2t} - 1$;
- minimum uzaklık, $d_{min} = 2^t + 1$;
- eşlik denetim matrisi boyutları: $n\ x\ n$;

- eşlik denetim matrisi satır ağırlığı, ρ= 2^t;
- eşlik denetim matrisi sütun ağırlığı, γ= 2^t.

Eşlik denetim matrisi H'nin satırları, EG'de tanımlı bir etki vektörü ve onun $2^{2t}-2$ adet kaydırılmış hali ile oluşturulur. Satırlar kaydırma yöntemi ile oluşturulduğundan, EG-LDPC çevrimsel bir koddur. H'nin satırları doğrusal bağımsız olması zorunlu değildir, H'nin kertesi satır sayısı ile belirtilemez ve $n-k$'ye eşittir. Bu nedenle bu kod bir (n,k) doğrusal koddur [21]. H matrisi $n\,x\,n$ olduğundan, $n-k$ yerine n adet sendrom biti üretilir; yani sadece bilgi bitleri için değil, kod sözcüğünde yer alan tüm bitler için sendrom biti üretilir. Kod çözme işlemi ise, her bit için eşlik denetim matrisinden eşlik denetim denklemlerinin elde edilmesi ve bu denklemlere uygun olacak şekilde Tek Basamaklı Çoğunluk Mantığı algoritmasının gerçeklenmesi basamaklarından oluşur.

EG-LDPC kodlarının eşlik denetim matrisleri seyrek (ρ << n, γ << n) olduğundan bu kodların karmaşıkları ve gecikmeleri düşüktür [21,22]. Bu özellik kodlayıcı ve kodu çözücü devrelerinin temel mantık kapıları ve çoğunluk mantık kapısıyla gerçeklenebilmesi imkanını ortaya çıkarır.

Kodlama temel olarak, kod sözcüğünün oluşturulması için bilgi bit vektörü ile üreteç matrisinin çarpılmasıdır, ör.: $c = i.G$. Üreteç matris G üreteç polinom vektörü ve bu vektörden kaydırma ile edilen $2^{2t}-3^t-1$ vektör ile oluşturulur. Eşlik denetim bitleri üreteç matristen elde edilen bilgi bitlerinin doğrusal çarpımları ile elde edilir.

Kompaktlık HDK'lar için önemli bir ölçüttür. Kompakt ya da kod oranı yüksek kod yapıları bellek birimlerini daha verimli kullanabilir. Kod oranı bilgi bitlerinin sayısının kod sözcüğünde yer alan bit sayısına oranı olarak tanımlanır. EG-LDPC kod oranı ulaşılabilir kod oranı üst sınırı olan Gilbert-Varshamov ve alt sınır olan Hamming sınırı arasındadır ve Hamming sınırına yakındır [21]. Aynı k ve d'ye sahip kodlar için kod oranlarını karşılaştırmak üzere'te verilebilir.

Çizelge 2.2 Kod Oranları

Hamming Sınırı		EG-LDPC		Gilbert-Varshamov Sınırı	
(n, k, d)	Kod Oranı (k/n)	(n, k, d)	Kod Oranı (k/n)	(n, k, d)	Kod Oranı (k/n)
(14,7,5)	0,5	(15,7,5)	0,47	(17,7,5)	0,41
(58,37,9)	0,64	(63,37,9)	0,59	(67,37,9)	0,55

Tip-I 2-D EG-LDPC kodları tek basamaklı çoğunluk mantık devresi ile hataları düzeltilebilen HDK'ler arasında yer alır. Tek basamaklı çoğunluğu düzelticisi alınan kod sözcüğünün maruz kaldığı $\gamma/2$'ye kadar hataları düzeltebilmektedir. Bu yöntem hızlı ve kompakt hata düzeltme metodudur.

Kodlayıcı ve kod çözücü gibi bellek destekleyici devrelerin gerçeklendiği birleşimsel devreler de depolama birimine benzer biçimde hatalara karşı hassastır. Bellek sistemleri için Hamming gibi sıkça kullanılan kodlar kodlayıcı ve kod çözücü devrelerinde görülebilecek anlık hatalara karşı herhangi bir koruma mekanizması içermemektedir [23]. Bu tür kodlara eş zamanlı eşlik denetimi veya mantık devrelerini yineleme gibi yöntemler uygulanarak kodlayıcı ve kod çözücü devrelerinin anlık hatalara karşı bağışıklığı arttırılabilir. EG-LDPC kendi doğal yapısında var olan Hata Koruma Saptayıcı (FSD) özelliği [21]'de verilmiş ve bu özelliğe dair ispatlar yapılmıştır. HKS yapısı kodlayıcı ve kod çözücü devrelerine, ek herhangi bir devreye ihtiyaç duymadan koruma sağlar. Bunu kodlayıcı ve kod çözücünün çıktılarının eşlik denetim denklemlerine uygun olup olmadığını gözlemleyerek yapar.

3. İLİŞKİLİ ÇALIŞMALAR

Bu bölümde hata nitelik ve niceliklerini belirlemek için yapılan çalışmalara ve önceki çalışmalarda hatalara karşı önlem olarak sunulan yöntemlere yer verilmiştir.

Daha hızlı, daha küçük ve daha az güç harcayan devreler elde etmek için tasarım ve üretim teknolojisinin gelişmesiyle transistor boyutları ve çekirdek voltajları küçülmüştür. İyonizasyon enerjisine maruz kalan ortamlarda bu özelliklerle üretilmiş transistorların kullanımı TOE'ler sonucu hata görülme olasılığını arttırır. Uzay ve havacılık alanlarında kullanılan elektronik cihazlarda kullanılan SRAM tabanlı CMOS FPGA'ler için radyasyon etkileriyle başa çıkabilmek için artıklı bellek, çoklu oylama gibi hata etkilerini azaltıcı tasarım yöntemleri [29] 'de verilmiştir. Bazı hata etkilerini azaltma yöntemleri tasarımda alan, harcanan güç ve karışıklık gibi kıstaslar için verimli olmayabilir. Bu kriterler tasarımın gereksinimleriyle birlikte göz önünde bulundurularak hata etkilerini azaltma ve hatayı önleme teknikleri seçilmelidir.

Uzay tabanlı uygulamalar yüksek hacimdeki veriyi işleyebilmek ve uydu-yer bağındaki darboğazı hafifletebilmek için daha fazla işlemsel güce sahiptir. Bu isteklerin yerine getirilmesinde askeri spesifikasyonlar için üretilmemiş COTS sistemlerin kullanılması performans gereksinimlerini karşılamak için hayati önem taşıyabilir [25]. Ancak bu teknoloji uzayın ağır şartlarındaki radyasyon etkilerine karşı duyarlıdır. Yüksek performanslı COTS ürünlerin uzay sistemlerinde kullanımında veri hatalarının neden olduğu kötü sonuçların önlenebilmesi için yazılımsal ve donanımsal önemler alınarak yeterli ilgi gösterilmelidir.

Hatalara karşı alınacak önlemlerin etkin ve verimli olabilmesi için hataların nitelikleri, etkilerinin, örüntülerinin ve hata özelliklerinin bağlı olduğu etkenlerin incelenmesi gereklidir.

3.1 Hata Karakterizasyonu

Bu kısımda SRAM belleklerde ve SRAM tabanlı FPGA'lerde gözlemlenen, radyasyon ortamı kaynaklı hata özellikleri ve hataların bağlı olduğu değişkenlerden bahsedilmektedir.

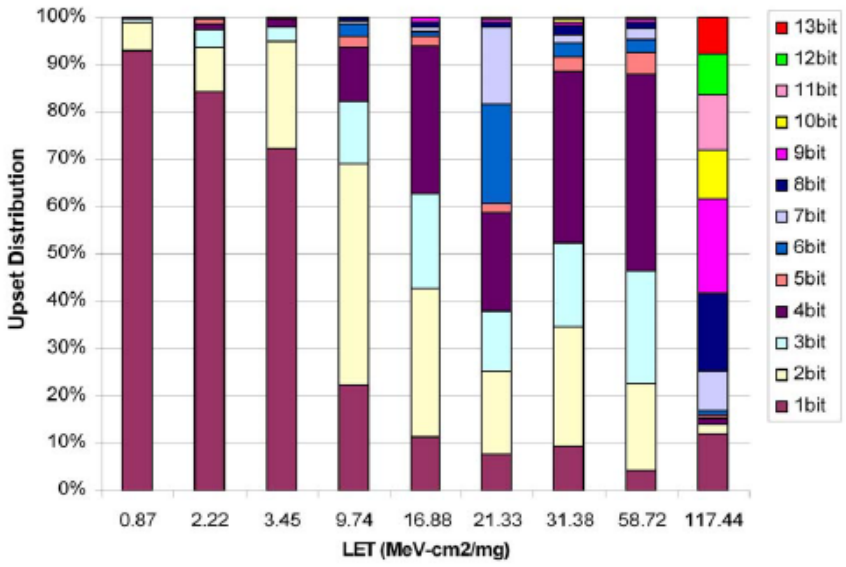
3.1.1 SRAM Belleklerde SEU ve MBU

MBU'lar iki özellikleriyle tanımlanabilir: uzamsal ve zamansal. Uzamsal MBU'lar tek bir parçacık çarpması sonucu bitlerdeki tüm hataların aynı anda oluştuğu türdür. Birden fazla parçacık çarpması nedeniyle zamanla oluşan MBU'lar ise zamansal olarak adlandırılır. Birden fazla nötron çarpması sonucu MBU oluşması olasılığı, tek bir nötron tarafından kaynaklanan MBU oluşması olasılığıyla karşılaştırıldığında, ihmal edilebilecek kadar küçüktür [12]. Silikon üretim endüstrisi 2015 yılından itibaren tüm SRAM dizilerinin uzamsal MBU'lara maruz kalacağına işaret etmektedir [26]. Bu nedenlerle tasarımcılar sistemlerde uzamsal MBU'lar nedeniyle görülebilecek işlevsel bozukluklara karşı önlem almaları gerekmektedir [12,27-30] . Birçok çalışmada uzamsal MBU'lar karakterize edilmiş ve modellenmiştir [9,11,13,16]. Sonuçlar hata bit genişliği ve bit dizilimi gibi MBU özelliklerinin işlem teknolojisi parametreleri, LET ve radyasyon ortamına göre çeşitlilik gösterdiği şeklindedir [11,27,29,32].

Elektronik ve fotonik bileşenlerin uzay radyasyonuna karşı tepkilerini gözlemlemek için 1997 yılında MPTB9 adı verilen bir deney uzaya gönderilmiştir [9]. MPTB9 Dünya'nın radyasyon kemerlerinden eş zamanlı yörüngeye kadar olan tüm ortamları kapsayan eliptik bir yörüngeye sahipti. Deneyde, uzay radyasyon etkilerinin SRAM bellek yongaları üzerindeki etkilerini gözlemlemek üzere test kartları tasarlanmıştır. Yörüngedeki seyri sırasında SRAM'ler radyasyon kemerleri dışında proton ve ağır iyonlardan oluşan kozmik ışınlara, kemerler içerisinde ise proton ve elektron akılarına maruz kalmıştır. Bu radyasyon etkileri sonucunda görülen hataların genişlikleri ve yerleşimleri kaydedilmiştir. Deney sonuçları SRAM'lerin farklı

genişlikteki MBU'lara maruz kaldığını belirtmektedir. Deney sonucunda en çarpıcı sonuç olarak tüm yongayı diagonal şeklinde kapsayan 10 hücre genişliğinde hatalar gözlemlenmiştir. Bu tür bir hatanın kaynağı olarak tam yonga yüzeyi altından geçen ve bir doğru üzerinde yer alan bellek hücreleri boyunca hareket eden iyon sebebiyle toplanan enerji olarak gösterilmiştir.

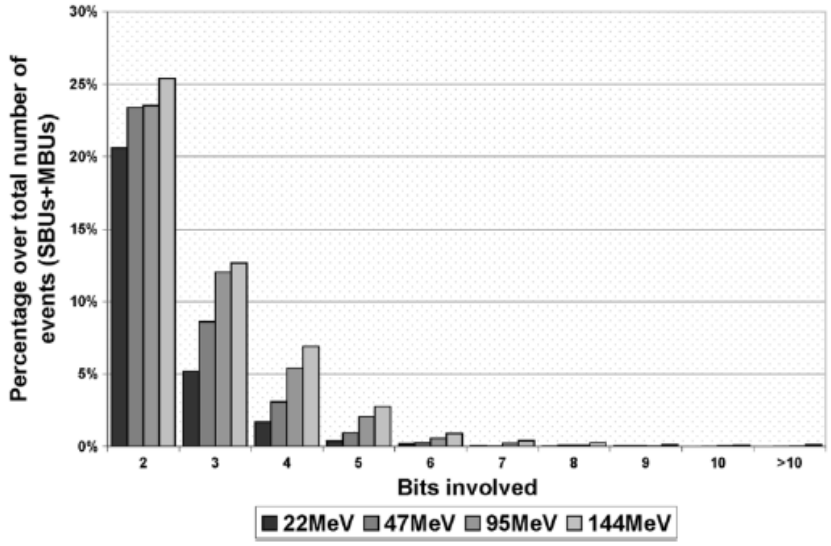
İşlem teknolojisi 90nm olan iki SRAM'in MBU duyarlılıklarını ortaya çıkarmak için ağır iyon test sonuçları [29]'te verilmiştir. Uzun etki alanına sahip tekli geri tepme mekanizmalarında, etki alanında kalan bitlerin çoğunda hata görülmesi beklenir. Elde edilen sonuçlara göre yüksek LET değerleri için 13-bit genişliğinde hatalar gözlemlenmektedir. 90nm işlem teknolojisine sahip bir SRAM için MBU bit genişlikleri ve bit genişliklerinin oranlar çeşitli LET değerlerine göre değişimi Şekil 3.1'de verilmiştir.



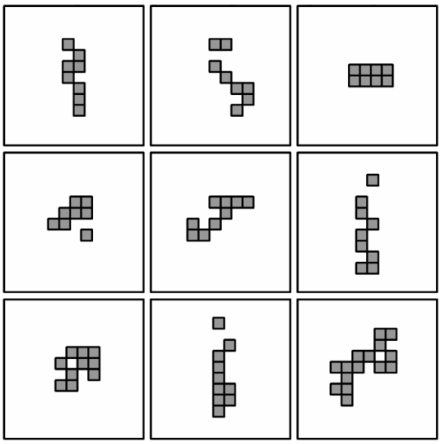
Şekil 3.1 90nm işlem teknolojisi için MBU dağılımı [29]

[27]'de 150nm işlem teknolojisine sahip bir SRAM'in hata oranı karakterizasyonu çalışmasına yer verilmiştir. Yerüstü yüksekliklerde bulunan yüksek enerjili nötronların bellek tümleşik devrelerini etkileyen en önemli radyasyon unsurları

olduğuna ve nötronlar tarafından tetiklenen MBU hassasiyetinin tasarımcılar için yüksek önemli bir probleme dönüştüğüne dikkat çekilmiştir. Yapılan çalışma sonucu farklı LET değerleri için hata bir genişliklerinin dağılımları verisi Şekil 3.2'de toplanmıştır. 2-, 3-, 4-bit hataların, oranlar göz önünde bulundurulduğunda, baskın olduğu görülmüştür ancak 10-bitten daha geniş hataların gözlenebileceği belirtilmiştir. 8-bitten fazla genişliğe sahip bazı hata örüntülerine Şekil 3.3'te yer verilmiştir.

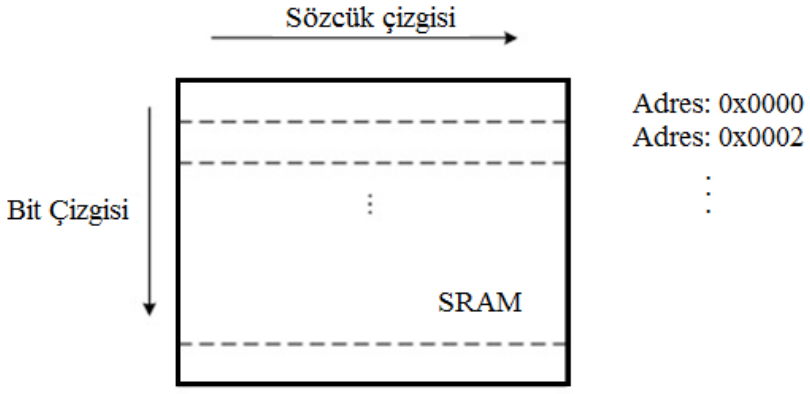


Şekil 3.2 150nm için MBU'ların LET'e göre dağılımı [27]



Şekil 3.3 Genişliği 8-bitten daha fazla olan bazı MBU'lar [27]

MBU karakteristik özellikleri değişimini etkileyen en önemli faktörlerden biri de işlem teknolojisi fiziksel boyutları ve buna bağlı olarak bellek hücreleri arasındaki aralıktır. Bir cümlede MBU görülme olasılığı, hücre uzaklıkları arttıkça, hızla düşmektedir.



Şekil 3.4 Verinin SRAM'de fiziksel yerleşimi

Hataya maruz kalan bitlerin yerleşimi veya hata örüntülerinin özellikleri belirtilirken ve fiziksel olguların etkileri incelenirken yardımcı olması için bitlerin ve sözcüklerin SRAM üzerindeki yerleşimlerini temsil eden çizgiler tanımlanabilir. SRAM'ler iki boyutlu matrislerle temsil edilebilir. Matrisin her satırının kendine has bir adresi vardır ve bu adres aracılığıyla satıra erişim sağlanır. Veri sözcüklerinin yer aldığı bu

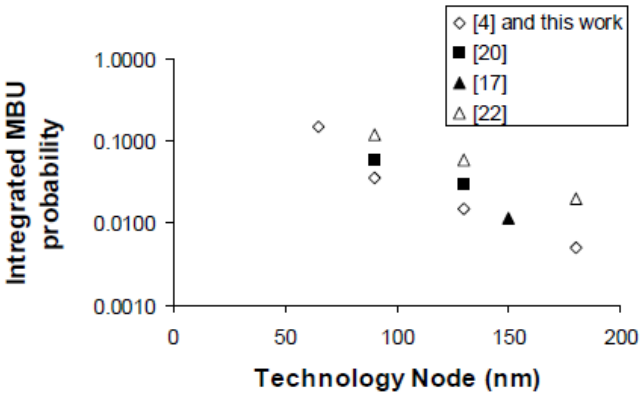
satırlar boyunca, satırlarla aynı doğrultuda çekilen çizgiye sözcük çizgisi denir. Veri sözcükleri içinde aynı sıraya sahip bitleri birleştiren ve sütun doğrultusunda olan hayali çizgilere ise bit çizgisi adı verilir.

Hücreler arasındaki aralık sözcük çizgileri doğrultusunda hücreden hücreye olan mesafe olarak tanımlanır. İşlem teknolojisi 90nm ve 65nm olan SRAM'ler [10]'da Kr ağır iyonlarına maruz bırakılmış ve bu SRAM teknolojilerinin MBU hassasiyetleri ve gözlemlenen MBU özellikleriyle ilgili bilgiler toplanmış ve karşılaştırılmıştır. MBU olasılığı, işlem teknolojisi düğüm ve transistor boyutları küçülmesiyle ve tümleşik devre yoğunluğu artmasıyla doğru orantılı olarak artar [10]. Aynı zamanda hatadan etkilenen bit sayılarında da artış gözlemlenir. Deney sonuçlarının toplandığı Grafik'e göre, 90nm SRAM'ler içim tüm hata olaylarının %27'si SEU iken ve 5-bit'ten daha büyük hatalar gözlemlenmez iken, 65nm teknolojisine geçildiğinde hiçbir SEU gözlemlenmemekte, 5-bit'ten daha büyük hatalar gözlemlenmeye başlamıştır. Aynı zamanda 3-,4- ve 5-bit hataların oranları artmıştır[11].

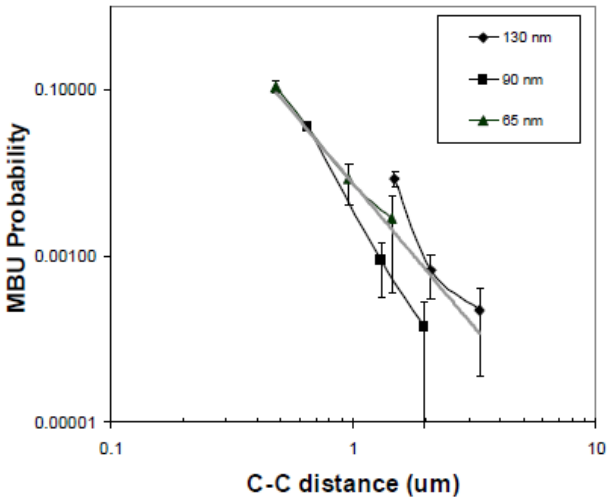
[10]'da işlem teknolojisi düğüm boyutunun MBU olasılığıyla ilişkisi daha geniş bir yelpazede incelenmiştir. MBU olasılığı küme halindeki bitlerin etkilendiği MBU olaylarının, MBU'dan etkilenen kümeler dışında kalan bitlerin maruz kaldığı SEU olaylarına oranı olarak [10]'da tanımlanmıştır. 130nm, 90nm ve 65nm teknolojilerinin MBU olasılıkları karşılaştırmalı olarak Şekil 3.5'de verildiği üzere incelenmiş ve daha önceki benzer çalışmalar özetlenmiştir. [10] 'da ve daha önceki çalışmaların sonuçları incelendiğinde teknoloji boyutları küçüldükçe MBU olasılığının üstel bir biçimde arttığı görülmüştür.

Tümleşik devreleri oluşturan transistorların fiziksel boyutları küçüldükçe tümleşik devrelerin MBU'lara karşı hassasiyeti artmaktadır [33]. Hücreler arasındaki fiziksel uzaklık azaldıkça MBU olasılığının üstel olarak arttığı [12]'de belirtilmiştir. Başka

bir deyişle, MBU olasılığı işlem teknolojisi düğüm boyutları yani transistor boyutları küçüldükçe tutarlı bir artış göstermektedir [10].



(a)

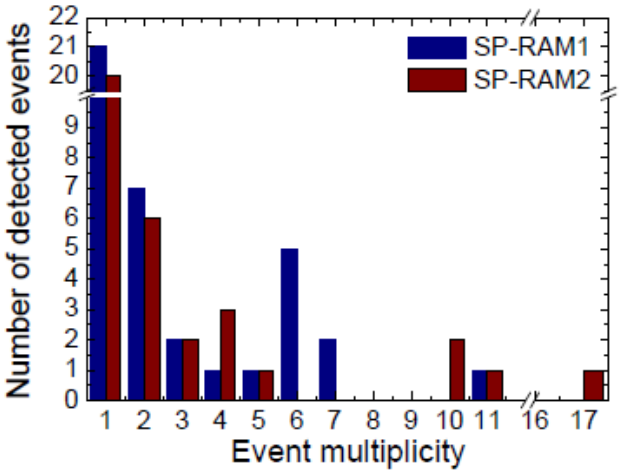


(b)

Şekil 3.5İşlem teknolojisi / Hücreler arası uzaklık - MBU oranları ilişkisi[10]

Güncel işlem teknolojilerinden olan 40nm öznitelik boyutuna sahip bir SRAM'in radyasyon etkilerine dayanımı ve tepkileri [34]'de incelenmiştir. Toplamda 7 Gigabitlik bir SRAM bellek kapasitesine sahip test kart atmosferik nötronlara yani doğal radyasyona maruz bırakılarak gözlemlenen MBU nitelikleri ve nicelikleri ile

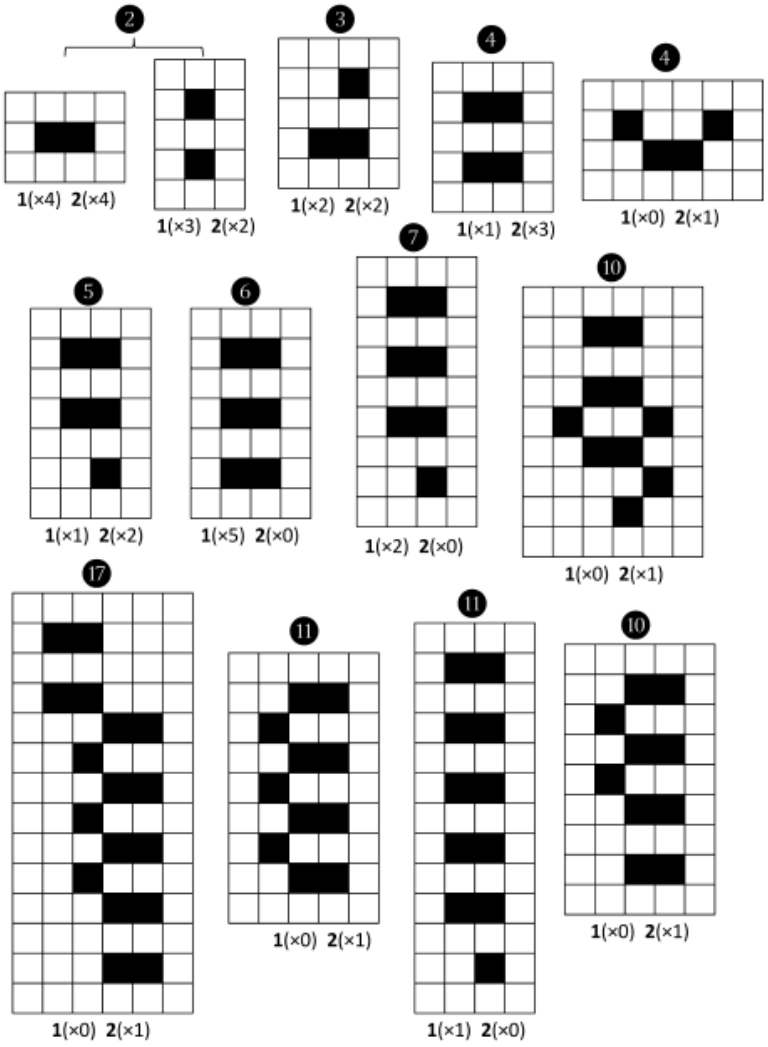
geçici hata oranları gibi deneysel veriler raporlanmıştır. Yapılan deney sonucu gözlemlenen hataların genişlikleri ve dağılımı Şekil 3.6'de verilmiştir. Deneysel veriler incelediğinde 1- ve 2-bit genişliğine sahip hataların tüm hataların büyük bir oranını oluşturduğu ancak 17-bit genişliğine kadar hataların gözlemlenebileceği görülmüştür.



Şekil 3.6 40nm işlem teknolojisi için MBU dağılımı[34]

Ayrıca yapılan geçek zamanlı deneylerde hataların bit genişlikleri yanı sıra hata örüntüleri de kaydedilmiştir ve bazı örüntüler örnek olarak Şekil 3.7'da verilmiştir.

Yapılan çalışmalar sonucu [10] radyasyon ortamına göre değişkenlik gösteren LET seviyelerinin artması ile MBU'ların SEU'lara oranının ve hata bit genişliklerinin arttığı gözlemlenmiştir. Parçacığın çarpma ve ilerleme açısına bağlı olarak, parçacıktan kaynaklanan yükü birden fazla bölgeye ulaşır ve MBU'ya neden olur [5].



Şekil 3.7 40nm üretim teknolojisine sahip SRAM hata örüntüleri[34]

Deniz seviyesinden olan yükseklik radyasyon ortamını etkilemektedir, yani değişen yükseklikle birlikte MBU karakteristiği de değişkenlik gösterir. Deniz seviyesinden uzay ortamına çıkıldığında SEU ve MBU'ların görülme oranı 100 ila 10000 kat artmaktadır [27].Çizelge 3.1'de Düşük İrtifa Yörüngesinde (DİY) ve YerdurağanYörüngede (YDY) görülen SEU ve MBU olaylarının Deniz Seviyesinde (DS) görülen olaylara oranları verilmiştir.

Çizelge 3.1 Yükseklikle SEU – MBU oranlarının değişimi

Hata Modu	*DİY-DS oranı*	*YDY-DS oranı*
SEU	7000	300
MBU	6000	2000
MBU+SEU	8000	11000

3.1.2 SRAM Tabanlı FPGA'ler

Xilinx Virtex II, 4 ve 5 serisi FPGA'lerin ağır iyonlara maruz bırakılmasıyla elde edilen durağan test sonuçları ve deneysel veriler [1]'da sunulmuştur. Farklı FPGA fiziksel kaynaklarının (CLB, BRAM) durağan bit kesit alanları ve gözlemlenen MBU özelliklerinden bahsedilmiştir. FPGA fiziksel kaynaklarının hata etkilerine karşı korunmasından TMR yönteminin etkinliği analiz edilmiştir.

[35] 'de FPGA belleklerinin (BRAM, LUTRAM, SRL'ler) için kullanılan bozukluluğa dayanıklılık sağlayan tekniklerin karşılaştırılması vardır. Bu çalışmada TMR, kopyalama ile eşlik, tek hata düzelten/çift hata tespit eden (THD/ÇHT) ve THD/ÇHT ile kopyalama teknikleri FPGA belleklerini korumak için kullanılmıştır. Bu yöntemler veri sürtmesi (scrubbing) ile birlikte kullanılmıştır. Her tekniğin etkililiği hatalara karşı hassas olan bit sayısı ve kritik hata sayısıyla ölçülmüştür.

Virtex-5 FPGA'ler için farklı LET değerleri için ağır iyon deneyleri sonucu elde edilen hata bit genişlikleri ve hata örüntüleri sonuçları [36]'da sunulmuştur. TMR MBU hataları üzerindeki etkinliği yapılan deneylerle incelenmiştir.

[8]'de Virtex-5 FPGA'lerde değişken LET değerleri için gözlemlenen hataların genişlikleri, genişliklere hata oranları ve CLB, BRAM gibi FPGA temel bileşenleri için MBU hata nitelik ve niceliklerine yer verilmiştir.

Uzayın radyasyon ortamının Virtex serisi FPGA'ler üzerindeki etkilerinin incelendiği bir diğer çalışma da [37]'de sunulmuştur. Yapılan testler sonucu radyasyon etkileri oluşan MBU'ların FPGA üzerindeki yerleşimleri, genişlikleri ve hangi bileşenlerin ne tür hatalara hassas olduğu bilgileri elde edilmiştir. Bunun yanı sıra hata niteliklerini etkileyen FPGA özellikleri irdelenmiştir.

3.1.2.1 SRAM Tabanlı FPGA'lerde SEU ve MBU

SRAM tabanlı FPGA'ler, devreleri programlanabilir rotalama bağlantı noktaları ve başvuru çizelgeleri kullanarak gerçekler. Bu iki birim de yeniden yapılandırmayı destekleyebilmek için SRAM bellek birimleri kullanır. Yeniden yapılandırma özelliği sayesinde FPGA'ler radyasyon etkilerine nedeniyle görülen hata etkilerinin azaltılması uygulama seviyesinde gerçekleştirilebildiği ve bunu FPGA işlevini sürdürürken yapıldığı için SRAM tabanlı FPGA'ler uzay uygulamaları için cazip hale gelmiştir[8]. Ancak, yapılandırma verilerinin saklandığı SRAM hücreleri radyasyon etkileri sonucu görülen SEU'lara karşı hassastır ve hatalar sonucu başvuru çizelgelerinin, bağlantı ve rotalama kaynaklarını içeriklerini değiştirebilmektedir. Bu nedenle hatalara karşı önlem metodu geliştirirken hata hassasiyetleri ve hata karakteristikleri öncelikle belirlenmelidir.

[1,8,36-38]'de SRAM tabanlı Virtex serisi FPGA'lerin SEU ve MBU'lara duyarlılığı ve gözlemlenen hataların nitelik ve niceliklerini konu alan çalışmalar sunulmuştur. 65nm işlem teknolojisiyle üretilen Virtex-5 serisi FPGA'ler için MBU tepkisi ve hata karakteristikleri [8]'de sunulmuştur. Düşük olasılıklı da olsa aynı anda birden fazla parçacığın çarpması ile birden fazla bitin etkilendiği hatalar meydana gelebilir. Daha önce de belirtildiği gibi hatanın MBU olarak adlandırabilmesi için tek bir parçacık tarafından kaynaklanıyor olması ve bitişik bitlerde oluşması gerekmektedir. Test sonuçlarında aynı anda birden fazla parçacığın neden olduğu hatalar kapsam dışı bırakılmıştır.

FPGA'lerin hata karakterizasyonu, FPGA'lerin radyasyon kaynaklarına karşı tepkileri ölçülerek elde edilir. Bu ölçüm kesit alanı adı verilen cihazın radyasyona karşı hassas bölgesinin büyüklüğünü belirler [7]. Hata kesit alanı iki şekilde ölçülebilir: tüm cihazın hatalardan nasıl etkilendiği (durağan kesit alanı) ve bir tasarımın radyasyon tepkisi (dinamik kesit alanı). Bu çalışmada sunulan yöntemler FPGA'lerde gerçeklenen uygulamalardan bağımsız ve cihaz bazında olacağından durağan kesit alanı dikkate alınacaktır.

Bir SRAM tabanlı FPGA için SEU durağan kesit alanı (σ_{cihaz}) hata sonucunda bellek içeriğini değiştirebilecek cihaz içindeki tüm düğümler için hassas alan veya hacmin ölçütüdür. Durağan kesit alanı çoğu durumda şu formülle hesaplanır:

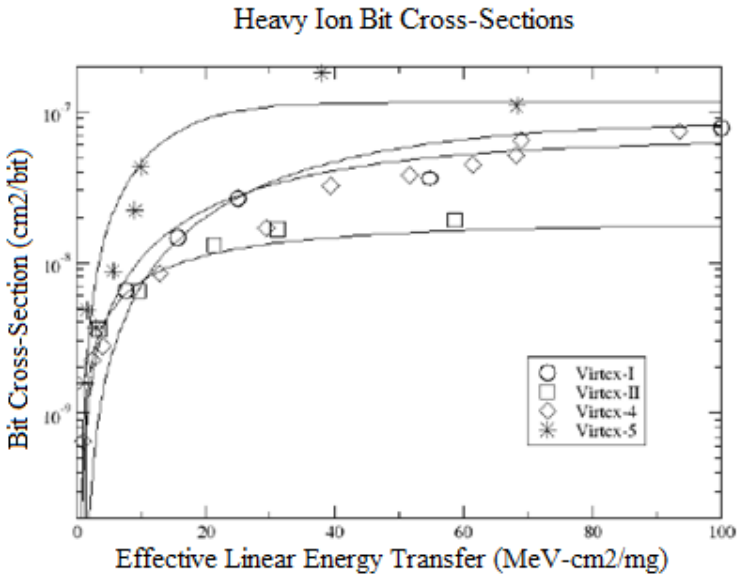
$$\sigma_{cihaz} = \frac{Hata\ sayısı}{akı\ x\ cos\theta} \tag{3.1}$$

(3.1)'de θ cihazın parçacık huzmesi etki açısıdır.

Cihazların birbiriyle karşılaştırabilmesi için bit bazında kesit alanı aşağıdaki formülle hesaplanır:

$$\sigma_{bit} = \frac{\sigma_{cihaz}}{bit\ sayısı} \tag{3.2}$$

Şekil 3.8'de Virtex-I'den Virtex-5'e tüm serilerin SEU kesit alanları verilmiştir. Virtex-I'den Virtex-5'e ilerledikçe kesit alanlarında yani radyasyona hassasiyette artış görülür.



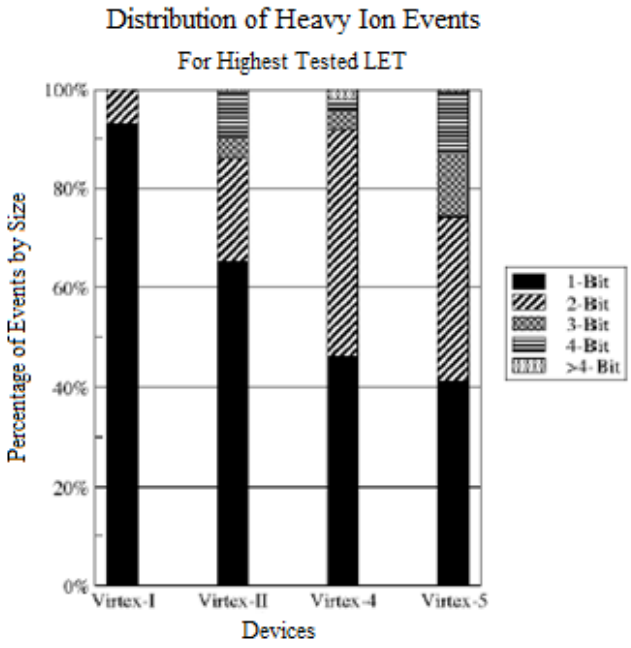
Şekil 3.8 Virtex Serileri için kesit alanları [7]

Kesit alanı nicemlendiğinde MBU'ların görülme sıklığı analiz edilebilir. MBU olgusu Virtex serileri için incelendiğinde CMOS işleme teknolojisinin gelişmesiyle birlikte daha küçük fiziksel boyutlara sahip olan daha yeni serilerin MBU'lara göre daha hassas olduğu görülmektedir.

Hata kipleri yıllardır değişmemesine rağmen özellik boyutlarının küçülmesi ve mimarilerinin değişmesi nedeniyle cihazların radyasyon etkilerine tepkilerinde değişim gözlemlenmiştir. [1,8,36]'da yapılan çalışmalarda Virtex-I, Virtex-II, Virtex-4 and Virtex-5 cihazları için ÇHT oranları ve ağır iyon olayları kesit alanları verilmiştir. Bu çalışmalardan elde edilen sonuçlara göre kesit alanı tüm cihazlar için yakın değerlerdeyken MBU oranlarında artış gözlemlenmiştir. Virtex-I ve Virtex-II serileri için SEU'lar oranları nedeniyle kritikken, Virtex-4 ve Virtex-5 serilerinde MBU'lara önem gösterilmelidir. Hata olayları bit kesit alanının tüm seriler için birbirine yakın büyüklüklerde olduğu gözlemlenmiştir. MBU oranında ise Virtex'in daha güncel seriler için bir artış gözlemlenmiştir. 25 MeV-cm2/mg veya daha yüksek doğrusal enerji transferi (LET) için sadece Virtex-I serisi cihazlarda tüm olayların %7.5 oranında MBU'lar gözlemlenmiştir. Diğer cihazlarda ise MBU'ların

sayısı LET'in artışı ile birlikte artmaktadır. Virtex-II 'de 58.7 MeV-cm2/mg LET için MBU'ların tüm olaylara oranı %21, Virtex-4'te 93.5 MeV- cm2/mg LET için oran %53, Virtex-5'te ise 68.3 MeV- cm2/mg LET için bu oran %59'dur.

Şekil 3.9'da tüm seriler için test edildikleri en yüksek LET değerleri için MBU oranları verilmiştir: Virtex-I için 100 MeV-cm2/mg, Virtex-II için 63 MeV-cm2/mg, Virtex-4 için 93.5 MeV-cm2/mg veVirtex-5 için 63.5MeV-cm2/mg. Veriler incelendiğinde tüm seriler için gözlemlenen MBU'ların %99'u 1 ila 4 bit genişliğindedir. Daha yeni cihazlarda MBU'ların olay dağılımında daha çok yer tuttuğu görülmektedir. Virtex-4 ve Virtex-5 serilerinde ise test edilen yüksek LET değerleri için MBU'lar tüm olayların %50'sinden fazladır.



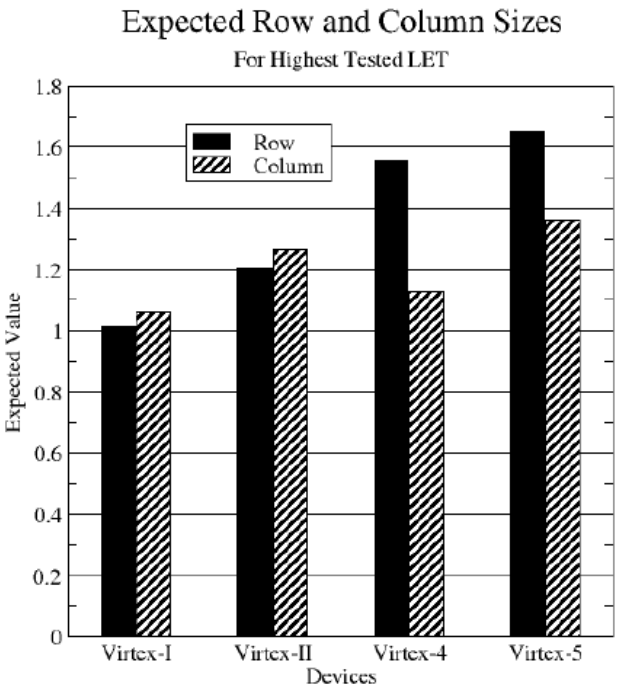
Şekil 3.9 Virtex Serileri için hata bit genişlikleri dağılımları [1]

Device	Energy (MeV)	Total Events	1-Bit Events	2-Bit Events	3-Bit Events	4-Bit Events
XCV1000	63.3	241,166	241,070 (99.96%)	96 (0.04%)	0 (0%)	0 (0%)
XC2V250	63.3	337,814	333,639 (98.76%)	4,129 (1.22%)	44 (0.01%)	2 (0.0006%)
XC2V1000	63.3	204,009	199,641 (97.86%)	2,164 (1.06%)	12 (0.006%)	1 (0.0005%)
XC4VLX25	63.3	152,577	147,902 (96.44%)	4,567 (2.99%)	78 (0.051%)	8 (0.0052%)
XC5VLX50	65.0	2,963	2,792 (94.23%)	161 (5.43%)	9 (0.30%)	1 (0.03%)
XC5VLX50	200.0	35,281	31,741 (89.86%)	3,105 (8.79%)	325 (0.92%)	110 (0.43%)

Şekil 3.10Virtex serileri için MBU dağılımı[7]

[7]'de yapılan çalışmada farklı LET değerleri için elde edilen MBU dağılımları Şekil 3.10'te verilmiştir. Birbirine yakın LET değerleri için (63.3-65.5 MeV) Virtex-5 de 3- ve 4-bit olayların görülme olasılığı Virtex-4'deki oranların yaklaşık 6 katıdır.

Virtex serisi FPGA'ler farklı LET değerlerindeki ışımalara maruz bırakılarak satır ve sütunların SEU'lara karşı hassasiyetleri ölçülmüştür [1].Test edilen en yüksek LET değeri için SEU'dan etkilenen satır ve sütunların beklenen değerleri Şekil 3.11**'de** verilmiştir. SEU'ya maruz kalan satır ve sütun sayısının beklenen değerlerinde gözlemlenen artış MBU oranlarındaki artışı yansıtmaktadır. Veriler incelendiğinde satırların MBU'lardan daha çok etkilenme eğiliminde olduğu görülür. Bunun nedeni güncel Virtex FPGA serilerinde iki satır arasındaki fiziksel mesafenin iki sütun arasındaki fiziksel mesafeden daha az olmasıdır. CLB içerisinde yer alan iki LUT veya iki dilim, iki ayrık CLB'ye nazaran MBU etkilerine maruz kalmaya daha yatkındır.



Şekil 3.11 Hatalardan etkilenen satır ve sütun sayılarının beklenen değerleri[1]

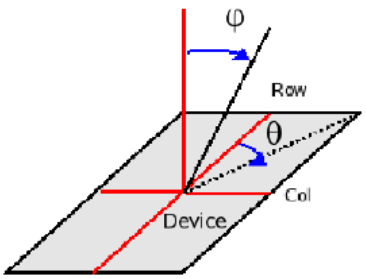
Hataların etkilerinin etki altında bıraktığı fiziksel alanın büyüklüğünden bahsedilirken derlitopluluk kavramı tanımlanır. Derlitopluluk bir MBU'nın satırlar ve sütunlar cinsinden ne kadar alan kapladığının ölçütüdür ve hatadan etkilenen bit sayısının hatadan etkilenen satır ve sütunların sayısının çarpımına oranıdır[1]. Örnek olarak; köşegen biçimdeki 2-bitlik bir hata için derlitopluluk $2/2x2 = 0.5$'tir. Birbirini izleyen seriler arasında derlitopluluk(compactness) giderek azaltmaktadır. Özellikle Virtex-4 ve Virtex-5 serileri arasındaki fark dikkat çekicidir. Derlitopluluğun azalmasıyla hata örüntülerinin çeşitliliklerinde kayda değer bir artış görülür.

TMR'ın MBU etkilerini azaltmadaki etkinliği [36]'de incelenmiştir. TMR tarafından korunan devredeki birbirinin yedeği olan iki alan bir TOE nedeniyle aynı hata değerini alırsa, oylayıcı tarafından hatalı bir çıkış verilir. Bu olgu alan geçişli olay(domain crossing event) olarak adlandırılır. MBU'ların alan geçişli olaylara kolaylıkla neden olabileceğine değinilmiştir. Bu nedenle TMR, MBU'ların çokca

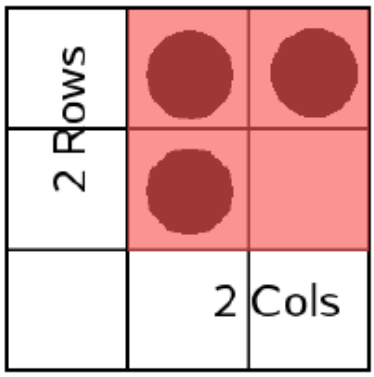
görüldüğü daha güncel Virtex-4 ve Virtex-5 serisi FPGA'ler için geçerli bir hata etkisi azaltma yöntemi olmaktan çıkmıştır. Veri sürtmesi(scrubbing) içeren yöntemlerin ise TMR'a göre daha etkin olabileceğine işaret edilmiştir.

[8]'de yapılan testler sonucu 65nm işlem teknolojisine sahip Virtex-5 FPGA'ler için SEU ve MBU'ların yanı sıra SEFI'ler de gözlemlenmiştir. Ancak SEFI'lerin kesit alanı yapılandırma verisine oranla ihmal edilebilir kadar küçük olduğu için yer verilmemiştir.

FPGA'lerde SRAM bellek yongalarından farklı olarak kaynakların dağılımının homojen biçimde olmaması nedeniyle parçacık çarpma açısının da hata karakteristiği üzerinde etkisi vardır. Bu nedenle test verileri elde edilirken farklı LET değerlerinin yanı sıra farklı ışıma açılarına da yer verilmiştir(Şekil 3.12). Işıma açıları tanımlanırken cihaz yapısının sütunlardan oluştuğu göz önünde bulundurulmuştur. φ ve θ ışıma açıları genişledikçe MBU'ların bit genişliklerinde artış olduğu gözlemlenmiştir.

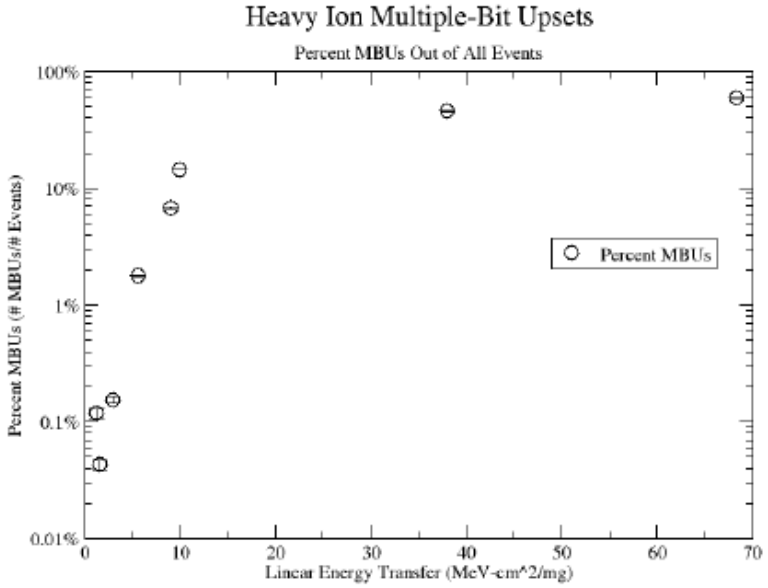


Şekil 3.12 Parçacık ışıma açıları[8]



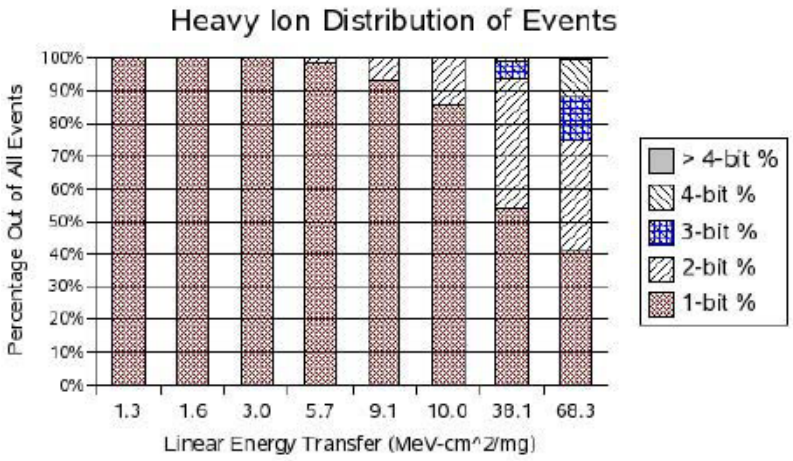
Şekil 3.13 2 x 2 Sınırlayıcı kutucuk[8]

Virtex-5 FPGA'lerde daha önceki serilerden farklı olarak hata örüntülerinde çok çeşitlilik görülmektedir. Hata örüntüleri hatanın etkilediği alanın satır ve sütun sayısı cinsinden boyutları tarafından tarif edilen ve sınırlayıcı kutucuk adı verilen yapı ile temsil edilebilir. Şekil 3.13'de 3-bit bir hata için 2x2 boyutlarında sınırlayıcı kutucuk örnek olarak verilmiştir.



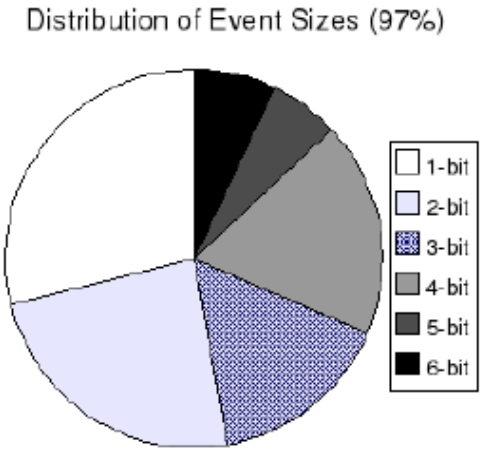
Şekil 3.14Ağır iyon testinde MBU oranlarının LET'e göre değişimi[8]

Virtex-5 FPGA'ler değişik LET değerleri için farklı tepkiler vermişlerdir. Bu tepkiler hata genişliklerine yansımıştır. Artan LET değerleri için MBU oranları artarken aynı zamanda MBU bit genişliklerinde de artış görülmüştür. 6.5 MeV LET değeri için olayların sadece %6'sı MBU iken, 200MeV için tüm olaylar MBU şeklinde görülmüştür. LET değişimine göre MBU oranları Şekil 3.14'de verilmiştir.



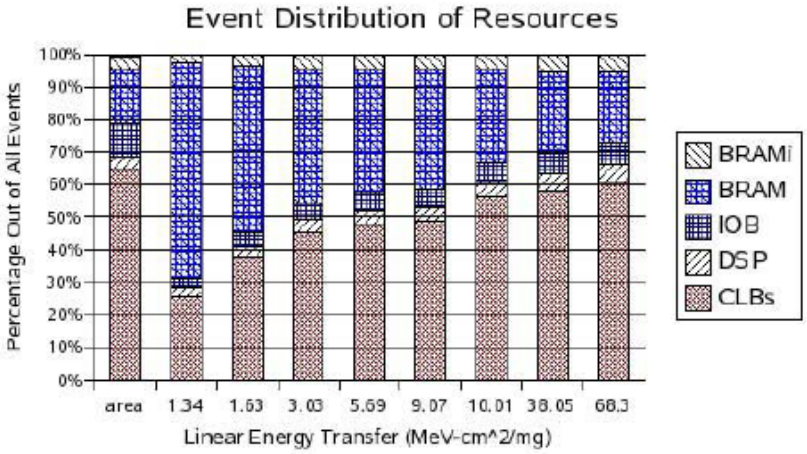
Şekil 3.15 LET'e göre hata bit genişlikleri dağılımı[8]

Şekil 3.15'da Virtex-5 için hata genişliklerinin farklı LET değerlerine göre dağılımları görülmektedir. 38.1 MeV LET değeri için 4-bite kadar genişliğe hatalar görülürken 72.8 MeV için hataların %16'sı 5- ve yüksek hata bit genişliğine sahiptir. Yüksek enerjili protonların sebep olduğu 9-bit hatalar çok nadir olsa da gözlemlenmiştir. Şekil 3.16'da 72.8 MeV LET değeri için hata olaylarının dağılımı verilmiştir.



Şekil 3.16 LET= 72.8 MeV için hata dağılımı[8]

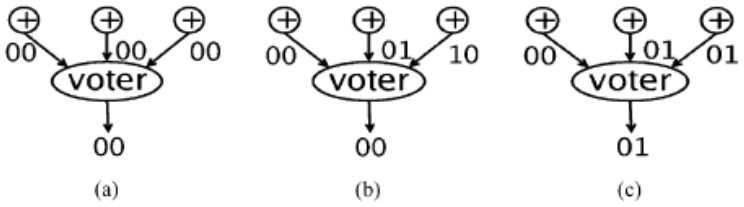
Her FPGA bileşeninin(BRAM, IOB, DSP, CLB) , Şekil 3.17'de görüldüğü üzere farklı hata karakteristiği vardır ve farklı LET değerleri için bileşen bazında hata dağılımları farklılık göstermektedir. Düşük LET değerleri için BRAM'lerde görülen hatalar baskınken, LET değeri arttıkça CLB'lerdeki hatalar ön plana çıkmaktadır.



Şekil 3.17 FPGA bileşenleri bazında hata oranları dağılımı[8]

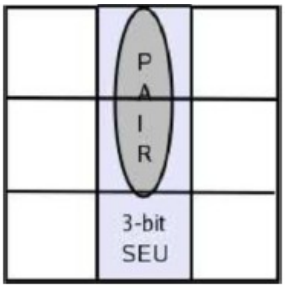
SRAM tabanlı FPGA tasarımlarında hata etkilerini azaltmak amacıyla en sık başvurulan yöntemlerden biri TMR'dır. TMR'ın Virtex serisi FPGA üzerindeki etkinliği [36] çalışmasında incelenmiştir.

TMR'ın Virtex I serisi FPGA'ler için SEU etkilerini azaltmak için etkili bir önlem olduğu [35]'de hata enjeksiyonu ve hızlandırılmış test yöntemleri ile doğrulanmıştır. Ancak, başka çalışmalarda diğer Virtex serileri içinTMR'ın SEU'lara karşı etkin olmadığı analitik olarak ispatlanmıştır. Etkileri daha baskın olan MBU'ler konu olduğunda ise hata etkilerini azaltıcı önemler karmaşıklaşır. TMR'ın etkin bir yöntem olduğu, sistemde bir anda sadece bir hata gözlemlenebileceği varsayımına dayanarak öne sürülmüştür. Ancak MBU'ler yedekli devrelerinin birden fazla kopyasında bozunmaya yol açabilir. Bu tür olaylar alan geçişli olaylar olarak adlandırılır ve Şekil 3.18'de gösterilmiştir. İki kopyada yanlış sonuçlar çıktı olarak verildiğinden sistem hatalı işleyiş olduğunu ve hatanın lokasyonunu tespit edemez.



Şekil 3.18 Alan Geçişli Olay. (a) Normal İşleyiş (b) sembollerde karşılık gelen birden fazla bitte hata gözlemlenmediği için çıkış beklenen şekildedir. (c) AGO sonucu iki kopya bozunur ve çıkış hatalıdır[36].

MBU bit genişlikleri arttıkça AGO görülme sıklığında ve sayılarında artış gözlemlenmiştir. 3-bit genişliğinde oluşan bir MBU ile MBU sonucunda oluşan 2-bitlik bir AGO'nun kesişimi Şekil 3.19'te örnek olarak verilmiştir.



Şekil 3.19 AGO MBU kesişimi[36]

Hata enjeksiyonu ile yapılan deneyler sonucunda elde edilen göre alan geçişli olayların %99'u CLB'lerde görülmektedir. %1 ise I/O blokları ve BRAM'ler görülmektedir. Başvuru çizelgesi bölgelerini etkileyen olayların %80'i birden fazla çerçevede bozunmaya neden olmuştur.

TMR uygulandığında yedekli üç alanının bir CLB'ye yerleştirilmesi sonucu üç alanın sinyalleri bir yönlendirici santralden geçirilmiş olur ve bu santral bir tek arıza noktası olur. Devrelerde bir bozunma görülmese de santralde oluşacak bir bozunma işleyiş bozukluğuna neden olur.

36.4 MeV LET değeri için yapılan testlerde Virtex-5 cihazında 4-bite kadar genişliği olan tüm olayların %1.2'si AGO'lara neden olmuştur. Bu olasılık verisi ele alındığında TMR etkin olamayacağı olay sayısı, GPS yörüngesi için radyasyon ortamı en iyi durum senaryosunda günde cihaz başına 0.6 iken en kötü durum senaryosunda günde cihaz başına 3700'dür[36].

Farklı çalışmalarda yörüngede günde görülen yada görülmesi beklenen hata sayıları yörünge deneyleriyle, yer ve test deneyleri ile modeller kullanılarak yapılan analizler sonucunda elde edilmiştir.

3.2 Hata Etkilerini Azaltıcı Önlemler

Bu kısımda SRAM bellekler ve SRAM tabanlı FPGA'lar için önceki çalışmalarda önerilen hata düzeltme yöntemlerine değinilecektir.

3.2.1 SRAM Bellekler

Kalkanlama ve paketleme teknikleri radyasyon etkilerine karşı koruma yöntemleri olarak düşünülebilir. Ancak bu yöntemler, SEU ve MBU'lar paket ve kalkanlardan kolaylıkla geçebilen nötronlar tarafından kaynaklanabileceğinden etkin değildir[2].

Hata etkilerini azaltmak için kullanılan geleneksel yöntemlerden biri de serpiştirmedir [39]. Serpiştirme, komşu bitlerin birbirinden ayrılarak farklı mantık kelimelerindeki fiziksel bölgelere yerleştirilmesidir. Serpiştirme yöntemi THD-ÇHT kodlarıyla kombine edilerek MBU etkilerini azaltıcı önlem olarak kullanılabilir. Serpiştirme uygulanmış bellek biriminde görülen MBU hataları farklı mantık kelimelerine dağıtılmış durumda olduğundan ayrık SEU'ları olarak ele alınır ve THD-ÇHT'ler uygulanır[40]. Serpiştirme içeren koruma yöntemleri komşu bitler arasında daha büyük mesafe ve karmaşık bağlantı yapıları gerektirdiği için, SRAM yongası tasarımı aşamasında güç tüketimi, yerleşim planı ve zamanlama üzerine olumsuz etkileri vardır[41]. Güncel işlem teknolojisiyle üretilen SRAM yongalarında bellek hücreleri aralarındaki mesafeler çok kısadır ve bu tarz yapılarda serpiştirmenin etkin kullanılabilmesi için serpiştirme uzaklığı arttırılmalıdır. Günümüz hata oranlarında verimli olarak koruma sağlayabilecek serpiştirmeyle elde edilecek hücre dizilimleri SRAM fiziksel yapılarını çok karmaşıklaştırdığından serpiştirme tercih edilmemektedir.

Hata tespiti ve düzeltme için sıkça kullanılan bir diğer yöntemde Üçlü Modüler Yedeklilik'tir(TMR)[26,42]. TMR metodunda bellek içeriği üç kopya olarak farklı fiziksel bölgelerde saklanır. Veri erişiminde aynı adresteki kopyalar üçlü oylamaya

tabi tutularak ideal olarak hatanın olmadığı verinin elde edilmesi hedeflenir. TMR, ancak iki veri sürtmesi döngüsü arasında bellekte birden fazla SEU görülmemesi ve birbirinin yedeği olan 3 veri kopyasından en fazla birinde bozulma olma şartıyla başarılı olmaktadır, yani THD-ÇHT bir yöntemdir. Aynı zamanda veri üç kopya olarak saklandığından fiziksel alan kullanımı ve güç tüketimi yüksektir[43].

Uzay ortamında radyasyon parçacıklarının yol açtığı bellek sistemlerindeki kalıcı ya da geçici hatalar kalın koruma kaplamalarıyla ya da HDK'ler ile sağlanır[50]. Geleneksel olarak THD-ÇHT kodlarına bellek birimlerini hatalardan korumak için sistemlerde yer verilir. Daha önce bahsedildiği gibi hataların yapısı teknolojinin gelişimine paralel olarak değişmektedir. Önceleri SEU'lar dikkat edilmesi gereken ana unsurdu ve THD-ÇHT kodlar yeterliydi. Ancak, işlem teknolojisinin küçülmesiyle MBU'lar ön plana çıkmıştır. Hamming ve TMR gibi THD-ÇHT yöntemler gerekli güvenilirliği sağlamak için yeterli değildir.

Uydular ve uzay araçlarını yönetmek için tasarlanan bilgisayarlarda kullanılan bellek birimlerini SEU ve SEFI'lere karşı korumak için kullanılan bir THD/ÇHT kodu [44]'da verilmiştir. 64-bit veri genişliği olan ve 9 yonga üzerine kurulmuş bellek organizasyonu sahip bir sistem kurulmuştur. THD/ÇHT özelliğinin yanı sıra Yerdurağan yörüngede görülen SEFI'ler nedeniyle bir yonga kaybını veri kaybı olmadan tolere edebilir. Belleğin çektiği akım bir akım denetleyici tarafından denetlenir, beklenenin dışında bir değerle karşılaşıldığında bellek sürtülerek(scrub) iki döngü arasındaki hatalar düzeltilir.

Komşu bitlerde Tek Olay MBU'su şeklinde görülen durumlarda hata etkilerini azaltmak için THD-ÇHT kodlar yetersizdir[45]. Ortogonal kodlar, evrişimsel kodlar, Bose-Chaudhuri-Hoquenghem (BCH) ve Reed-Solomon (RS) kodları bellek sistemlerini hatalara karşı korumak için sıklıkla kullanılan Hata Düzeltme Kodlarıdır(HDK)[35,41,50]. Bunlar içinden BCH ve RS Çoklu Bit Hata Düzeltme Kodlarındandır(ÇBHDK). Ancak bu kodların hata tespit ve düzeltme yetenekleri güncel SRAM ve FPGA'lerde görülen hata genişlikleri ve örüntüleri için yetersizdir.

Bunun yanı sıra, sıkça kullanılan RS ve BCH kodlarının kod çözümleri birden fazla döngü gecikmesi gerektirir[46,47]. Bu gecikmeler çoğu sistemin işleyişini olumsuz etkiler.

[48]'de evrişimsel kodların avantajlarını kullanarak daha az fiziksel alan ve destek işletim biti kullanımı sağlayan hata etkilerini azaltma yöntemi önerilmiştir. Evrişimsel kodlar veri aktarımı sırasındaki veri akımlarını korumak için en sıklıkla kullanılan hata tespit ve düzeltme yöntemlerinden biridir. Evrişimsel kodlar çoklu bit hatalarını düzeltmek için kullanıldığında kod çözücü yapıları genellikle karmaşıktır. Bu özellikleri nedeniyle daha az karmaşık kod çözücülere sahip olan blok kodları evrişimsel kodlara göre daha caziptir.

Yukarda verilen kodların kodlayıcı ve kod çözücü devreleri birleşimsel mantık devreleriyle gerçeklendiğinde geçici hatalara açıktır. Kodlayıcı ve kod çözücü devrelerin güvenilirliğini arttırmak için ek önlemler alınmalıdır. Bu devrelerdeki hata etkilerini azaltmak için genellikle yedeklilik kullanılır. Yedekliliğin sistemde uygulanması için fazladan fiziksel kaynağa ihtiyaç vardır. Yedeklilik sonucunda ekstra güç tüketimi ve gecikme kaçınılmazdır.

[49]'da GF(2) alanı üzerine tanımlanmış MBU'lara karşı güçlendirilmiş yapılar TMR ile karşılaştırılmıştır. Örnek olarak BCH kodu seçilmiştir ve BCH korumalı bir sonlu olan çarpma devresi gerçeklenmiştir. Deneysel çalışma sonucunda dinamik olarak hata olmayan veri paketlerinin direkt olarak kod çözücüye uğramadan hedefe geçirilmesiyle TMR'a göre kritik patika gecikmesinde %50 iyileştirme, destek işlem bitlerinde %33 azalma gözlemlenmiştir.

[9]'te düşük irtifa yörüngesinde görev yapan uydu uçuş bilgisayarlarının SRAM program belleklerinde hata düzeltme rutini olarak kullanılmak üzere dönüşsel kodların FPGA üzerinde gerçeklenmesi anlatılmıştır. Tasarım CPU ve bellek arasında transparan olarak çalışmaktadır. TMR fiziksel kaynaklarda %200 artış gerektirirken, uygulamada kullanılan dönüşsel kod sadece %100 artışa neden

olmaktadır. TMR bellekte yüksek işletim destek bitine ve çok sayıda giriş/çıkış pinine ihtiyaç duyduğu için dezavantajlıdır. Dönüşsel kodlarda kodlama gecikmesi 12ns iken TMR'de 2ns'dir. Kod çözücü devresinde TMR için gecikme 10 ns iken dönüşsel kod için 26nsdir.

Data aktarım sistemlerinde yaygın olan LDPC kodların uzayda kullanılan bellek sistemlerinde nasıl kullanılabileceği [50]'te gösterilmiştir. Uzay radyasyon ortamında hataları modelleyen bir kanal oluşturulmuştur ve LDPC kodları, Reed Solomon (RS) ve Bose-Chaudruhi-Hoquenghem(BCH) ile spesifik bellek yapısı için kod çözücü performansları açısından karşılaştırılmıştır. Yapılan simülasyonlar sonucu LDPC kodlarının eşlenik RS ve BCH kodlarına göre bellek ömrünü uzattığı sonucuna ulaşılmıştır.

Mantık devrelerinde görülen geçici hataların artmasıyla birlikte bellek hücreleri ile birlikte kodlayıcı de kod çözücü devrelerinde de hata etkilerini azaltıcı ve hataları düzeltmeye yönelik önlemler alınmalıdır. Hatalara karşı korumalı kodlayıcı ve kod çözücü tasarımı için bir yaklaşım [21,22]'de verilmiştir. FSD'ler kullanımıyla birlikte kodlayıcı ve kod çözücü devreleri geçici hatalara karşı korumalı olan bir kod ailesi tanımlanmıştır. EG-LDPC kodların bu ailede yer aldığı ispat edilmiştir. Hatalara karşı korumalı kodlayıcı ve kod çözücü devrelerin gerekliliği vurgulanmış ve bunun sistem güvenilirliği üzerindeki etkisiyle ilgili ölçütler verilmiştir. Uygulama nano-bellekler için tasarlanmış. Yukarda verilen özelliklere sahip kodlayıcı ve kod çözücü devreler nano-belleklerin VLSI tasarımında yer almıştır.

[41]'de tek parçacık nedeniyle komşu bitlerde oluşan MBU'ların etkilerinin azaltılması için 2 boyutlu bir hata kodu önerilmiştir. Bir veri cümlesi eş parçalara bölnümüş ve bu parçalar bir matrisin satırlarına yerleştirilmiştir. Her satırda hata tespiti için MBU hata tespit kodu, her sütunda ise hataları düzeltmek için eşlik denetim kodları uygulanmıştır. Yapılan simülasyonlar sonucu bu yapının literatürde yer alan hata düzeltme kodlarına göre dahadüşük sayıda destek bitine ihtiyaç duyduğu belirtilmiştir. Patika gecikmesi ve güç tüketimi ise Hamming kodlarından

düşüktür. Ancak SRAM bellek yongalarında karşılaşılan hata genişlikleri hata düzeltme yetenekleri yetersizdir. Önerilen 2-boyutlu yapı 3-bit hataların %68.51'ini, 4-bit hataların ise ancak %35.66'sını düzeltebilmektedir.

[2] ve [52]'de Matrix Kodu adı verilen SRAM tabanlı belleklerin MBU'lere karşı korunması hedefleyen üst seviye bir yöntem sunulmuştur. Bu yöntem Hamming Kodu ve Eşlik(Parity) Kodunu birleştirerek düşük fiziksel alanı kullanımı ve düşük destek biti sayısı hedeflemektedir. Bir milyon hata enjeksiyonu deneyi yapılarak güvenilirlik ve MTTF değerleri için çıkarımlar yapılmıştır. Sonuçlar Reed-Muller(RM) ve Hamming kodlarıyla karşılaştırılmıştır. [52]'de önerilen yöntem 3-bit hataların %79.31, 4-bit hataların %57,86, 5-bit hataların %35.02, 6-bit hataların %16.73, 7-bit hataların %5.54 ve 8-bit hataların %0.97'sini düzeltebilmektedir.

[53]'de LDPC kodlarının oluşturulması için geometrik bir yaklaşım sunulmuştur. İçlerinde EG-LDPC'nin de bulunduğu LDPC'nin dört sınıfı sonlu alanlar üzerinde tanımlı EG ve PG'nin doğru ve noktalarını kullanarak oluşturulmuştur. Bu kodların dönüşsel yapıya sokularak düşük karmaşıkla ve iyiden çok iyi kadar performansla kod çözücüler elde edilebileceği gösterilmiştir. Bu kodların seri olarak geri kaydıran yazmaçlarla gerçeklenmesi mümkün olduğu gibi paralel olarak gerçeklenerek gecikmelerin en aza indirgenmesi mümkündür.

EG-LDPC kodların kod çözümü çokluk mantığı tekniğiyle seri olarak basit bir devreyle gerçeklenebilir ancak, bu kod çözme zamanını uzatmaktadır. [54]'te verilen yöntemde kod cümlesinde bir hata olup olmadığı ilk iterasyonda tespit edilerek, hata yoksa alınan kod cümlesi kod çözme rutini durdurularak direkt olarak uygulamaya iletilmektedir. Hataya sahip kod cümlesi yüzdesinin düşük olması beklendiği göz önünde bulundurularak bu yöntem ortalama kod çözme süresinde bir hayli iyileştirme sağlamaktadır. Bu çalışmada tek basamak çoğunluk mantığı yöntemiyle kod çözebilen EG-LPDC kodlarına benzer yapıdaki DS-LDPC kodları incelenmiştir. Hata sayısı 4 'e kadar olan MBU'lar için hata tespiti ve 2 bite kadar düzeltme ile ilgili çalışmalar yapılmıştır. Üç ve dört bit hataların düzeltilebilmesi için uygulamada

iyileştirme yapılması gerektiği vurgulanmıştır. İterasyona dayalı bir çözüme sahip olduğu için hata düzeltme yeteneği ve kod çözme süresi arasında tercihli bir tasarım yapılması gerekmektedir.

EG-LDPC'nin hataları çözmedeki yeteneği ve EG-LDPC uygulanan tasarım için MTTF parametresi Reed- Muller(RM) kodu, Matrix kodu ve Hamming kodu ile [55]'de karşılaştırılmıştır. EG-LDPC'nin bu kodlara göre sırasıyla MTTF oranları %419, %104 ve %118'dir. MBU düzeltebilen, kod çözme devresi hatalara karşı korumalı olan ve düşük destek biti sayısı sunan EG-LDPC'nin MTTF'i, MBU düzeltebilen RM ve Matrix kodlarına göre sırasıyla %18.9 ve %4.5 daha yüksektir.

3.2.2 SRAM Tabanlı FPGA'ler

SRAM tabanlı FPGA'lerde hata etkilerinin azaltmak için sıklıkla kullanılan TMR yöntemi irdelenmiştir[1,8,36]. Yapılan test ve deney sonuçlarına göre TOE'ler sonucu birbirini yedekleyen üç kopya devreden iki veya üç tanesinde oluşabilecek AGO oluşması durumunda TMR etkin bir yöntem değildir.

Genel olarak sürtme(scrubbing) diye adlandırılan yapılandırma verisinin geri okunması ve yeniden yapılandırma gibi basamakları içeren yöntemleri de çok sayıda tasarımda görmek mümkündür[50,51]. Sürtme temel olarak yapılandırma verisinin hatasız kopyasının, FPGA üzerinde yer alan hatalardan etkilenmesi muhtemel olan kopyanın üzerine yazılarak hataların düzeltilmesidir.

Veri sürtmesi denetleyici tipi ve sürtmeyi gerçekleştiren devrenin yerleşimine bağlı olarak farklı şekillerde gerçeklenebilir[56]. Denetleyici FPGA üzerinde veya harici bir cihaz üzerinde olabilir, sürtme devresi işlemci tabanlı veya durum makinesi tabanlı olabilir.

BRAM'lerin dinamik olarak içerikleri(sayaçlar, durum makinesi, kontrol yazmaçları vs…) için koruma sağlamaz. Sürtme tek başına yapılandırma verisi hata oranını düşürmek için yeterli değildir, ancak yapılandırma verisinde hata birikmesini önler.

Veri sürtmesi(Scrubbing) için 3 temel strateji izlenebilir[57]:

- Hata Tespit ve Düzeltme(HTD) yöntemi ile birlikte sürtme kullanımı.
- Yapılandırma verisinin hatasız kopyasınınFPGA üzerinde yer alan hatalardan etkilenmesi muhtemel olan kopyanın üzerine yazılmasıdır.
- Geri okuma yapılarak, sadece bozunma olan veri tespit edilerek düzeltilir.

Sürtme, kodlama gibi bir HTD yöntemi ile birlikte kullanıldığında etkinliği artmaktadır. FPGA içerisinde dahili olarak gerçeklenen sürtme denetleyici devresi de TOE tipinde hatalara açık olacağından harici olarak daha güvenilir bir ortamda gerçeklenmesi sistem güvenilirliğini arttırır.

4. ÖNERİLEN HATA DÜZELTME YÖNTEMİ

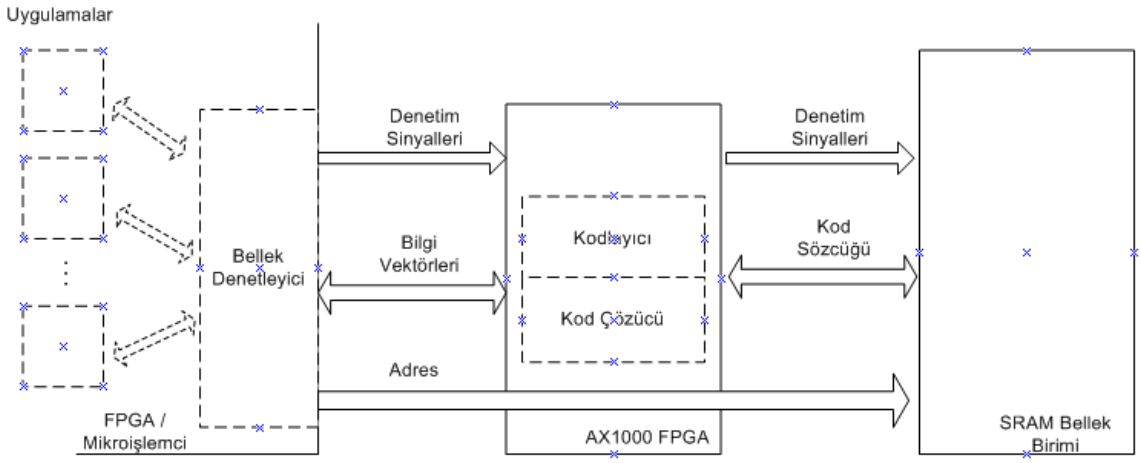
Bu bölümde hata korumalı bellek mimarisi ve hata koruması için gerçeklenen EG-LDPC tabanlı 2-boyutlu hata düzeltme kodları verilecektir.

SRAM bellek yongaları, mikroişlemciler ve FPGA'ler üzerinde koşan uygulamaların dinamik bellek ihtiyaçlarını karşılamak için sıkça kullanılır[58]. Bu çalışmada önerilen kod yapısını gerçeklenen kodlayıcı ile kod çözücü devreleri bir FPGA üzerinde yer alır. Gerçeklemelerin yapıldığı FPGA olarak Actel firmasının ürettiği AX1000 modeli seçilmiştir. Sistem güvenilirliğinin yeterli seviyeye getirebilmesi için kodlayıcının, kod çözücünün ve bellek yöneticisinin hatalara karşı gürbüz bir ortamda gerçeklenmesi gerekmektedir. Bu birimlerde yüksek işlem gücü gerekmemektedir. Bu nedenlerle anti-fuse teknolojisi ile üretilmiş olması ve anlık hatalara karşı SRAM tabanlı FPGA'lere oranla daha dirençli olan AX1000 bu görev içim uygundur [59]. Seçilen FPGA bellek destekleyici devrelerin gerektirdiğinden çok daha fazla fiziksel alana sahiptir ve bellek ihtiyacı olan uygulamalar bu FPGA üzerinde gerçeklenebilir. Uygulamaların başka bir ortamda koşturulması(ör: mikroişlemci) tercih edilirse daha küçük boyutlu bir FPGA tercih edilebilir.

SRAM bellek birimine erişim mekanizması Şekil 4.1'de gösterilmiştir ve şu adımlardan oluşur:

1- Bir uygulamadan bellek denetleyiciye yazma işlemi talebi geldiğinde, bellek denetleyici veriyi kodlayıcı birimine iletir. SRAM adresini ve kontrol sinyallerini üretir.
2- Veri bitleri seçilen yönteme göre kodlanır, kod sözcükleri oluşturulur.
3- Kodlayıcının çıktıları anlık hatalara karşı FSD tarafından doğrulanır, anlık hata tespit edilirse kodlama işlemi tekrarlanır.
4- Kod sözcükleri SRAM'e yazılır.

5- Bir uygulamadan SRAM'de saklanan veriye ulaşım talebi geldiğinde, bellek denetleyici adres ve kontrol sinyallerini oluşturarak veriyi geçici olarak FPGA'ye kopyalar.
6- Kod cümlelerine kodlama yöntemiyle eşleşen kod çözme yöntemi uygulanır.
7- Kod çözme işlemi HKS tarafından anlık hatalara karşı denetlenir, hata var ise kod çözme işlemi tekrarlanır.
8- Hatalardan arındırılmış bilgi vektörü, talebi yapan uygulamaya iletilir.



Şekil 4.1 SRAM Bellek birimine erişim

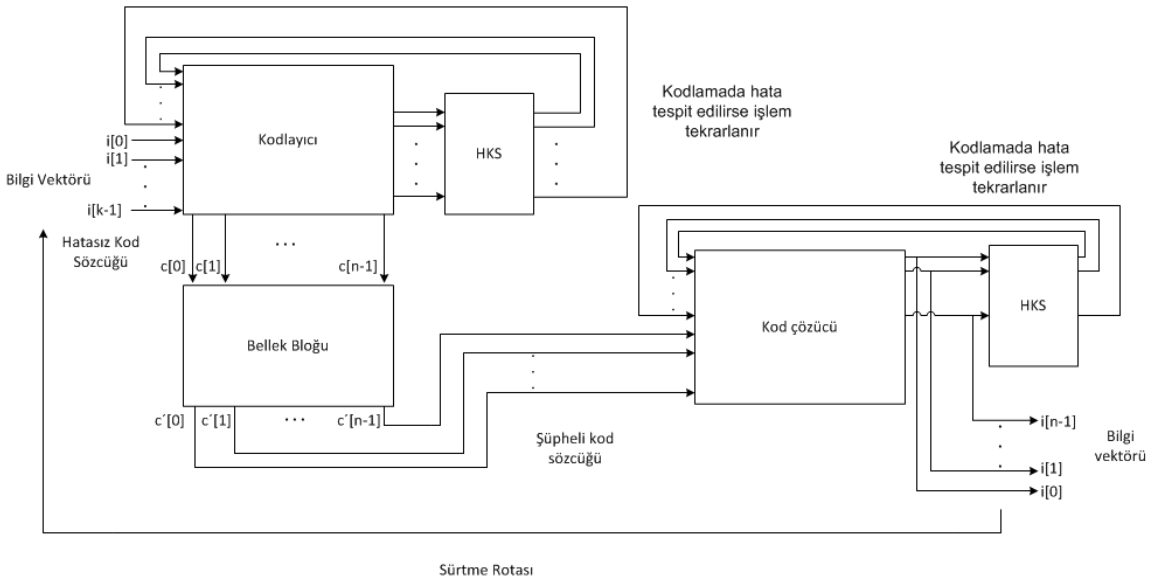
Hata düzeltme kodları belleklerin hatalara karşı dayanımlarını arttırmakta sıklıkla kullanılır. 2.4'te verilen veri kodlama teorisinin üzerinde durduğu özellikler göz önünde bulundurulduğunda EG-LDPC kodu tabanlı yaklaşımların SRAM tabanlı bellekler için etkin bir koruma yöntemi olarak kullanılabileceği görülür.

1) EG-LDPC diğer blok kodlara kod çözücü devresinin yapısı nedeniyle göre düşük gecikmeye sahiptir.
2) Hata düzeltme yeteneği yüksektir.
3) Kodlayıcı ve kod çözücü basit mantık devreleriyle gerçeklenebilir
4) Kodlayıcı ve kod çözücü devresi HKS yapısı sayesinde anlık hatalara karşı korumalıdır.

Kodlayıcı/kod çözücü ikilisi olarak bu çalışmada 4 adet yöntem sunulacaktır. İlk yöntem (63, 37, 9) EG-LDPC kodunu kullanacaktır. Diğer yöntemler 2-boyutlu çözümler sunmaktadır, EG-LDPC ve Hamming kodlarının kombinasyonları kullanılmaktadır.

4.1 (63,37,9) EG-LDPC Kodu İle Düzeltme

(63, 37, 9) EG-LDPC kodu'nun minimun uzaklığı $d = 9$'dur. Bu kod 63-bitlik kod sözcüğünde $d - 1 = 8$-bit hataları tespit edebilip, $\lfloor d/2 \rfloor = 4$-bit hataları düzeltebilmektedir. Yapı SRAM'lerin giriş/çıkışları için uygun olan 32-bit veri genişliğine uygun olacak şekilde kurulmuştur (Şekil 4.2).



Şekil 4.2 (63, 37, 9) EG-LDPC Kodlama / Kod çözme yapısı

4.1.1 Kodlayıcı

k-bit bilgi vektörünü kodlayarak n-bit kod sözcüğünü elde etmek için, bilgi vektörü kxn boyutlarında bir üreteç matris olan G ile çarpılır.

$$c = i.G \tag{4.1}$$

Üreteç polinom vektörü ve bu vektörden kaydırma ile edilen $n-1$ adet vektör G üreteç matrisinin satırlarını oluşturmaktadır. $EG(2,2^3)$ için bir $g(x)$ üreteç polinomu şöyledir:

$$g(x) = X^0 + X^2 + X^6 + X^{10} + X^{12} + X^{13} + X^{14} + X^{15} + X^{16} + X^{24} + X^{26} \tag{4.2}$$

EG-LDPC sistematik bir kod olmadığından, üreteç polinomundan elde edilen üreteç matris sistematik bir matris değildir; giriş verileri blok olarak kod sözcüğünde yer almaz. Başka bir deyişle kod çözme işleminden sonra bilgi bitlerine ek işlem gerektirmeden direk olarak erişmek mümkün değildir. Sistematik yapıda ise bilgi bitleri yerleşimlerini korurken, bunlarına sonuna eşlik denetim bitleri eklenerek kod sözcüğü oluşturulur.

Sistematik yapıdaki kod sözcüklerinde bilgi bitlerine direkt erişim sağlanabilmesi nedeniyle üreteç matrisini sistematik yapıya getirmek faydalı olur. Doğrusal sütun işlemlerinin temel alındığı bir algoritma kullanılarak $G = [I \mid X]$ şeklinde ifade edilebilen sistematik yapıda üreteç matris elde edilebilir. I, $k\ x\ k$ boyutlarında bir birim matristir. X ise $k\ x\ (n-k)$ boyutlarında bir matristir. Sistematik üreteç matris elde edilirken kullanılan kaynak kod Ek. A kısmında verilmiştir.

Sistematik üreteç matris elde edilirken kullanılan algoritma şu şekildedir:

for i=1: satır sayısı

for j=1:sütun sayısı

"1" elemanının bulunduğu ilk sütunu bul

Bu sütunu sabitle

end

for i=1: satır sayısı

ilk "1" elemanının bulunduğu satırda "1" elemanı bulunduran sütunları belirle,

Bu sütunlardan sabitlenen sütunu çıkar(Eleman – elemana XOR'la)

end

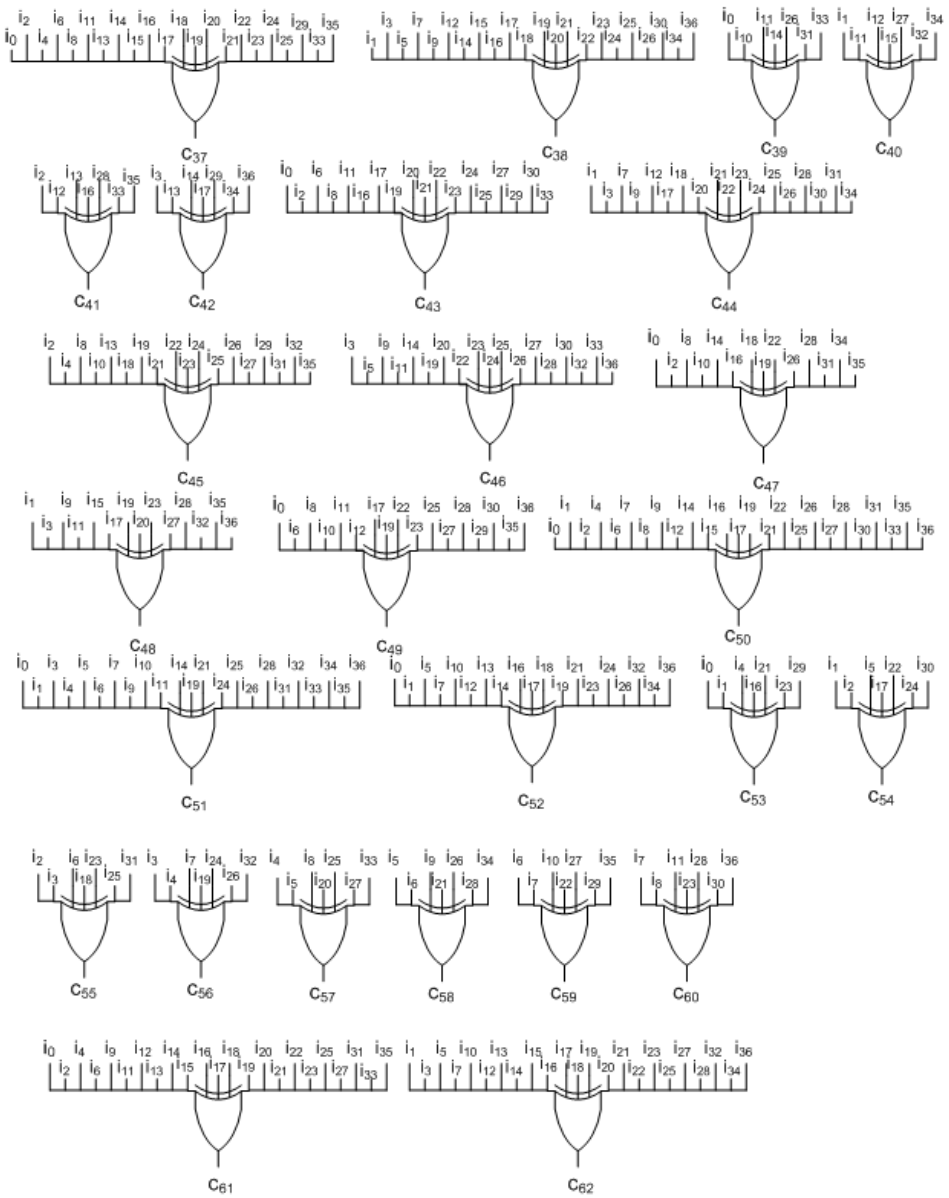
end

(63, 37, 9) EG- LDPC kodu için elde edilen sistematik üreteç matrisi Şekil 4.3'de verilmiştir.

Şekil 4.3 (63, 37, 9) EG-LDPC Üreteç Matrisi

Üreteç matris incelediğinde görüldüğü üzere $(i_0, i_1, \ldots, i_{36})$ bitleri kod sözcüğünün $c_0, c_1, \ldots, c_{36}$ bitlerine eşittir ve kodlayıcının çıkışına doğrudan kopyalanabilir. Kod sözcüğünü geri kalan bitleri ise bilgi bitlerinin doğrusal toplamlarıdır, diğer bir deyişle bilgi bitlerinin XOR'lanması ile elde edilir. Yukarda verilen üreteç matrise uygun şekilde elde edilen kodlayıcı devresi

Şekil 4.4'de verilmiştir.



Şekil 4.4 (63, 37, 9) EG-LDPC Kodlayıcı Devresi

4.1.2 Hata Korumalı Saptayıcı(FSD)

FSD, alınan kodlanmış c vektörü ve H eşlik denetim matrisini girdi olarak alıp, vektör-matris çarpımı uygulayarak sendrom vektörünü hesaplar.

$$\mathbf{s} = \mathbf{c} \,.\, \mathbf{H}^{\mathbf{T}} \tag{4.3}$$

Sendrom bitleri kod sözcüğü ile o bitin eşlik denetim matrisinde denk geldiği satırla çarpılmasıyla elde edilir. Kod sözcüğünde hataya maruz kalan bitin, sendrom vektöründeki karşılığının değeri '1'dir. Eğer sendrom vektörü tamamıyla '0'lardan oluşuyorsa bu kod sözcüğünde hata oluşmadığı anlamına gelir.

EG-LDPC kodları $\mathrm{d} - 1$ kadar hata tespit edebilir. (63, 37, 9) EG-LDPC kodu 8 adet, (15, 7, 5) EG-LDPC kodu ise 4 adet hata tespit edebilir.

Kodlayıcı ve kod çözücü devrelerinin çıkışları ideal koşullar altında yani bu devrelerde anlık hatalar gözlemlenmediği durumlarda, eşlik denetim matrislerini sağlamalıdırlar. Kodlayıcı ve kod çözücü çıkışları için sendrom vektörleri hesaplanır, eğer sıfıra eşit değilse, hesaplamaların anlık hataya maruz kaldığı ortaya çıkar ve hesaplama yeniden yapılır.

4.1.3 Kod Çözücü

Bu çalışmada kod çözücü devrelerinde uygulanmak üzere Tek Basamaklı Çoğunluk Mantığı(TBÇM) yöntemi seçilmiştir. Hataya maruz kalan bitlerin doğru değerleri doğrudan kod cümlesinden tek-basamaklı düzeltici aracılığıyla tek döngüde elde edilir. Kod çözücü için bir başka seçenek olan mesaj geçirmeli yöntemler hatayı tespit etmek ve tanımlamak için birden fazla deneme ve döngüye ihtiyaç duyarlar[23]. TBÇM kod çözücü gibi iterasyona ihtiyaç duymayan yöntemlerin

kullanımıyla gecikme ve karmaşıklık seviyeleri daha makul seviyelerde tutulabilir. [21] ve [22]'de olduğu üzere seri yöntemle gerçeklendiği biçimiyle kod çözücü devresi bir döngüde ancak bir kod sözcüğü biti düzeltebilmektedir. Bu çalışmada paralel yöntem tercih edilerek gecikme minimize edilmiştir. 63-bit genişliğine sahip bir kod sözcüğü için kod çözme işlemi seri yöntemle 63 döngü sürmekteyken, paralel yöntem için sadece bir döngü yeterlidir. Paralel yöntemde her bir kod sözcüğü biti ayrı kod çözücü devresi kurulurken, seri yöntemde bir kod çözücü devresi kurulur ve kod sözcüğü bitleri bir kaydıran yazmaç aracılığıyla kod çözücüye iletilir.

TBÇM ile ilgili teorik bilgiler [18]'da verilmiştir. Bu çalışmada eşlik denetim matrisi ile kod çözücü devresinin oluşturulması ve FPGA ile gerçeklenmesi detaylıca açıklanmıştır. Önce eşlik denetim matrisinin özelliklerinden bahsedilecek, daha sonra kod çözücü devresinin yapısı ve hata düzeltme basamaklarıyla devam edilecektir.

Eşlik denetim matrisinin satırlarını EG'de tanımlı bir etki vektörü ve bu etki vektöründen kaydırma ile edilen $2^{2t}-2$ vektör oluşturur. İki boyutlu EG$(2,2^3)$ için 2.5'de verilen özellikleri sağlayan bir etki vektörü $\{\alpha^0,\alpha^6,\alpha^{30},\alpha^{40},\alpha^{41},\alpha^{44},\alpha^{56},\alpha^{61}\}$noktalarından geçmektedir. Yani etki vektörü v_L0, 6, 30, 40, 41, 44, 56 ve 61 indisine sahip mantık değerleri '1' diğer elemanlar ise '0'dır.

$$\begin{aligned}\boldsymbol{v_L} = [&\mathbf{1\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0}\ldots\\ \ldots&\mathbf{0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0}\ldots\\ \ldots&\mathbf{1\,0\,0\,0\,0\,1\,0}]\end{aligned} \tag{4.4}$$

$\boldsymbol{v_L}$ ve $\boldsymbol{v_L}$'nin kaydırılmasıyla elde edilen vektörler kullanılarak $\mathbf{63\,x\,63}$ boyutunda eşlik denetim matrisi $\boldsymbol{H}$ oluşturulur. $\boldsymbol{n}$'inci elemanı '1' olan satırların elemanları kod

sözcüğünün $\boldsymbol{n}$'inci biti için geçerli olan eşlik denetim denklemlerinin katsayılarını oluşturur. Her bit için $\boldsymbol{\gamma}$ adet eşlik denetim toplamını hesaplamak için eşlik denetim denklemleri kullanılır. Temel olarak yapılan işlem alınan kod sözcüğü vektörü ile düzeltilecek bit için geçerli olan eşlik denetim denklemi katsayılarının bulunduğu $\boldsymbol{H}$ satırının iç çarpımıdır. $\boldsymbol{c_{62}}$biti için eşlik denetim denklemleri şu şekildedir:

$$
\begin{aligned}
&v_{1:}c_1 + c_7 + c_{31} + c_{41} + c_{42} + c_{45} + c_{57} + c_{62} = 0\\
&v_{6:}c_4 + c_6 + c_{12} + c_{36} + c_{46} + c_{47} + c_{50} + c_{62} = 0\\
&v_{18:}c_{11} + c_{16} + c_{18} + c_{24} + c_{48} + c_{58} + c_{59} + c_{62} = 0\\
&v_{21:}c_2 + c_{14} + c_{19} + c_{21} + c_{27} + c_{51} + c_{61} + c_{62} = 0\\
&v_{22:}c_0 + c_3 + c_{15} + c_{20} + c_{22} + c_{28} + c_{52} + c_{62} = 0\\
&v_{32:}c_9 + c_{10} + c_{13} + c_{25} + c_{30} + c_{32} + c_{38} + c_{62} = 0\\
&v_{56:}c_{23} + c_{33} + c_{34} + c_{37} + c_{49} + c_{54} + c_{56} + c_{62} = 0\\
&v_{62:}c_5 + c_{29} + c_{39} + c_{40} + c_{43} + c_{55} + c_{60} + c_{62} = 0
\end{aligned}
\tag{4.5}
$$

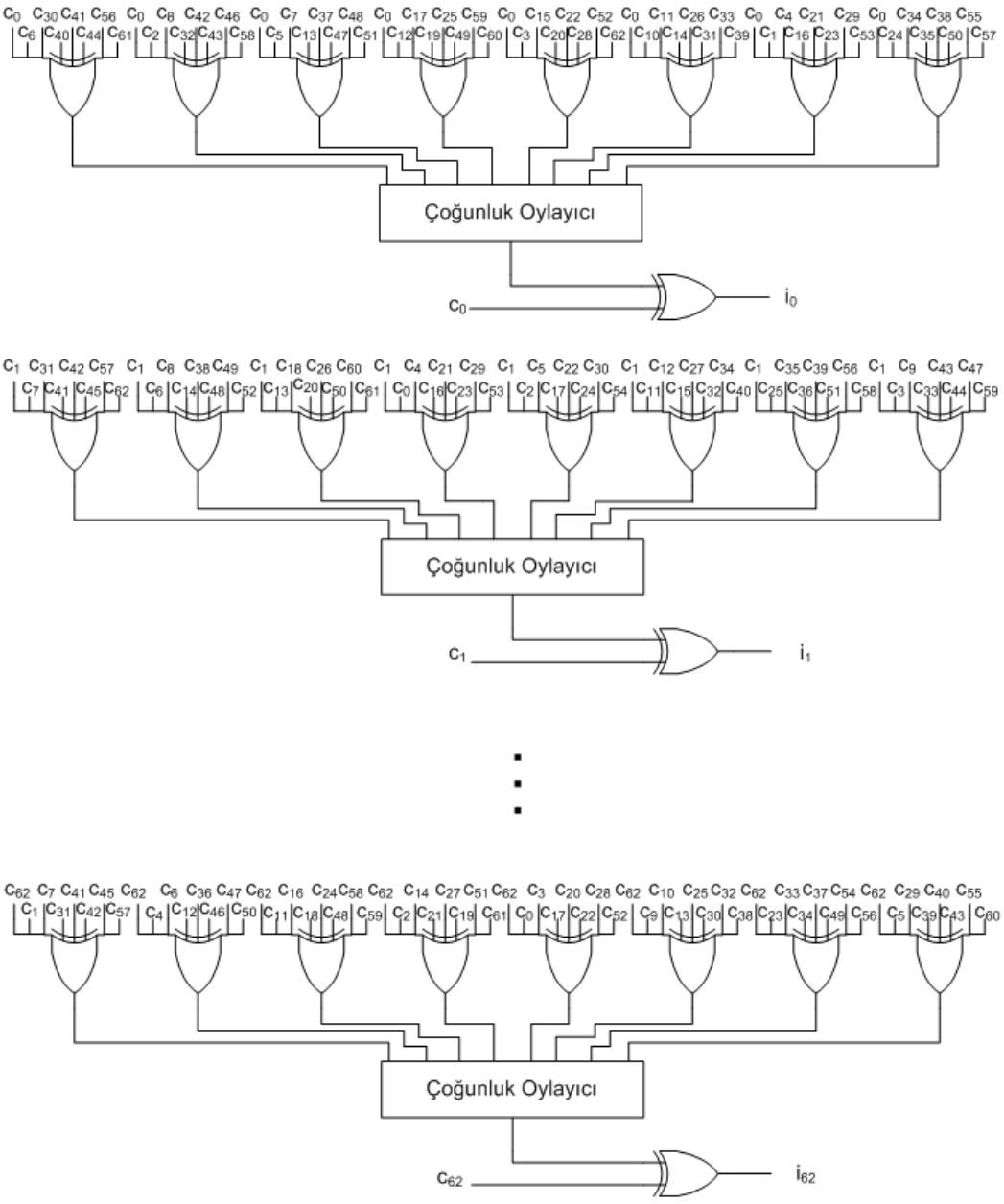
Eşlik denetim denklemleri elde edildiğinde, bu denklemlerin birden fazla bitin düzeltme işlemlerinde kullanıldığı görülmektedir. Bir kod sözcüğü tüm bitleri kapsayan denklemlerin birer kez hesaplanıp, bu denklemlerin ortak olduğu bitler için kullanılması yeterlidir. Örneğin yukarda verilen v_1 denklemi $c_1, c_7, c_{31}, c_{41}, c_{42}, c_{45}, c_{57}, c_{62}$ bitlerinin düzeltmesinde kullanılacaktır ve sadece bir kez gerçeklenmesi yeterlidir. Toplamda tüm kod cümlesi için $n\,x\,\gamma$ adet eşlik denetim matrisi tanımlanmaktadır, ancak bunların sadece n tanesi tekrarsızdır[60]. Tekrarlanan denklemlerin budanmasıyla fiziksel kaynak kullanımında iyileştirme sağlanır.

Eşlik denetim matrislerini hesaplamaları için gerekli doğrusal toplama işlemleri XOR kapıları ile gerçeklenir. Bu γ adet toplam çoğunluk kapısının girişleri olur. Çoğunluk kapısının çıkışı girişlerinin ortancasıdır. Çoğunluk kapısının, d adet girişinin $\lceil (d-1)/2 \rceil$ veya daha fazlası '1' ise çıkış '1'dir, aksi takdirde '0'dır. Çoğunluk

kapısı düz, iki adımlı mantık(ör., çarpımların toplamı) ile AND ve OR kapıları kullanılarak gerçeklenir. Alınan kod sözcüğü biti ile çoğunluk kapsının değeri XOR kapısından geçirilerek doğru bit değeri elde edilir.

Yukarda anlatılan işleyiş, her bir kod sözcüğü için ayrı ayrı üretilen kod çözücü devreler tarafından paralel olarak yürütülür. Seri olarak gerçeklendiğinde tüm kod sözcüğü bitleri bir kaydıran yazmaç aracılığıyla sıra ile kod çözücü devreye iletilir ve teker teker işlenir. n-bitin düzeltilmesi için seri gerçeklemede n döngü gerekirken, paralel gerçeklemede tek döngüde düzeltilir. Öbür taraftan fiziksel alan kullanımında tam tersi bir ilişki mevcuttur. Paralel gerçekleme için, seri gerçeklemenin yaklaşık n-katı kadar fiziksel kaynağa ihtiyaç vardır.

(63, 37, 9) EG-LDPC kodu için kod çözücü devresi Şekil 4.5'de ve VHDL kaynak kodu Ek.B kısmında verilmiştir.



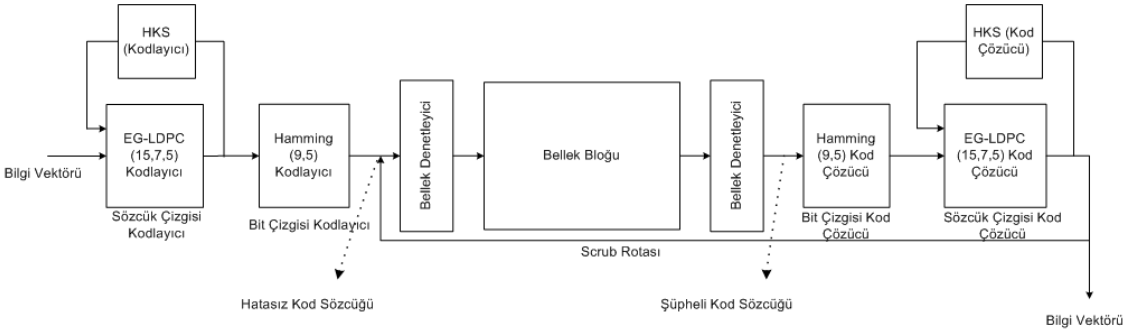
Şekil 4.5 (63, 37, 9) EG-LDPC Kod çözücü devresi

4.2 2- Boyutlu Mimariler

Gerçeklemesini yaptığımız (63, 37, 9) EG-LDPC kodu sözcük doğrultusunda 4-bit hata düzeltme yeteneğine sahiptir. Yapılan çalışmalar görülen hataların büyük oranda 1-,2-,3- ve 4-bit genişliklere sahip olduğunu ancak 17-bit genişliğine kadar hatalar gözlemlenebildiğini göstermektedir[38]. Önceki kısımlarda açıklandığı üzere işlem

teknolojisi geliştikçe SRAM'lerin MBU karakteristiklerinin değiştiği görülmüştür.(ör.hata örüntü çeşitliğinin ve hata genişliklerinin artması). Bu değişimlerle birlikte şartları daha sert olan bir radyasyon ortamı hedeflendiğinde (63, 37, 9) EG-LDPC kodunun yetenekleri yetersiz kalabilir. Bu durumda 2-boyutlar mimariler ortaya konarak hata tespit ve düzeltme yeteneklerinde iyileştirmeler elde edilebilir.

4.2.1 2- boyutlu EG-LDPC (15, 7, 5) ve Hamming (9, 5) Kodları



Şekil 4.6 (15, 7, 5) EG-LDPC & (9,5) Hamming

Standart SRAM port genişlikleri, kodların (n, k) değerleri ve destek bitlerinin bilgi bitlerine oranı göz önünde bulundurularak bilgi bitlerinin $5\,x\,7$ boyutlarında bir matrise yerleştirilmesi uygun görülmüştür. Böylece satırlar boyunca (15, 7, 5) EG-LDPC kodunun, sütunlar boyunca ise (9,5) Hamming kodunun kullanımı için uygun bir yapı oluşturulmuştur. Uygulamalarda sıkça kullanılan veri genişliği 32-bitlik vektörünün sonuna 3 adet '0' eklenerek matrise yerleştirilen 35-bit vektör elde edilir. Kodlanmış veri sözcüğü matrisinin satırları 15-bit genişliğe sahiptir ve SRAM'lerde sıklıkla kullanılan 16-bit giriş/çıkış genişliğine uygundur. Kodlama, kod çözme ve bellek birimine erişim mekanizması Şekil 4.6'da gösterilmiştir.

4.2.1.1 Kodlayıcı

Kodlayıcının giriş çıkışı arasındaki ilişkiyi ifade edebilmek ve eşlik denetim denklemlerini çıkarabilmek için öncelikle üreteç polinom vektöründen üreteç matrisinin elde edilmesi gerekir.

(15, 7, 5) EG-LDPC kodu için üreteç polinomu [18]'da şu şekilde verilmiştir:

$$\boldsymbol{g(x) = 1 + X^4 + X^6 + X^7 + X^9} \tag{4.6}$$

G üreteç matrisi, üreteç polinom vektörü v_G ve onun kaydırılması ile elde edilen 14 vektörden elde edilir.

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \tag{4.7}$$

G'den de gözlemlendiği üzere EG-LDPC kodlarının üreteç matrisleri sistematik yapıda değildir. 4.1.1'de verilen algoritma ile elde edilen sistematik yapıdaki G' üreteç matrisi şöyledir:

$$G' = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \tag{4.8}$$

Elde edilen sistematik G' incelendiğinde kod sözcüğü bitleri $c_0, c_1, \dots, c_6$'nın bitleri $i_0, i_1, \dots, i_6$'ya eşit olduğu görülür. Sözcük doğrultusundaki $c_7, c_8, \dots, c_{14}$ eşlik denetim bitleri ise şu şekilde hesaplanır:

$$\begin{aligned}
\boldsymbol{c_7} &= \boldsymbol{i_0} + \boldsymbol{i_1} + \boldsymbol{i_2} \\
\boldsymbol{c_8} &= \boldsymbol{i_2} + \boldsymbol{i_3} + \boldsymbol{i_4} + \boldsymbol{i_6} \\
\boldsymbol{c_9} &= \boldsymbol{i_2} + \boldsymbol{i_5} + \boldsymbol{i_6} \\
\boldsymbol{c_{10}} &= \boldsymbol{i_0} + \boldsymbol{i_1} + \boldsymbol{i_3} + \boldsymbol{i_4} \\
\boldsymbol{c_{11}} &= \boldsymbol{i_2} + \boldsymbol{i_3} + \boldsymbol{i_4} + \boldsymbol{i_5} + \boldsymbol{i_6} \\
\boldsymbol{c_{12}} &= \boldsymbol{i_0} + \boldsymbol{i_5} + \boldsymbol{i_6} \\
\boldsymbol{c_{13}} &= \boldsymbol{i_0} + \boldsymbol{i_4} \\
\boldsymbol{c_{14}} &= \boldsymbol{i_1} + \boldsymbol{i_5}
\end{aligned} \tag{4.9}$$

Sözcük doğrultusunda kodlama işlemleri bittikten sonra bit çizgisi doğrultusunda Hamming kodu ile kodlama işlemleri başlar. (9,5) Hamming kodu THD-ÇHT bir koddur. 5 bilgi biti için 4 adet eşlik denetim biti üretilir.

Hamming kodlarının kodlayıcı ve kod çözücü devreleri birleşimsel bloklardan oluşur. Kodlayıcı devresi, XOR kapılarını kullanarak eşlik denetim bitlerini üretir. Eşlik denetim bitlerinin $r$ile, veri bitlerinin c ile olarak sembolize edildiği kod sözcüğü yapısı şöyledir:

$\boldsymbol{r_0}$	$\boldsymbol{r_1}$	$\boldsymbol{c_0}$	$\boldsymbol{r_2}$	$\boldsymbol{c_1}$	$\boldsymbol{c_2}$	$\boldsymbol{c_3}$	$\boldsymbol{r_3}$	$\boldsymbol{c_4}$

(4.10)

Eşlik denetim bitleri aşağıdaki denklemler ile hesaplanır:

$$\begin{aligned} r_{0y} &= c_{0y} + c_{1y} + c_{3y} + c_{4y} \\ r_{1y} &= c_{0y} + c_{2y} + c_{3y} \\ r_{2y} &= c_{1y} + c_{2y} + c_{3y} \\ r_{3y} &= c_{4y} \end{aligned} \tag{4.11}$$

Hamming ile üretilen eşlik denetim bitlerinin yerleşimi bilgi bitlerinden bağımsız olduğu için 2- boyutlu kodlamada ve fiziksel yerleşimde kolaylık sağlaması açısından eşlik denetim bitleri bilgi bitlerinin sonuna eklenerek kod sözcüğü oluşturulabilir.

Bilgi bitleri, sözcükler ve bitler doğrultusundaki eşlik denetim bitlerinden oluşan ve SRAM'e yazılacak olan kodlanmış verinin yapısı aşağıda verilmiştir:

$$\begin{array}{ccccccccccccccc} c_{00} & c_{01} & c_{02} & c_{03} & c_{04} & c_{05} & c_{06} & c_{07} & c_{08} & c_{09} & c_{010} & c_{011} & c_{012} & c_{013} & c_{014} \\ c_{10} & c_{11} & c_{12} & c_{13} & c_{14} & c_{15} & c_{16} & c_{17} & c_{18} & c_{19} & c_{110} & c_{111} & c_{112} & c_{113} & c_{114} \\ c_{20} & c_{21} & c_{22} & c_{23} & c_{24} & c_{25} & c_{26} & c_{27} & c_{28} & c_{29} & c_{210} & c_{211} & c_{212} & c_{213} & c_{214} \\ c_{30} & c_{31} & c_{22} & c_{33} & c_{34} & c_{35} & c_{36} & c_{37} & c_{38} & c_{39} & c_{310} & c_{311} & c_{312} & c_{313} & c_{314} \\ c_{40} & c_{41} & c_{42} & c_{43} & c_{44} & c_{45} & c_{46} & c_{47} & c_{48} & c_{49} & c_{410} & c_{411} & c_{412} & c_{413} & c_{414} \\ r_{00} & r_{01} & r_{02} & r_{03} & r_{04} & r_{05} & r_{06} & & & & & & & & \\ r_{10} & r_{11} & r_{12} & r_{13} & r_{14} & r_{15} & r_{16} & & & & & & & & \\ r_{20} & r_{21} & r_{22} & r_{23} & r_{24} & r_{25} & r_{26} & & & & & & & & \\ r_{30} & r_{31} & r_{32} & r_{33} & r_{34} & r_{35} & r_{36} & & & & & & & & \end{array} \tag{4.12}$$

4.2.1.2 Kod çözücü

Herhangi bir uygulama tarafından SRAM'de saklanan veriye erişim talebi iletildiğinde, kodlanarak SRAM' e yazılan kod sözcüğü matrisinin tüm içeriği FPGA içindeki geçici belleğe kopyalanır. İlk olarak bit çizgisi doğrultusunda Hamming kod çözme işlemleri gerçekleştirilir. Daha sonra ise sözcük çizgisi doğrultusundaki EG-LDPC kod çözme işlemleri yerine getirilir. İki boyutta da kod çözme işlemleri tamamlandıktan sonra 32-bitlik veri sözcüğü oluşturularak veriyi talep eden uygulamaya iletilir.

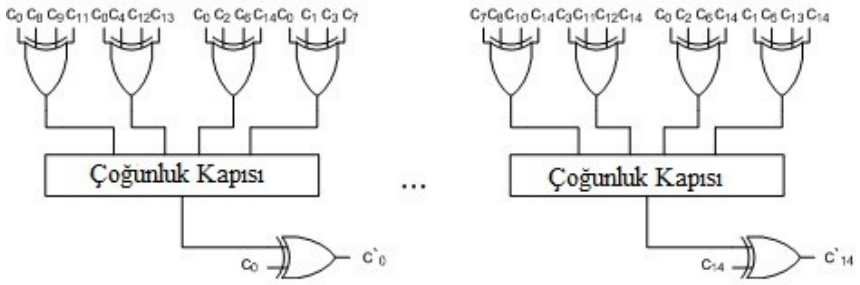
Hamming kod çözücüsü eşlik denetim bitlerini hesaplayıcı ve hata düzeltici olmak üzere iki mantık bloğundan oluşmaktadır. Kod çözücü devresi XOR ve AND kapıları kullanılarak gerçeklenebilir. Hatalı bitin pozisyonunu tespit edebilmek için öncelikle eşlik denetim denklemleri yeniden hesaplanır. Buradan elde edilen eşlik denetim bitlerinden 4-bitlik bir vektör oluşturulur. Bu vektörün sayısal karşılığı bize hatalı bitin pozisyonunu verir. Örneğin eşlik denetim bitlerinin oluşturduğu vektör "0111" ise 7. bitte hata görülmüştür. Tespit edilen pozisyondaki bit evrilerek hata etkisi ortadan kaldırılır.

Bit çizgisi doğrultusundaki Hamming kod çözme işlemleri tamamlandıktan sonra sözcük çizgisi doğrultusunda EG-LDPC kod çözme işlemleriyle devam edilir. Her satır için eş zamanlı, paralel çalışan kod çözücü devresi gerçeklenmiştir ve TBÇM algoritması tercih edilmiştir.

(15, 7, 5) EG-LDPC kodu eşlik denetim matrisinin satırlarını EG'de tanımlı bir doğrunun etki vektörü ve bu vektörden kaydırma ile edilmiş $2^{2t}-2$ adet vektör oluşturur. 2.5'de verilen 2-boyutlu EG$(2,2^2)$ koşullarını sağlayan bir etki vektörü$\{\alpha^0,\alpha^6,\alpha^{30},\alpha^{40}\}$ noktalarından oluşturulabilir[29]. Etki vektörü $v_L=$ [0 0 0 0 0 0 0 1 1 0 1 0 0 0 1] ve v_L'nin kaydırılmasıyla elde edilen 14 vektör $15\,x\,15$ boyutlarındaki eşlik denetim matrisi H'yi oluşturur. H'nin n'inci pozisyonundaki bitinin değeri '1' olan satırları kod sözcüğünün n'inci biti için geçerli olan eşlik denetim denklemlerinin katsayılarını verir. Kod sözcüğünün her bir biti için γ adet eşlik denetim toplamı hesaplanır. Bu toplamlar temel olarak alınan vektör ile H'nin bu vektöre denk gelen satırlarının iç çarpımları ile elde edilir. Kod sözcüğünün c_0 biti için H'den elde edilen eşlik denetim denklemleri şöyledir:

$$\begin{aligned} &\boldsymbol{c_0+c_8+c_9+c_{11}=0}\\ &\boldsymbol{c_0+c_4+c_{12}+c_{13}=0}\\ &\boldsymbol{c_0+c_2+c_6+c_{14}=0}\\ &\boldsymbol{c_0+c_1+c_3+c_7=0} \end{aligned} \tag{4.13}$$

Her bit için γ adet ρ- girişli XOR kapısı ve γ girişli çoğunluk kapısından oluşan TBÇM gerçeklenir. $c_0$vec_{14} bitleri için hata düzeltici devreler Şekil 4.7'da verilmiştir.



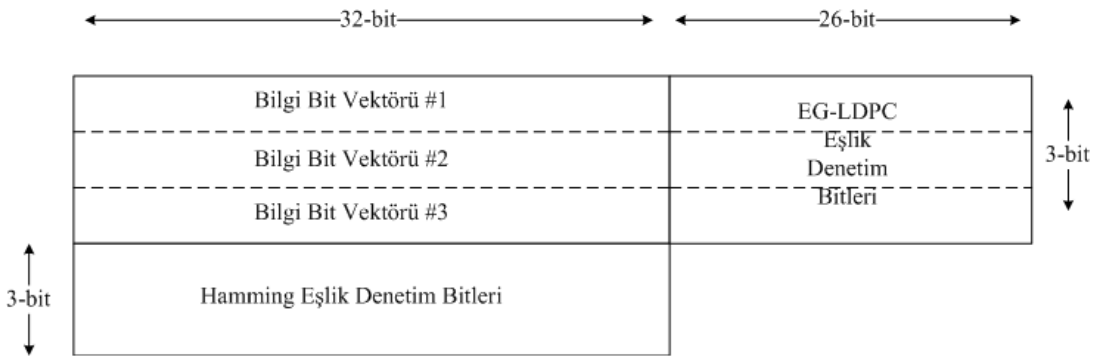
Şekil 4.7 (15,7,5) EG-LDPC düzeltme devresi

Her bir bit için eşlik denetim denklemleri çıkarıldığında, denklemlerin birden fazla bitin kod çözme işlemlerinde yer aldığı gözlemlenmiştir. Toplamda 15 denklemin yer aldığı bir küme tüm kod çözme işlemlerini kapsar. Bu 15 denklemin çıkışları bir kez hesaplanarak, yani kod çözücü devresi bir kez gerçeklenerek, sonuçların ortak olan hesaplamalarda paylaşılmasıyla fiziksel kaynak kullanımı daha aza indirgenebilir[60]. Örneğin, yukarıda verilen v_1 denklemi c_0'ın yanı sıra c_8, c_9, c_{11} bitleri için de kullanılmaktadır.

4.2.2 2- boyutlu EG-LDPC (58, 32, 9) ve Hamming (6, 3) Kodları

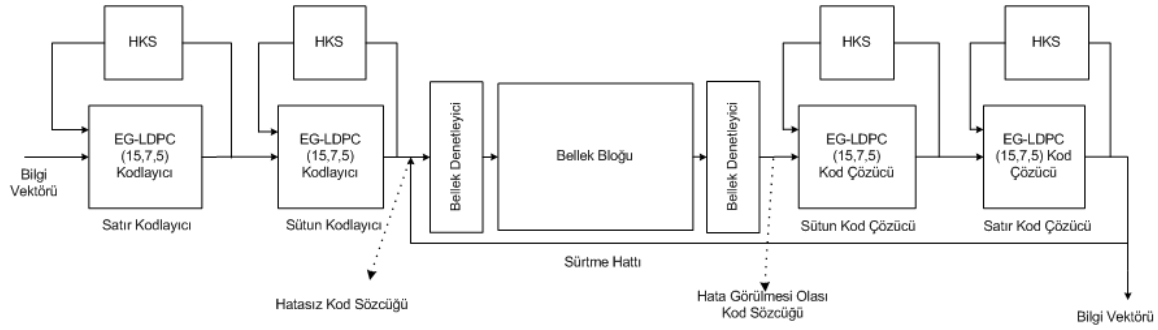
Kodlama ve kod çözme mantığı ve sıralaması 4.2.1'de verilen yapı ile benzerdir(Şekil 4.6). Üç adet 32-bitlik bilgi vektörü kullanılarak kodlanacak 3 x 32 boyutlarındaki matris oluşturulur. Satırlar yani sözcük çizgileri doğrultusu (63, 37, 9) EG-LDPC kodundan elde edilen (58, 32, 9) EG-LDPC kodu ile kodlanır. Sütunlarda ise (6,3) Hamming kodu kullanılır. Elde edilen eşlik denetim bitleri eklenerek bellek yongasına yazılacak kod sözcüğü matrisi oluşturulur(Şekil 4.8). Gecikmeyi en aza indirgemek için fiziksel kaynak kullanımından feragat edilerek satırlarda yer alan 3

bilgi bit vektörü eş zamanlı olarak kodlama kod çözme işlemleri gerçekleştirilir. Sütunlarda da aynı şekilde bu işlemler eş zamanlı olarak gerçekleştirilmektedir. Matris ve kod yapısı nedeniyle bir bilgi vektörüne kendinden önceki ve sonraki fiziksel adreslerde yer alan bilgi bit vektörleriyle birlikte erişilmesi gerekir.



Şekil 4.8 2- boyutlu EG-LDPC (58, 32, 9) & Hamming (6, 3)

4.2.3 2-Boyutlu EG-LDPC(15, 7, 5) Kodu



Şekil 4.9 2-boyutlu (15, 7, 5) EG-LDPC kodu

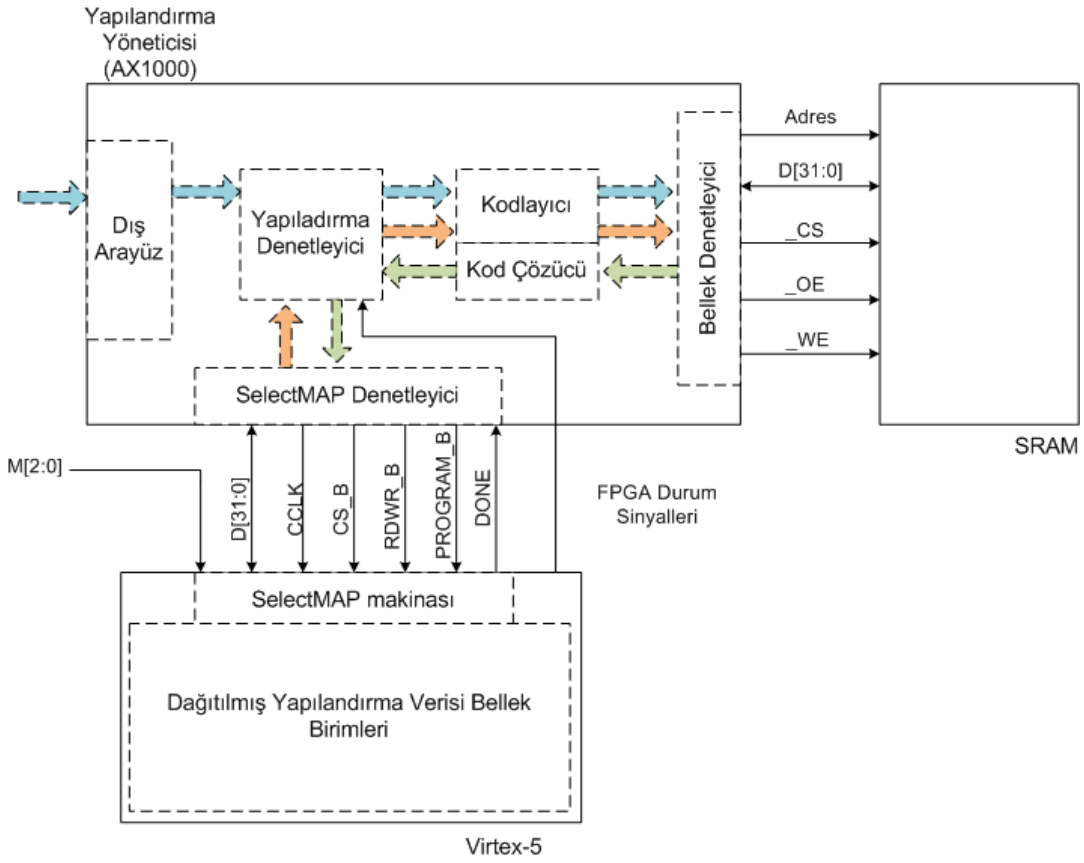
Kodlama işlemlerine başlanmadan önce veri 2-boyutlu düzene uygun hale getirilerek yeniden düzenlenir. Uygulamalarda sıklıkla kullanılan 32-bitlik veri genişliğine uygun olması için bilgi bitleri bölünerek $5\ x\ 7$ boyutlarında bir matrise yerleştirilir.

Kodlanmış veri sözcüğü matrisinin satırları 15-bit genişliğe sahiptir ve SRAM'lerde sıklıkla kullanılan 16-bit giriş/çıkış genişliğine uygundur. Kodlama ve kod çözme mantığı diğer 2-boyutlu yapılar ile benzerdir. G ve H matrisleri de 4.2.1'de sunulan diğer 2-boyutlu yapıyla aynı seçilmiştir. Satırlar boyunca kodlama için 7 adet birbiriyle eş kodlayıcı (15,7,5) EG-LDPC kodu kodlayıcı devresi gerçeklenerek ve eş zamanlı çalıştırılarak gecikme en düşük seviyede tutulmuştur. Sütun doğrultusunda kullanılan (13,5,5) EG-LDPC kodu kodlayıcı devresi (15, 7, 5) EG-LDPC kodunun son iki bilgi biti için gerçeklenen kodlayıcı ve kod çözücü devrelerinin budanmasıyla elde edilir. İki boyutta da kodlama tamamlanıp eşlik denetim bitlerinin eklenmesiyle oluşturulan matris SRAM bellek yongasına yazılır. Bu matrisin bit dizilimi aşağıda verilmiştir (Şekil 4.10). SRAM'de bulunan verilere erişim talebi olduğunda, kodlanmış matris FPGA üzerinde yer alan geçici hafızaya kopyalanır. Öncelikle sütunlar doğrultusunda kod çözme işlemi, daha sonra ise satırlar boyunca kod çözme işlemi gerçekleştirilir. Hem sütun hem satırlar için aynı kod çözücü devreler kullanılır. Bu devreler ilk döngüde sütunlarda kod çözümü, ikinci döngüde satırlar için kod çözümü aşamalarında kullanılır. Kodlanarak SRAM bellekte saklanacak kod sözcüğünün matris yapısı Şekil 4.10'da verilmiştir.

$$
\begin{array}{ccccccccccccccc}
\boldsymbol{c_{00}} & \boldsymbol{c_{01}} & \boldsymbol{c_{02}} & \boldsymbol{c_{03}} & \boldsymbol{c_{04}} & \boldsymbol{c_{05}} & \boldsymbol{c_{06}} & c_{07} & c_{08} & c_{09} & c_{010} & c_{011} & c_{012} & c_{013} & c_{014} \\
\boldsymbol{c_{10}} & \boldsymbol{c_{11}} & \boldsymbol{c_{12}} & \boldsymbol{c_{13}} & \boldsymbol{c_{14}} & \boldsymbol{c_{15}} & \boldsymbol{c_{16}} & c_{17} & c_{18} & c_{19} & c_{110} & c_{111} & c_{112} & c_{113} & c_{114} \\
\boldsymbol{c_{20}} & \boldsymbol{c_{21}} & \boldsymbol{c_{22}} & \boldsymbol{c_{23}} & \boldsymbol{c_{24}} & \boldsymbol{c_{25}} & \boldsymbol{c_{26}} & c_{27} & c_{28} & c_{29} & c_{210} & c_{211} & c_{212} & c_{213} & c_{214} \\
\boldsymbol{c_{30}} & \boldsymbol{c_{31}} & \boldsymbol{c_{32}} & \boldsymbol{c_{33}} & \boldsymbol{c_{34}} & \boldsymbol{c_{35}} & \boldsymbol{c_{36}} & c_{37} & c_{38} & c_{39} & c_{310} & c_{311} & c_{312} & c_{313} & c_{314} \\
\boldsymbol{c_{40}} & \boldsymbol{c_{41}} & \boldsymbol{c_{42}} & \boldsymbol{c_{43}} & \boldsymbol{c_{44}} & \boldsymbol{c_{45}} & \boldsymbol{c_{46}} & c_{47} & c_{48} & c_{49} & c_{410} & c_{411} & c_{412} & c_{413} & c_{414} \\
c_{70} & c_{71} & c_{72} & c_{73} & c_{74} & c_{75} & c_{76} & & & & & & & & \\
c_{80} & c_{81} & c_{82} & c_{83} & c_{84} & c_{85} & c_{86} & & & & & & & & \\
c_{90} & c_{91} & c_{92} & c_{93} & c_{94} & c_{95} & c_{96} & & & & & & & & \\
c_{100} & c_{101} & c_{102} & c_{103} & c_{104} & c_{105} & c_{106} & & & & & & & & \\
c_{110} & c_{111} & c_{112} & c_{113} & c_{114} & c_{115} & c_{116} & & & & & & & & \\
c_{120} & c_{121} & c_{122} & c_{123} & c_{124} & c_{125} & c_{126} & & & & & & & & \\
c_{130} & c_{131} & c_{132} & c_{133} & c_{134} & c_{135} & c_{136} & & & & & & & & \\
c_{140} & c_{141} & c_{142} & c_{143} & c_{144} & c_{145} & c_{146} & & & & & & & &
\end{array}
$$

Şekil 4.10 SRAM bellekte saklanan kodlanmış veri matrisi

4.3 SRAM Tabanlı FPGA'lerin Yapılandırma Verisinin korunması



Şekil 4.11 FPGA Yapılandırma Verisinin Korunması

Bu kısımda SRAM tabanlı bir FPGA'e tasarımda yer verildiğinde FPGA'in yapılandırma verisinin TOE'lere karşı korunması için bir mimari önerilecektir (Şekil 4.11).

Hem uygulamanın koşacağı FPGA hem de yapılandırma verisinin saklandığı bellek birimleri SRAM tabanlı olduğundan TOE hata etkilerine hassastırlar. Bu nedenle yapılandırma verisi kodlanarak eşlik denetim bitleri bellek biriminde saklanacaktır. Belirli aralıklarla FPGA üzerindeki yapılandırma verisi ve yapılandırma verisinin SRAM bellek birimi üzerinde saklanan eşlik denetim bitleri bir araya getirilip kod çözülerek hata bitleri düzeltilecektir. Hataların düzeltilmesiyle elde kod sözcükleri

kullanılarak hem FPGA'in yapılandırma verisi belleği hem de eşlik denetim bitlerinin saklandığı harici bellek birimine veri sürtmesi uygulanır.

Mimarinin temel bileşenleri şunlardır:

- Uygulama FPGA(Yapılandırma verisi korunan SRAM tabanlı FPGA)
- Yapılandırma yöneticisi FPGA
- Yapılandırma verisinin saklandığı bellek birimi

Uygulama FPGA: Uygulamaların üzerinde koşacağı FPGA'dir. Bu tasarımda uzay uygulamalarında hesaplama işlerinde sıklıkla kullanılmakta olan Virtex-5 serisi bir FPGA'e yer verilmiştir. 2.3'te verilen yapılandırma verisinin bozunması nedeniyle gözlemlenen hata kiplerinin önlenmesi için Yapılandırma Yönetici FPGA üzerinde gerçeklenen bir yöntem önerilmiştir.

Yapılandırma Yönetici FPGA: Hata etkisi azaltıcı önlemlerin gerçeklendiği FPGA'dir. Gerçekleme yüksek işlem gücü gerektirmediği için ve yapılandırma verileri bozunmalarına karşı gürbüz olmaları nedeniyle antifuse teknolojisine sahip bit FPGA seçilmiştir [59]. FPGA üzerinde gerçeklenen tasarım şu alt bloklardan oluşmaktadır:

- Dış arayüz: Uygulama FPGA ilk kez ayağa kaldırılırken yapılandırma verisinin hem bellek birimine hem de uygulama FPGA' yazılırken verinin alındığı arayüzdür.
- SelectMAP denetleyici: Uygulama FPGA'inin SelectMAP yapılandırma arayüzüne bağlı olan arayüzdür. Yapılandırma ve yapılandırmanın geri okunması sırasında uygulama FPGA ile yapılandırma FPGA arasında veri alışverişinin SelectMAP protokolü ile sağlanır.
- Yapılandırma denetleyici: Yapılandırma verisinin saklandığı dış arayüz, bellek birimi ile uygulama FPGA arasındaki veri akışını sağlar. Sürtme döngüsünü başlatır ve denetler. Hata etkilerini azaltma işleyişini denetleyen durum makineleri burada yer alır.

- Kodlayıcı / Kod çözücü: Yapılandırma verisi bu çalışmada önerilen 2-boyutlu (15, 7, 5) EG-LDPC kodu ile kodlanır. Sistematik yapıda olduğundan yapılandırma verisi ve eşlik denetim bitlerini hiçbir ek işleme ihtiyaç duymadan ayırmak mümkündür. Yapılandırma verisi uygulama FPGA üzerinde tutulurken, eşlik denetim bitleri SRAM bellek birimi üzerinde tutulur.
- Bellek birimi: kodlanmış yapılandırma verisinin, eşlik denetim bitlerinin saklandığı bellek birimidir.

İşleyiş şu basamaklardan oluşur.

1- Uygulama FPGA ilk kez ayağa kaldırılırken yapılandırma verisi dış arayüzden alınır.
2- Yapılandırma verisi kodlanır, bilgi bitleri ile uygulama FPGA yapılandırılarak ayağa kaldırılırken, eşlik denetim bitleri bellek birimine yazılır.
3- Belirli döngülerle, uygulama FPGA'inden yapılandırma verisi geri okuma ile elde edilir.
4- Elde edilen yapılandırma verisi ve SRAM bellek biriminde saklanan eşlik denetim bitleri ile birleştirilerek elde edilen kod sözcüğü kod çözücüden geçirilerek hatalardan arındırılır.
5- Hatalardan arındırılmış kod sözcüğünün bilgi bitleri ile FPGA yapılandırma verisine, eşlik denetim bitleri ile bellek birimine sürtme uygulanır.

Bilgi bitleri ile eşlenik denetim bitleri iki farklı ortamda saklanır. İki sürtme döngüsü arasında iki farklı fiziksel kaynakta aynı kod sözcüğü etkileyen iki veya daha fazla hatanın aynı anda görülme olasılığı göz ardı edilecek küçüktür. İki sürtme döngüsü arasında tek bir kod sözcüğünü ilgilendiren, bilgi bitlerinde yada eşlik denetim bitlerinde en fazla tek bir olay gözlenmesi beklenir. SRAM belleklerin ve SRAM tabanlı FPGA'lerin yapılandırma belleklerinin maruz kaldığı hata örüntüleri incelendiğinde 2-boyutlu (15, 7, 5) EG-LDPC kodlamasının bu hataların etkilerini azaltmak / hataları düzeltmek için yeterli olduğu görülür.

5. DENEYSEL SONUÇLAR

Kodlayıcı, FSD ve kod çözücü devreleri Actel AX1000 FPGA için VHDL dili kullanılarak gerçeklenmiş ve Synopsis Synplify aracı kullanılarak sentezlenmiştir. MATLAB'da bir referans model oluşturularak VHDL ile yapılan gerçeklemenin doğrulanması için kullanılmıştır. Deneysel verinin hata enjeksiyonu ile hatalara maruz kalması sağlanmıştır. Yapılan simülasyonlarla bu çalışmada önerilen yöntemlerin etkinlikleri ölçülmüştür.

[1]'da *hücre bitişikliği* kavramının tanımı MBU'nin incelenmesinde kullanılmak üzere yapılmıştır. Eğer bir bit, başka bir biti çevreleyen 8-bit yerleşiminden birinde yer alıyorsa, iki bit birbirine bitişiktir. Bitişik bitlerde grup olarak gözlemlenen hatalar için bu çalışmada önerilen EG-LDPC kodu tabanlı tek boyutlu ve iki boyutlu yapıların hata tespiti ve düzeltme kabiliyetleri incelenmiştir. Bu tez kapsamında gerçeklemesi yapılan kod çözücü devreleri, daha önce önerilen Matrix kodları ve diğer 2- boyutlu kodlar ile hata tespit ve düzeltme yetenekleri, fiziksel alan kullanımı ve gecikme gibi performans kriterleri açısından karşılaştırılmıştır.

Tek boyutta etkinliğini gözlemleyebilmek adına (15,7,5) EG-LDPC kodu, Hamming ve Evrişimsel kodlar ile karşılaştırılabilir. [62]'te verilen (21,16) Hamming kodu ve (32,16) Evrişimsel kod tek hata düzeltebilen kodlardır. Bu iki kod verilen kaynaklarda 16-bit genişliğine sahip bilgi bit vektörü için gerçeklenmiştir. EG-LDPC uygulamasında ise 16-bitlik vektör üç parçaya bölünmüştür ve bu parçalar ayrı ayrı (15, 7, 5) EG-LDPC kodu ile kodlanmıştır. Çizelge 2.1'de görüldüğü üzere EG-LDPC kullanılarak Hamming ve Evrişimsel kodlara göre düşük gecikme ile hata tespit ve düzeltme yeteneklerinde çok büyük gelişme elde edilmiştir.

Çizelge 5.1 Tek boyutlu kodların karşılaştırması

	(21,16) Hamming [34]	(15, 7, 5) EG-LDPC	(32, 16) Evrişimsel Kod [37]
# 4-girişli LUT sayısı	99	77	48
Gecikme	21.7 ns	3,778 ns	9.344 ns
Artıklık Bitleri	5	24	16
Yetenek	THD-ÇHT	12-bit tespit / 6-bit düzeltme	THD

Bu tez kapsamında gerçeklemesi yapılan (63, 37, 9) EG-LDPC kodu, optimize edilmiş (63, 37, 9) EG-LDPC kodu ve 2-boyutlu mimariler için kod çözücülerin FPGA üzerinde kapladıkları alan, gecikme ve hata düzeltme yeteneği gibi özelliklerinin karşılaştırması Çizelge 5.2'de verilmiştir. 2-boyutlu yapılarda daha az fiziksel kaynak kullanımı ile hata tespit ve düzeltme yeteneklerinde büyük gelişme kaydedilirken, güç tüketimi ve gecikmede artış olduğu gözlemlenmiştir.

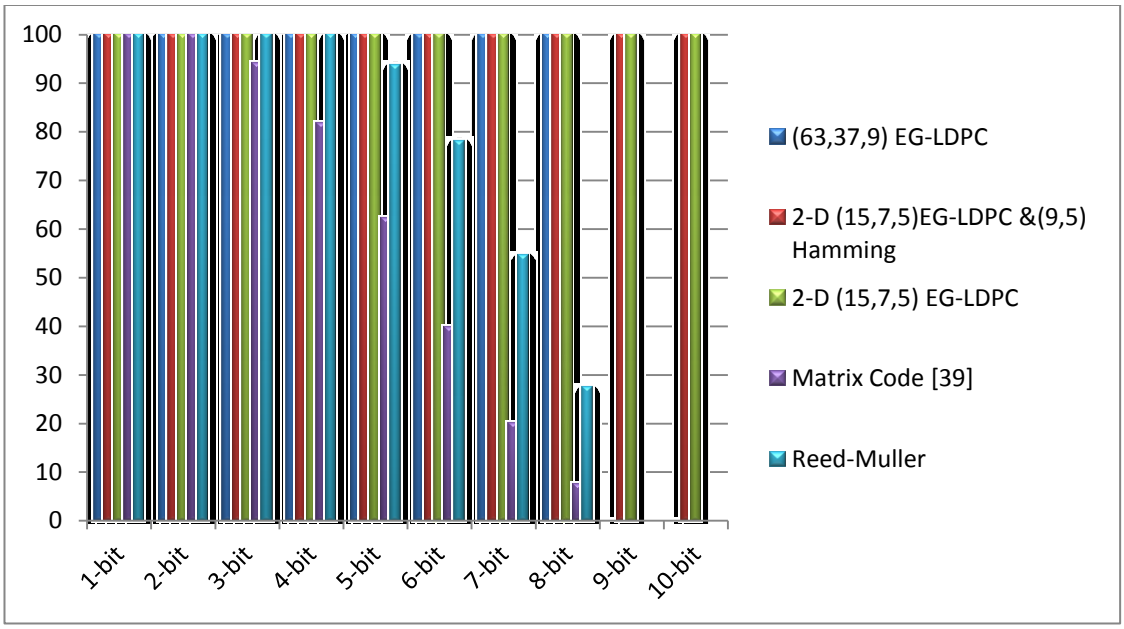
[52] ve [61]'ta hata etkilerini azaltmak için iki boyutlu Matrix kodları ve bu kodların hata tespit ve düzeltme yetenekleri ile ilgili deney sonuçlarına yer verilmiştir. Çizelge 5.3 ve Çizelge 5.4'de önceki çalışmalarda önerilen kodlar ile bu tez kapsamında gerçeklenen kod yapılarının hata tespit ve düzeltme yetenekleri karşılaştırılmıştır. [41]'ta verilen iki boyutlu yapı 3-bit hataların %68.51'ini ve 4-bit hataların %35.66'sını düzeltebilmektedir. Bu çalışmada önerilen yapıların tümü 3-bit ve 4-bit hataların %100'ünü düzeltebilmektedir. Bu çalışmada önerilen kodların hata genişlikleri arttıkça diğer kodlara oranla etkinliklerinin de arttığı gözlemlenmiştir. [52] ve [61]'da verilen matriks kodlar 6-bit hataların %16.73'ünü, 7-bit hataların %5.54'ünü düzeltirken, $k = 32$ için 2-boyutlu (15, 7, 5) EG-LDPC kodu ile 8-bite kadar olan hataların %100'ü, 9-bit hataların %80.5'i düzeltilebilirken 30-bite kadar belirli dizilimlerdeki hataları düzeltebilmektedir. EG-LDPC (58, 32, 9) & Hamming (6,3) kodu ile belirli örüntülerde en çok 44- bite kadar olan hatalar düzeltilebilir.

Çizelge 5.2 Bu çalışmada gerçeklenen 2-boyutlu kodların karşılaştırması

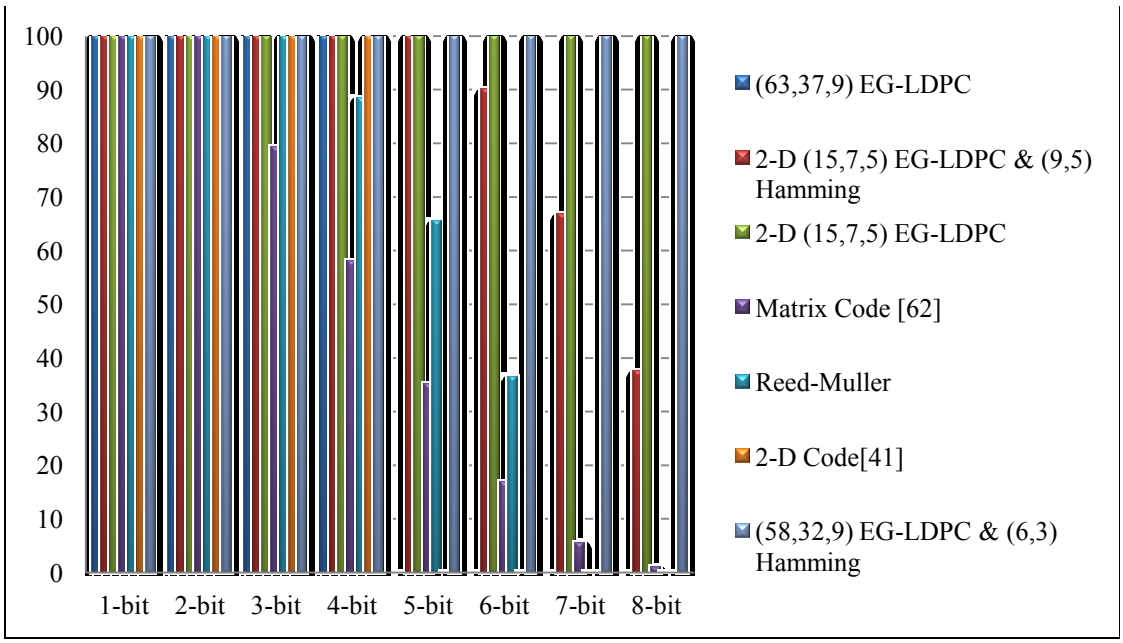
	EG-LDPC (63,37,9)	EG-LDPC (63,37,9) optimized	EG-LDPC (15,7,5) & Hamming (9,5)	EG-LDPC (15,7,5) & EG-LDPC (15, 7, 5)	EG-LDPC (58,32,9) & Hamming (6,3)
Cihaz kullanımı (LUT sayısı)	552	543	224	210	1821
Gecikme	8.336ns	7.821ns	6.043ns	9.318ns	6,047ns
Hata Düzeltme Yeteneği ($k = 32$ için)	4-bit	4-bit	%100 - 5-bite kadar %90 - 6-bite kadar %66,7 - 7-bite kadar	%100 - 8-bite kadar 30-bit (en çok)	%100 - 8-bite kadar 44-bit (en çok.)

Bu çalışmada önerilen yapılar güvenilirlik, güç tüketimi ve gecikme özellikleri için iyileştirmeler sunarken destek bit sayısı bakımından dezavantajlıdır. Bilgi bitleri(k) ve eşlik denetim bitleri($n-k$) genişlikleri ve bilgi bitlerinin kod sözcüğü uzunluğuna oranı(k/n)Error! Reference source not found. Çizelge 5.5'de verilmiştir. 2-boyutlu (15,7,5) EG-LDPC kodu yapısı en iyi hata düzeltme özelliklerine sunarken(k/n) oranı bakımından dezavantajlıdır. THD özelliğine sahip TMR'ın (k/n) oranı %33.3 iken 30-bite kadar hataları düzeltebilen 2-boyutlu (15,7,5) EG-LDPC kodunun (k/n) oranı %30.4'tür, EG-LDPC (58,32,9) ve Hamming (6,3) kombinasyonu 44-bite kadar hataları ve [34]'te verilen tüm hata örüntülerini düzeltebilirken (k/n) oranı %35.6'dır(Çizelge 5.5).

Çizelge 5.3 Kodların Hata Tespit Yetenekleri



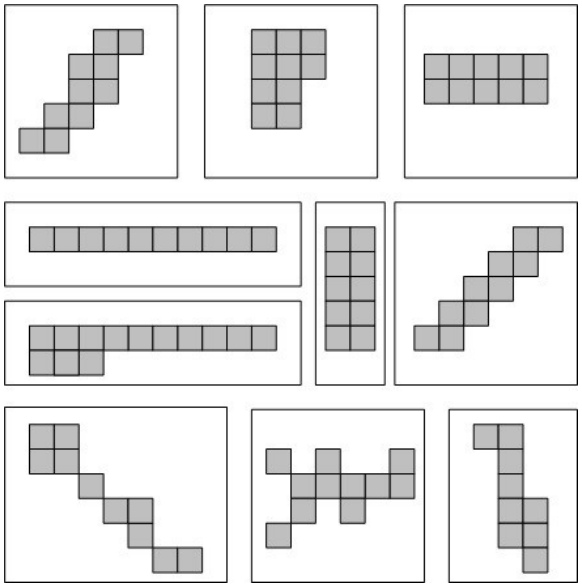
Çizelge 5.4 Kodların Hata Düzeltme Yetenekleri



Çizelge 5.5 Veri / Kod sözcüğü oranları

Kod Yapısı	Bilgi Bitleri(k)	Eşlik Denetim Bitleri ($n-k$)	k/n
EG-LDPC(63,37,9)	37	26	%58.7
EG-LDPC(15,7,5) & Hamming(9,5)	35	68	%33.9
EG-LDPC(15,7,5) & EG-LDPC(15,7,5)	49	112	%30.4
EG-LDPC(58,32,9) & Hamming(6,3)	96	174	%35.6
TMR	32	64	%33.3
Matrix Code[39]	32	40	%44.4
Reed-Muller	32	32	%50

2-boyutlu (15, 7, 5) EG-LDPC kodu ile düzeltilebilen 8-bitten daha fazla genişliğe sahip dizilimlerden bazıları Şekil 5.1'de verilmiştir. Bu yapı ile örüntüye bağlı olarak 30-bite kadar hatalar düzeltilebilmektedir. İkiden fazla satırda hatadan etkilenen bit sayısı 3 yada daha fazla olmaması koşulu ile hata örüntüleri düzeltilebilmektedir.



Şekil 5.1 Bazı 8-bitten fazla genişliğe sahip ve 2-boyutlu (15, 7, 5) EG-LDPC kodu ile düzeltilebilen hata örüntüleri

6. SONUÇ

Bu çalışmada SRAM bellek yongaları ve SRAM tabanlı FPGA'ların yapılandırma verilerinin uzay radyasyon etkileri sonucu meydana gelen hatalara karşı korunması / hata etkilerinin azaltması için hata tespit ve düzeltme yöntemleri sunulmuştur. İşlem teknolojisi özellik boyutları küçüldükçe SRAM bellek hücrelerinin hatalara karşı hassasiyeti artmaktadır. Bu artışla birlikte hata genişlikleri ve hata örüntüleri çeşitliliğinde artışlar görülür. Bu nedenle zorlu radyasyon ortamlarında, özellikle uzayda, veri ve görev kaybını engellemek için etkin yöntemlere ihtiyaç vardır. Bu noktada 3 adet 2 boyutlu hata düzeltme yöntemi sunulmuştur: (9, 5) Hamming ve (15, 7, 5) EG-LDPC kodları kombinasyonu, EG-LDPC (58, 32 9) ve Hamming (6, 3) kodları ve (15,7,5) EG-LDPC kodu. Bu kodlar daha önceki çalışmalarda sunulan 2 boyutlu kodlar, TMR yöntemi ve RM kodlarıyla hata tespit ve düzeltme yetenekleri ve eşlik denetim bitleri, kod sözcüğü bitleri oranları bakımından karşılaştırılmıştır. Simülasyon sonuçları karşılaştırıldığında bu çalışmada önerilen yöntemlerin hata düzeltme başarımlarının başka çalışmalarda önerilen yöntemlere göre daha yüksek olduğu sonucuna varılmıştır.

2.4'te verilen, veri kodlama teorisinin üzerinde yoğunlaştığı kod özelliklerinin, EG-LDPC kodu tabanlı sistemler tarafından daha önceki çalışmalarda önerilen 2-boyutlu kod yapılarına oranla daha yüksek başarımla sağlandığı sonucuna varılmıştır. Kod çözücü devresi yapısı nedeniyle EG-LDPC kodu sistemlerinin gecikmeleri düşüktür. Basit mantık devreleri tarafından gerçeklenebildiği için tasarım karmaşıklığı ve tasarımın kapladığı fiziksel alan. düşüktür. HKS yapısı sayesinde kodlayıcı ve kod çözücü devrelerinde görülebilecek anlık hataların önüne geçilerek kod çözme hataları ortadan kaldırılır.

KAYNAKLAR

[1] Quinn, H., Morgan, K., Graham, P., Krone, J., Caffrey, M., A review of Xilinx FPGA architectural reliability concerns from Virtex to Virtex-5,9th European Conference on Radiation and Its Effects on Components and Systems, 1-8, Eylül 2007.

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

[3] "International Traffic in Arms Regulations" erişim adresi: http://pmddtc.state.gov/regulations_laws/itar_official.html, erişim tarihi: 11 Aralık 2012.

[4] Duzellier, S.,Radiation effects on electronic devices in space, Aerospace Science Technology, 9(1), 93-99, 2005.

[5] Dodd, P.E., Massengill, L.W., Basic mechanisms and modeling of single-event upset in digital microelectronics, IEEE Transactions onNuclear Science, 50(3), 583- 602, Haziran 2003.

[6] Srour, J.R., McGarrity, J.M., Radiation effects on microelectronics in space, Proceedings of the IEEE , 76(11), 1443-1469, Kasım 1988.

[7] "Xilinx Virtex FPGA Design Guide for Space" erişim adresi: www.cosmiacpubs.org/pubs/design_guide_with_cover.pdf, erişim tarihi: 9 Kasım 2012.

[8] Quinn, H., Morgan, K., Graham, P., Krone, J., Caffrey, M., Static Proton and Heavy Ion Testing of the Xilinx Virtex-5 Device, IEEE Radiation Effects Data Workshop,177-184, Temmuz 2007.

[9] Buchner, S., Campbell, A.B., Meehan, T., Clark, K.A., McMorrow, D., Dyer, C., Sanderson, C., Comber, C., Kuboyama, S., Investigation of single-ion multiple-bit upsets in memories on board a space experiment, Fifth European Conference onRadiation and Its Effects on Components and Systems, 558-564, 1999.

[10] Seifert, N., Slankard, P., Kirsch, M., Narasimham, B., Zia, V., Brookreson, C., Vo, A., Mitra, S., Gill, B., Maiz, J., Radiation-Induced Soft Error Rates of Advanced CMOS Bulk Devices, 44th Reliability Physics Symposium Proceedings, 217-225, Mart 2006.

[11] Giot, D., Roche, P., Gasiot, G., Autran, J.-L., Harboe-Sorensen, R., , Heavy Ion Testing and 3-D Simulations of Multiple Cell Upset in 65 nm Standard SRAMs, IEEE Transactions onNuclear Science, 55(4), 2048-2054, Ağustos 2008.

[12] Maiz, J., Hareland, S., Zhang, K., Armstrong, P., Characterization of multi-bit soft error events in advanced SRAMs, IEEE InternationalElectron Devices Meeting, 21(4), 1-4, Aralık 2003.

[13] Zoutendyk, J.A., Smith, L.S., Edmonds, L.D., Response of a DRAM to single-ion tracks of different heavy-ion species and stopping powers, IEEE Transactions onNuclear Science, 37(6), 1844-1848, Aralık 1990.

[14] Zoutendyk, J.A., Edmonds, L.D., Smith, L.S., Characterization of multiple-bit errors from single-ion tracks in integrated circuits, IEEE Transactions on Nuclear Science, 36(6), 2267-2274, Aralık 1989.

[15] Musseau, O., Gardic, F., Roche, P., Corbiere, T., Reed, R.A., Buchner, S., McDonald, P., Melinger, J., Tran, L., Campbell, A.B., Analysis of multiple bit upsets (MBU) in CMOS SRAM, IEEE Transactions onNuclear Science, 43(6), 2879-2888, Aralık 1996.

[16] Calvel, P., Lamothe, P., Barillot, C., Ecoffet, R., Duzellier, S., Stassinopoulos, E.G., Space radiation evaluation of 16 Mbit DRAMs for mass memory applications, IEEE Transactions onNuclear Science, 41(6), 2267-2271, Aralık 1994.

[17] "Xilinx UG191 Virtex-5 FPGA Configuration User Guide" erişim adresi: www.xilinx.com/support/documentation/user_guides_ug191.pdf, erişim tarihi: 1 Aralık 2012.

[18] S. Lin and D. J. Costello, Error Control Coding, *Prentice-Hall*, Englewood Cliffs, New Jersey, 2004.

[19] Shannon C.E., A Mathematical Theory of Communication, The Bell System Technical Journal, 27, 379–423, 623–656, Ekim 1948.

[20] Seifert, N., Xiaowei Zhu, Massengill, L.W., Impact of scaling on soft-error rates in commercial microprocessors, IEEE Transactions onNuclear Science, 49(6), 3100- 3106, Aralık 2002.

[21] Naeimi, H., DeHon, A., Fault Secure Encoder and Decoder for Memory Applications, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 17(4), 473-486, Nisan 2009.

[22] Naeimi, H., DeHon, A., , Fault Secure Encoder and Decoder for Memory Applications, 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems, 409-417, Eylül 2007.

[23] Sipser M., Spielman D., Expander codes, IEEE Transactions on Information Theory, 42(6), 1710–1722, Kasım 1996.

[24] Berg M., Fault Tolerance Implementation within SRAM Based FPGA Design Based upon the Increased Level of Single Event Upset Susceptibility,Proceedings of the 12th IEEE International Symposium on On-Line Testing, 2006.

[25] Cieslewski, G., Jacobs, A., George, A.D.,Fault-Tolerant 2D Fourier Transform with Checksum Encoding, IEEE Aerospace Conference, 1-11, Mart 2007.

[26] Durna, M., Atar, O., Ceylan, M., Cakmakci, Y., Demirci, M., Kozal, O.A., Oturak, M., Ozdemir, A., Turhan, O.,On board data handling subsystem featuring BiLGE, 5th International Conference on Recent Advances in Space Technologies, 932-937, Haziran 2011.

[27] Radaelli, D., Puchner, H., Skip Wong, Daniel, S.,Investigation of multi-bit upsets in a 150 nm technology SRAM device, IEEE Transactions on Nuclear Science, 52(6), 2433- 2437, Aralık 2005.

[28] Song, Y., Vu, K.N., Cable, J.S., Witteles, A.A., Kolasinski, W.A., Koga, R., Elder, J.H., Osborn, J.V., Martin, R.C., Ghoniem, N.M.,Experimental and analytical investigation of single event, multiple bit upsets in poly-silicon load, 64 K×1 NMOS SRAMs, IEEE Transactions on Nuclear Science, 35(6), 1673-1677, Aralık 1988.

[29] Naseer, R., Draper, J.,Parallel double error correcting code design to mitigate multi-bit upsets in SRAMs, 34th European Solid-State Circuits Conference, 222-225, Eylül 2008.

[30] Tang, H., Xu, J., Lin, S., Abdel-Ghaffar, K.A.S., Codes on finite geometries, IEEE Transactions on Information Theory, 51(2), 572-596, Şubat 2005.

[31] Calvel, P., Lamothe, P., Barillot, C., Ecoffet, R., Duzellier, S., Stassinopoulos, E.G.,Space radiation evaluation of 16 Mbit DRAMs for mass memory applications, IEEE Transactions on Nuclear Science, 41(6), 2267-2271, Aralık 1994.

[32] Johansson, K., Ohlsson, M., Olsson, N., Blomgren, J., Renberg, P.-U.,Neutron induced single-word multiple-bit upset in SRAM, IEEE Transactions on Nuclear Science, 46(6), 1427-1433, Aralık 1999.

[33] Makihara, A., Shindou, H., Nemoto, N., Kuboyama, S., Matsuda, S., Oshima, T., Hirao, T., Itoh, H., Buchner, S., Campbell, A.B.,Analysis of single-ion multiple-bit upset in high-density DRAMs, IEEE Transactions on Nuclear Science, 47(6), 2400-2404, Aralık 2000.

[34] Autran, J.L., Serre, S., Munteanu, D., Martinie, S., Semikh, S., Sauze, S., Uznanski, S., Gasiot, G., Roche, P.,Real-time Soft-Error testing of 40nm SRAMs, IEEE International Reliability Physics Symposium, 3(5), 1-9, Nisan 2012.

[35] Rollins, N., Fuller, M., Wirthlin, M.J.,A Comparison of fault-tolerant memories in SRAM-based FPGAs, IEEEAerospace Conference, 1-12, Mart 2010.

[36] Quinn, H., Morgan, K., Graham, P., Krone, J., Caffrey, M., Lundgreen, K.,Domain Crossing Errors: Limitations on Single Device Triple-Modular Redundancy Circuits in Xilinx FPGAs, IEEE Transactions onNuclear Science, 54(6), 2037-2043, Aralık 2007.

[37] Quinn, H., Graham, P., Krone, J., Caffrey, M., Rezgui, S.,Radiation-induced multi-bit upsets in SRAM-based FPGAs, IEEE Transactions on Nuclear Science, 52(6), 2455- 2461, Aralık 2005.

[38] Quinn, H., Graham, P., Morgan, K., Baker, Z., Caffrey, M., Smith, D., Bell, R., On-Orbit Results for the Xilinx Virtex-4 FPGA, IEEERadiation Effects Data Workshop, 1-8, Temmuz 2012.

[39] Cannon, E.H., Gordon, M.S., Heidel, D.F., Kleinosowski, A.J., Oldiges, P., Rodbell, K.P., Tang, H.,Multi-bit upsets in 65nm SOI SRAMs,IEEE InternationalReliability Physics Symposium, 195-201, Nisan- Mayıs 2008.

[40] Sanghyeon Baeg, Shijie Wen, Wong, R.,SRAM Interleaving Distance Selection With a Soft Error Failure Model,IEEE Transactions onNuclear Science, 56(4), 2111-2118, Ağustos 2009.

[41] Ming Zhu, Liyi Xiao, Shuhao Li, Yanjing Zhang,Efficient Two-Dimensional Error Codes for Multiple Bit Upsets Mitigation in Memory, IEEE 25th International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT),129-135, Ekim 2010.

[42] Bentoutou, Y., Djaifri, M.,Observations of single-event upsets and multiple-bit upsets in random access memories on-board the Algerian satellite,IEEENuclear Science Symposium Conference Record,2568-2570, Ekim 2008.

[43] Bentoutou, Y.,Efficient Memory Error Coding for Space Computer Applications, Information and Communication Technologies, 2, 2347-2352, 2006.

[44] Pontarelli, S., Cardarilli, G.C., Re, M., Salsano, A.,Error Correction Codes for SEU and SEFI Tolerant Memory Systems, 24th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 425-430, Ekim 2009.

[45] Rockett, L., Patel, D., Danziger, S., Cronquist, B., Wang, J.J.,Radiation Hardened FPGA Technology for Space Applications, IEEE Aerospace Conference,1-7, Mart 2007.

[46] Shyue-Win Wei, Che-Ho Wei,High-speed hardware decoder for double-error-correcting binary BCH codes, IEEE Proceedings Communications, Speech and Vision, 136(3), 227- 231, Haziran 1989.

[47] El-Medany, W.M., Harrison, C.G., Garrell, P.G., Hardy, C.J., , VHDL implmentation of a BCH minimum weight decoder for double error, Proceedings of the Eighteenth National Radio Science Conference, vol.2, 361-368, 2001.

[48] Frigerio, L., Radaelli, M.A., Salice, F.,Convolutional Coding for SEU mitigation, 13th European Test Symposium, 2008, 191-196, Mayıs 2008.

[49] Poolakkaparambil, M., Mathew, J., Jabir, A., Multiple Bit Error Tolerant Galois Field Architectures Over GF (2^m). Electronics2012, 1, 3-22, 2012.

[50] Seungjune Jeon, Euiseok Hwang, Kumar, B.V.K.V., Cheng, M.K.,LDPC Codes for Memory Systems with Scrubbing, 2010 IEEE Global Telecommunications Conference (GLOBECOM 2010), ,1-6, Aralık 2010.

[51] Cetin E., Diessel O., Guaranteed Fault Recovery Time for FPGA-based TMR Circuits Employing Partial Reconfiguration, DAC Workshop 2nd International Workshop on Computing in Heterogeneous, Autonomous 'N' Goal-oriented Environments, 2012

[52] Argyrides, C., Zarandi, H.R., Pradhan, D.K.,Multiple Upsets Tolerance in SRAM Memory, IEEE International Symposium onCircuits and Systems, 365-368, Mayıs 2007.

[53] Kou, Y., Lin, S., Fossorier, M.P.C.,Low-density parity-check codes based on finite geometries: a rediscovery and new results, IEEE Transactions on Information Theory, 47(7), 2711-2736, Kasım 2001.

[54] Reviriego, P., Maestro, J.A., Flanagan, M.F.,Error Detection in Majority Logic Decoding of Euclidean Geometry Low Density Parity Check (EG-LDPC) Codes, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 21(1), 1, 156-159, Ocak 2013.

[55] Xiao Li Yi, Zhu Ming, Zhang Yan Jing, Luo Hong Wei, Design EG-LDPC codes for soft error mitigation in memory, Academic International Symposium on Optoelectronics and Microelectronics Technology (AISOMT), 328-332, Ekim 2011.

[56] "Xilinx Virtex-Family Scrubbing Methodologies Developed by The NASA Goddard Radiation Effects and Analysis Group (REAG)" erişim adresi http://microelectronics.esa.int/fiws/WFIFT_P3_NASAGoddardScrubbing.pps , erişim tarihi 13.11.2012

[57] Martin, Q., George, A.D.,Scrubbing optimization via availability prediction (SOAP) for reconfigurable space computing, 2012 IEEE Conference on High Performance Extreme Computing (HPEC), Eylül 2012.

[58] Lyke, J.,Reconfigurable systems: A generalization of reconfigurable computational strategies for space systems, IEEE Aerospace Conference Proceedings,4(4), 1935- 1950, 2002.

[59] Katz, R., LaBel, K., Wang, J.J., Cronquist, B., Koga, R., Penzin, S., Swift, G., Radiation effects on current field programmable technologies, IEEE Transactions on Nuclear Science, 44(9),1945-1956, Aralık 1997.

[60] Reviriego, P., Flanagan, M. F., Liu, S.-F., Maestro, J. A.,On the Use of Euclidean Geometry Codes for Efficient Multibit Error Correction on Memory Systems, IEEE Transactions onNuclear Science, 59(4), 824-828, Ağustos 2012.

[61] Argyrides, C., Zarandi, H.R., Pradhan, D.K., Matrix Codes: Multiple Bit Upsets Tolerant Method for SRAM Memories, 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems, 340-348, Eylül 2007.

[62] Gustavo Neuberger, Fernanda de Lima, Luigi Carro, and Ricardo Reis, A multiple bit upset tolerant SRAM memory. ACM Trans. Des. Autom. Electron. Syst. 8(4) 577-590, Ekim 2003.

EKLER

A. Sistematik Üreteç Matrisi Elde Etmek için Kullanılan MATLAB Kaynak Kodu

Aşağıda verilen kod girdi olarakGF(2)'de tanımlı bir etki vektörü alarak sistematik yapıda bir üreteç matris oluşturur.

```
function [G_sys] = Gmat(x)
G=[x;
   circshift(x,[0 1]);
   circshift(x,[0 2]);
   circshift(x,[0 3]);
   circshift(x,[0 4]);
   circshift(x,[0 5]);
   circshift(x,[0 6]);
   circshift(x,[0 7]);
   circshift(x,[0 8]);
   circshift(x,[0 9]);
   circshift(x,[0 10]);
   circshift(x,[0 11]);
   circshift(x,[0 12]);
   circshift(x,[0 13]);
   circshift(x,[0 14]);
   circshift(x,[0 15]);
   circshift(x,[0 16]);
   circshift(x,[0 17]);
   circshift(x,[0 18]);
   circshift(x,[0 19]);
   circshift(x,[0 20]);
   circshift(x,[0 21]);
   circshift(x,[0 22]);
   circshift(x,[0 23]);
   circshift(x,[0 24]);
   circshift(x,[0 25]);
   circshift(x,[0 26]);
   circshift(x,[0 27]);
   circshift(x,[0 28]);
   circshift(x,[0 29]);
   circshift(x,[0 30]);
   circshift(x,[0 31]);
   circshift(x,[0 32]);
   circshift(x,[0 33]);
   circshift(x,[0 34]);
   circshift(x,[0 35]);
   circshift(x,[0 36])
];

[row column]=size(G);
G_sys=G;
for ii=1:row
```

```
for mm=1:column
    if G_sys(ii,mm)==1
        aa=mm;
break;
    end
end

    for ll=1:row
        if G_sys(ll,aa)==1
            if ll~=ii
            G_sys(ll,:)=xor(G_sys(ll,:),G_sys(ii,:));
            end
        end
    end
end
```

B. (63,37,9) EG-LDPC Kodu Kod Çözücü Devresi Kaynak Kodu

Aşağıda verilen VHDL kaynak kodu (63,37,9) EG-LDPC kodunun kod çözücü devresinin gerçeklemesi için yazılmıştır. Tüm bitler için benzer kod çözücü devreleri kullanıldığı için örnek olarak sadece kod sözcüğünün ilk 4 bitinin düzeltilmesi için gerekli kod parçası verilmiştir.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity decoder is
    Port ( cw : in  STD_LOGIC_VECTOR (62 downto 0);
                    clk: in STD_LOGIC;
message : out  STD_LOGIC_VECTOR (62 downto 0));
end decoder;

architecture Behavioral of decoder is

component majority
port (      A : in  STD_LOGIC;
B : in  STD_LOGIC;
C : in  STD_LOGIC;
D : in  STD_LOGIC;
E : in  STD_LOGIC;
F : in  STD_LOGIC;
G : in  STD_LOGIC;
H : in  STD_LOGIC;
O : out  STD_LOGIC);
end component;


signal cw_dec  : STD_LOGIC_VECTOR (62 downto 0);
signal synd: STD_LOGIC_VECTOR (62 downto 0);
signal syndrome : STD_LOGIC;

signal c_s0 : STD_LOGIC_VECTOR (7 downto 0);
signal c_s1 : STD_LOGIC_VECTOR (7 downto 0);
signal c_s2 : STD_LOGIC_VECTOR (7 downto 0);
signal c_s3 : STD_LOGIC_VECTOR (7 downto 0);
signal c_s4 : STD_LOGIC_VECTOR (7 downto 0);
signal c_s5 : STD_LOGIC_VECTOR (7 downto 0);
signal c_s6 : STD_LOGIC_VECTOR (7 downto 0);
signal c_s7 : STD_LOGIC_VECTOR (7 downto 0);
signal c_s8 : STD_LOGIC_VECTOR (7 downto 0);
signal c_s9 : STD_LOGIC_VECTOR (7 downto 0);
signal c_s10: STD_LOGIC_VECTOR (7 downto 0);
signal c_s11: STD_LOGIC_VECTOR (7 downto 0);
```

```
signal c_s12: STD_LOGIC_VECTOR (7 downto 0);
signal c_s13: STD_LOGIC_VECTOR (7 downto 0);
signal c_s14: STD_LOGIC_VECTOR (7 downto 0);
signal c_s15: STD_LOGIC_VECTOR (7 downto 0);
signal c_s16: STD_LOGIC_VECTOR (7 downto 0);
signal c_s17: STD_LOGIC_VECTOR (7 downto 0);
signal c_s18: STD_LOGIC_VECTOR (7 downto 0);
signal c_s19: STD_LOGIC_VECTOR (7 downto 0);
signal c_s20: STD_LOGIC_VECTOR (7 downto 0);
signal c_s21: STD_LOGIC_VECTOR (7 downto 0);
signal c_s22: STD_LOGIC_VECTOR (7 downto 0);
signal c_s23: STD_LOGIC_VECTOR (7 downto 0);
signal c_s24: STD_LOGIC_VECTOR (7 downto 0);
signal c_s25: STD_LOGIC_VECTOR (7 downto 0);
signal c_s26: STD_LOGIC_VECTOR (7 downto 0);
signal c_s27: STD_LOGIC_VECTOR (7 downto 0);
signal c_s28: STD_LOGIC_VECTOR (7 downto 0);
signal c_s29: STD_LOGIC_VECTOR (7 downto 0);
signal c_s30: STD_LOGIC_VECTOR (7 downto 0);
signal c_s31: STD_LOGIC_VECTOR (7 downto 0);
signal c_s32: STD_LOGIC_VECTOR (7 downto 0);
signal c_s33: STD_LOGIC_VECTOR (7 downto 0);
signal c_s34: STD_LOGIC_VECTOR (7 downto 0);
signal c_s35: STD_LOGIC_VECTOR (7 downto 0);
signal c_s36: STD_LOGIC_VECTOR (7 downto 0);
signal c_s37: STD_LOGIC_VECTOR (7 downto 0);
signal c_s38: STD_LOGIC_VECTOR (7 downto 0);
signal c_s39: STD_LOGIC_VECTOR (7 downto 0);
signal c_s40: STD_LOGIC_VECTOR (7 downto 0);
signal c_s41: STD_LOGIC_VECTOR (7 downto 0);
signal c_s42: STD_LOGIC_VECTOR (7 downto 0);
signal c_s43: STD_LOGIC_VECTOR (7 downto 0);
signal c_s44: STD_LOGIC_VECTOR (7 downto 0);
signal c_s45: STD_LOGIC_VECTOR (7 downto 0);
signal c_s46: STD_LOGIC_VECTOR (7 downto 0);
signal c_s47: STD_LOGIC_VECTOR (7 downto 0);
signal c_s48: STD_LOGIC_VECTOR (7 downto 0);
signal c_s49: STD_LOGIC_VECTOR (7 downto 0);
signal c_s50: STD_LOGIC_VECTOR (7 downto 0);
signal c_s51: STD_LOGIC_VECTOR (7 downto 0);
signal c_s52: STD_LOGIC_VECTOR (7 downto 0);
signal c_s53: STD_LOGIC_VECTOR (7 downto 0);
signal c_s54: STD_LOGIC_VECTOR (7 downto 0);
signal c_s55: STD_LOGIC_VECTOR (7 downto 0);
signal c_s56: STD_LOGIC_VECTOR (7 downto 0);
signal c_s57: STD_LOGIC_VECTOR (7 downto 0);
signal c_s58: STD_LOGIC_VECTOR (7 downto 0);
signal c_s59: STD_LOGIC_VECTOR (7 downto 0);
signal c_s60: STD_LOGIC_VECTOR (7 downto 0);
signal c_s61: STD_LOGIC_VECTOR (7 downto 0);
signal c_s62: STD_LOGIC_VECTOR (7 downto 0);

signal cs01: STD_LOGIC;
signal cs02 : STD_LOGIC;
signal cs03 : STD_LOGIC;
signal cs05 : STD_LOGIC;
signal cs06 : STD_LOGIC;
signal cs012 : STD_LOGIC;
```

```
signal cs010  : STD_LOGIC;
signal cs024  : STD_LOGIC;
signal cs12 : STD_LOGIC;
signal cs13 : STD_LOGIC;
signal cs16 : STD_LOGIC;
signal cs17 : STD_LOGIC;
signal cs111 : STD_LOGIC;
signal cs113 : STD_LOGIC;
signal cs125 : STD_LOGIC;
signal cs23 : STD_LOGIC;
signal cs24 : STD_LOGIC;
signal cs27 : STD_LOGIC;
signal cs212 : STD_LOGIC;
signal cs214 : STD_LOGIC;
signal cs226 : STD_LOGIC;
signal cs34 : STD_LOGIC;
signal cs35 : STD_LOGIC;
signal cs38 : STD_LOGIC;
signal cs313 : STD_LOGIC;
signal cs327 : STD_LOGIC;
signal cs45 : STD_LOGIC;
signal cs46 : STD_LOGIC;
signal cs49 : STD_LOGIC;
signal cs414 : STD_LOGIC;
signal cs428 : STD_LOGIC;
signal cs56 : STD_LOGIC;
signal cs510 : STD_LOGIC;
signal cs515 : STD_LOGIC;
signal cs529 : STD_LOGIC;
signal cs67 : STD_LOGIC;
signal cs616 : STD_LOGIC;
signal cs611 : STD_LOGIC;
signal cs78 : STD_LOGIC;
signal cs712 : STD_LOGIC;
signal cs717: STD_LOGIC;
signal cs89 : STD_LOGIC;
signal cs813 : STD_LOGIC;
signal cs818 : STD_LOGIC;
signal cs910 : STD_LOGIC;
signal cs914 : STD_LOGIC;
signal cs919 : STD_LOGIC;
signal cs1015 : STD_LOGIC;
signal cs1020: STD_LOGIC;
signal cs1116: STD_LOGIC;
signal cs1121: STD_LOGIC;
signal cs1222: STD_LOGIC;
signal cs1323: STD_LOGIC;
signal cs1424: STD_LOGIC;
signal cs1525: STD_LOGIC;
signal cs1626: STD_LOGIC;
signal cs1727: STD_LOGIC;
signal cs1828: STD_LOGIC;
signal cs1929: STD_LOGIC;
signal cs2030: STD_LOGIC;
signal cs2131: STD_LOGIC;
signal cs2232: STD_LOGIC;
signal cs2333: STD_LOGIC;
```

```
signal m0   : STD_LOGIC;
signal m1   : STD_LOGIC;
signal m2   : STD_LOGIC;
signal m3   : STD_LOGIC;
signal m58  : STD_LOGIC;
signal m59  : STD_LOGIC;
signal m60  : STD_LOGIC;
signal m61  : STD_LOGIC;
signal m62  : STD_LOGIC;


begin

cs01 <= cw(0) xor cw(1) xor cw(4)xor cw(16) xor cw(21) xor cw(23)
xor cw(29) xor cw(53);
cs02 <= cw(0) xor cw(2) xor cw(8)xor cw(32) xor cw(42) xor cw(43)
xor cw(46) xor cw(58);
cs03 <= cw(0) xor cw(3) xor cw(15)xor cw(20) xor cw(22) xor cw(28)
xor cw(52) xor cw(62);
cs05 <= cw(0) xor cw(5) xor cw(7)xor cw(13) xor cw(37) xor cw(47)
xor cw(48) xor cw(51);
cs06 <= cw(0) xor cw(6) xor cw(30)xor cw(40) xor cw(41) xor cw(44)
xor cw(56) xor cw(61);
cs012 <= cw(0) xor cw(12) xor cw(17)xor cw(19) xor cw(25) xor cw(49)
xor cw(59) xor cw(60);
cs010  <= cw(0) xor cw(10) xor cw(11)xor cw(14) xor cw(26) xor
cw(31) xor cw(33) xor cw(39);
cs024  <= cw(0) xor cw(24) xor cw(34)xor cw(35) xor cw(38) xor
cw(50) xor cw(55) xor cw(57);
cs12 <= cw(1) xor cw(2) xor cw(5)xor cw(17) xor cw(22) xor cw(24)
xor cw(30) xor cw(54);
cs13 <= cw(1) xor cw(3) xor cw(9)xor cw(33) xor cw(43) xor cw(44)
xor cw(47) xor cw(59);
cs16 <= cw(1) xor cw(6) xor cw(8)xor cw(14) xor cw(38) xor cw(48)
xor cw(49) xor cw(52);
cs17 <= cw(1) xor cw(7) xor cw(31)xor cw(41) xor cw(42) xor cw(45)
xor cw(57) xor cw(62);
cs111 <= cw(1) xor cw(11) xor cw(12)xor cw(15) xor cw(27) xor cw(32)
xor cw(34) xor cw(40);
cs113 <= cw(1) xor cw(13) xor cw(18)xor cw(20) xor cw(26) xor cw(50)
xor cw(60) xor cw(61);
cs125 <= cw(1) xor cw(25) xor cw(35)xor cw(36) xor cw(39) xor cw(51)
xor cw(56) xor cw(58);
cs23 <= cw(2) xor cw(3) xor cw(6)xor cw(18) xor cw(23) xor cw(25)
xor cw(31) xor cw(55);
cs24 <= cw(2) xor cw(4) xor cw(10)xor cw(34) xor cw(44) xor cw(45)
xor cw(48) xor cw(60);
cs27 <= cw(2) xor cw(7) xor cw(9)xor cw(15) xor cw(39) xor cw(49)
xor cw(50) xor cw(53);
cs212 <= cw(2) xor cw(12) xor cw(13)xor cw(16) xor cw(28) xor cw(33)
xor cw(35) xor cw(21);
cs214 <= cw(2) xor cw(14) xor cw(19)xor cw(21) xor cw(27) xor cw(52)
xor cw(61) xor cw(62);
cs226 <= cw(2) xor cw(26) xor cw(36)xor cw(37) xor cw(40) xor cw(52)
xor cw(57) xor cw(59);
cs34 <= cw(3) xor cw(4) xor cw(7)xor cw(19) xor cw(24) xor cw(26)
xor cw(32) xor cw(56);
```

```
cs35 <= cw(3) xor cw(5) xor cw(11)xor cw(35) xor cw(45) xor cw(46)
xor cw(49) xor cw(61);
cs38 <= cw(3) xor cw(8) xor cw(10)xor cw(16) xor cw(40) xor cw(50)
xor cw(51) xor cw(54);
cs313  <= cw(3) xor cw(13) xor cw(14)xor cw(17) xor cw(29) xor
cw(34) xor cw(36) xor cw(42);
cs327 <= cw(3) xor cw(27) xor cw(37)xor cw(38) xor cw(41) xor cw(53)
xor cw(58) xor cw(60);
cs45 <= cw(4) xor cw(5) xor cw(8)xor cw(20) xor cw(25) xor cw(27)
xor cw(33) xor cw(57);
cs46 <= cw(4) xor cw(6) xor cw(12)xor cw(36) xor cw(46) xor cw(47)
xor cw(50) xor cw(62);
cs49 <= cw(4) xor cw(9) xor cw(11)xor cw(17) xor cw(41) xor cw(51)
xor cw(52) xor cw(55);
cs414 <= cw(4) xor cw(14) xor cw(15)xor cw(18) xor cw(30) xor cw(35)
xor cw(37) xor cw(43);
cs428 <= cw(4) xor cw(28) xor cw(38)xor cw(39) xor cw(42) xor cw(54)
xor cw(59) xor cw(61);
cs56 <= cw(5) xor cw(6) xor cw(9)xor cw(21) xor cw(26) xor cw(28)
xor cw(34) xor cw(58);
cs510 <= cw(5) xor cw(10) xor cw(12)xor cw(18) xor cw(42) xor cw(52)
xor cw(53) xor cw(56);
cs515 <= cw(5) xor cw(15) xor cw(16)xor cw(19) xor cw(31) xor cw(36)
xor cw(38) xor cw(44);
cs529 <= cw(5) xor cw(29) xor cw(39)xor cw(40) xor cw(43) xor cw(55)
xor cw(60) xor cw(62);
cs67 <= cw(6) xor cw(7) xor cw(10)xor cw(22) xor cw(27) xor cw(29)
xor cw(35) xor cw(59);
cs616 <= cw(6) xor cw(16) xor cw(17)xor cw(20) xor cw(32) xor cw(37)
xor cw(39) xor cw(45);
cs611 <= cw(6) xor cw(11) xor cw(13)xor cw(19) xor cw(43) xor cw(53)
xor cw(54) xor cw(57);
cs78 <= cw(7) xor cw(8) xor cw(11)xor cw(23) xor cw(28) xor cw(30)
xor cw(36) xor cw(60);
cs712 <= cw(7) xor cw(12) xor cw(14)xor cw(20) xor cw(44) xor cw(54)
xor cw(55) xor cw(58);
cs717<= cw(7) xor cw(17) xor cw(18)xor cw(21) xor cw(33) xor cw(38)
xor cw(40) xor cw(46);
cs89 <= cw(8) xor cw(9) xor cw(12)xor cw(24) xor cw(29) xor cw(31)
xor cw(37) xor cw(61);
cs813 <= cw(8) xor cw(13) xor cw(15)xor cw(21) xor cw(45) xor cw(55)
xor cw(56) xor cw(59);
cs818 <= cw(8) xor cw(18) xor cw(19)xor cw(22) xor cw(34) xor cw(39)
xor cw(41) xor cw(47);
cs910 <= cw(9) xor cw(10) xor cw(13)xor cw(25) xor cw(30) xor cw(32)
xor cw(38) xor cw(62);
cs914 <= cw(9) xor cw(14) xor cw(16)xor cw(22) xor cw(46) xor cw(56)
xor cw(57) xor cw(60);
cs919 <= cw(9) xor cw(19) xor cw(20)xor cw(23) xor cw(35) xor cw(40)
xor cw(42) xor cw(48);
cs1015 <= cw(10) xor cw(15) xor cw(17)xor cw(23) xor cw(47) xor
cw(57) xor cw(58) xor cw(61);
cs1020<= cw(10) xor cw(20) xor cw(21)xor cw(24) xor cw(36) xor
cw(41) xor cw(43) xor cw(49);
cs1116 <= cw(11) xor cw(16) xor cw(18)xor cw(24) xor cw(48) xor
cw(58) xor cw(59) xor cw(62);
cs1121 <= cw(11) xor cw(21) xor cw(22)xor cw(25) xor cw(37) xor
cw(42) xor cw(44) xor cw(50);
```

```
cs1222 <= cw(12) xor cw(22) xor cw(23)xor cw(26) xor cw(38) xor
cw(43) xor cw(45) xor cw(51);
cs1323 <= cw(13) xor cw(23) xor cw(24)xor cw(27) xor cw(39) xor
cw(44) xor cw(46) xor cw(52);
cs1424 <= cw(14) xor cw(24) xor cw(25)xor cw(28) xor cw(40) xor
cw(45) xor cw(47) xor cw(53);
cs1525 <= cw(15) xor cw(25) xor cw(26)xor cw(29) xor cw(41) xor
cw(46) xor cw(48) xor cw(54);
cs1626 <= cw(16) xor cw(26) xor cw(27)xor cw(30) xor cw(42) xor
cw(47) xor cw(49) xor cw(55);
cs1727 <= cw(17) xor cw(27) xor cw(28)xor cw(31) xor cw(40) xor
cw(48) xor cw(50) xor cw(56);
cs1828 <= cw(18) xor cw(28) xor cw(29)xor cw(32) xor cw(44) xor
cw(49) xor cw(51) xor cw(57);
cs1929 <= cw(19) xor cw(29) xor cw(30)xor cw(33) xor cw(45) xor
cw(50) xor cw(52) xor cw(58);
cs2030 <= cw(20) xor cw(30) xor cw(31)xor cw(34) xor cw(46) xor
cw(51) xor cw(53) xor cw(59);
cs2131 <= cw(21) xor cw(31) xor cw(32)xor cw(35) xor cw(47) xor
cw(52) xor cw(54) xor cw(60);
cs2232 <= cw(22) xor cw(32) xor cw(33)xor cw(36) xor cw(48) xor
cw(53) xor cw(55) xor cw(61);
cs2333 <= cw(23) xor cw(33) xor cw(34)xor cw(37) xor cw(49) xor
cw(54) xor cw(56) xor cw(62);



c_s0(0) <= cs06;
c_s0(1) <= cs02;
c_s0(2) <= cs05;
c_s0(3) <= cs012;
c_s0(4) <= cs03;
c_s0(5) <= cs010;
c_s0(6) <= cs01;
c_s0(7) <= cs024;

MAJ0: majority port
map(c_s0(0),c_s0(1),c_s0(2),c_s0(3),c_s0(4),c_s0(5),c_s0(6),c_s0(7),
m0);
cw_dec(0) <= m0 xor cw(0);


c_s1(0) <= cs17;
c_s1(1) <= cs16;
c_s1(2) <= cs113;
c_s1(3) <= cs01;
c_s1(4) <= cs12;
c_s1(5) <= cs111;
c_s1(6) <= cs125;
c_s1(7) <= cs13;

MAJ1: majority port
map(c_s1(0),c_s0(1),c_s1(2),c_s1(3),c_s1(4),c_s1(5),c_s1(6),c_s1(7),
m1);
cw_dec(1) <= m1 xor cw(1);

c_s2(0) <= cs214;
c_s2(1) <= cs23;
```

```
c_s2(2) <= cs212;
c_s2(3) <= cs226;
c_s2(4) <= cs02;
c_s2(5) <= cs24;
c_s2(6) <= cs27;
c_s2(7) <= cs12;

MAJ2: majority port
map(c_s2(0),c_s2(1),c_s2(2),c_s2(3),c_s2(4),c_s2(5),c_s2(6),c_s2(7),
m2);
cw_dec(2) <= m2 xor cw(2);

c_s3(0) <= cs03;
c_s3(1) <= cs23;
c_s3(2) <= cs34;
c_s3(3) <= cs313;
c_s3(4) <= cs327;
c_s3(5) <= cs35;
c_s3(6) <= cs38;
c_s3(7) <= cs13;

MAJ3: majority port
map(c_s3(0),c_s3(1),c_s3(2),c_s3(3),c_s3(4),c_s3(5),c_s3(6),c_s3(7),
m3);
cw_dec(3) <= m3 xor cw(3);
```

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : DEMİRCİ, Mustafa

Uyruğu : T.C.

Doğum tarihi ve yeri : 22.05.1986 Ankara

Medeni hali : Evli

Telefon : 0 (533) 819 13 41

Faks :

e-mail : mdemirci@etu.edu.tr

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	Bilkent Üniversitesi / Elektrik ve Elektronik Müh.	2007

İş Deneyimi

Yıl	Yer	Görev
2012-…	ASELSAN	Uzman Mühendis
2007-2012	TÜBİTAK UZAY Teknolojileri Araştırma Enstitüsü	Araştırmacı

Yabancı Dil

İngilizce

Yayınlar

Durna, M., Atar, O., Ceylan, M., Cakmakci, Y., Demirci, M., Kozal, O.A., Oturak, M., Ozdemir, A., Turhan, O., On board data handling subsystem featuring BiLGE, 5th International Conference on Recent Advances in Space Technologies, 932-937, Haziran 2011.