HAVACILIK UYGULAMALARI İÇİN KISMİ DİNAMİK DONANIM ŞEKİLLENDİRME VE ÜÇLÜ MODÜL YEDEKLEME KULLANILARAK HATA TOLERANSLI FPGA TASARIMI

FAULT TOLERANT FPGA DESIGN FOR AEROSPACE APPLICATIONS WITH USING PARTIAL DYNAMIC RECONFIGURATION AND TRIPLE MODULE REDUNDANCY

ALİCAN DÖNMEZ

PROF. DR. ALİ ZİYA ALKAR
Tez Danışmanı

Hacettepe Üniversitesi
Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin
ELEKTRİK ve ELEKTRONİK Mühendisliği Anabilim Dalı için Öngördüğü
YÜKSEK LİSANS TEZİ olarak hazırlanmıştır.

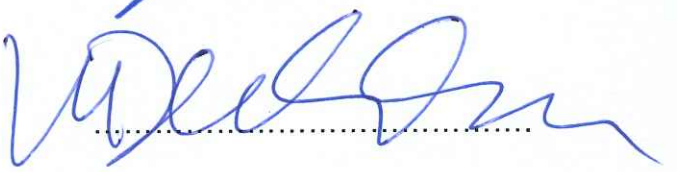
2019

ALİCAN DÖNMEZ 'in hazırladığı "**Havacılık Uygulamaları İçin Kısmi Dinamik Donanım Şekillendirme ve Üçlü Yedekleme Kullanılarak Hata Toleranslı FPGA Tasarımı**" adlı bu çalışma aşağıdaki jüri tarafından **ELEKTRİK ve ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI**'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Doç.Dr. Cüneyt Bazlamaçcı
Başkan ..... 

Prof.Dr. Ali Ziya Alkar
Danışman ..... 

Dr.Öğr.Üyesi Dinçer Gökcen
Üye ..... 

Dr.Öğr.Üyesi Derya Altunay
Üye ..... 

Doç.Dr. Harun Artuner
Üye ..... 

Bu tez Hacettepe Üniversitesi Fen Bilimleri Enstitüsü tarafından **YÜKSEK LİSANS TEZİ** olarak ... / ... /.... tarihinde onaylanmıştır.

Prof. Dr. Menemşe GÜMÜŞDERELİOĞLU
Fen Bilimleri Enstitüsü Müdürü

Fark yaratan mühendislere
ve onları yetiştiren insanlara
adanmıştır.

ETİK

Hacettepe Üniversitesi Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında,

- tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- ve bu tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

23 / 09 / 2019

ALİCAN DÖNMEZ

YAYINLANMA FİKRİ MÜLKİYET HAKKLARI BEYANI

Enstitü tarafından onaylanan lisansüstü tezimin/raporumun tamamını veya herhangi bir kısmını, basılı (kağıt) ve elektronik formatta arşivleme ve aşağıda verilen koşullarla kullanıma açma iznini Hacettepe üniversitesine verdiğimi bildiririm. Bu izinle Üniversiteye verilen kullanım hakları dışındaki tüm fikri mülkiyet haklarım bende kalacak, tezimin tamamının ya da bir bölümünün gelecekteki çalışmalarda (makale, kitap, lisans ve patent vb.) kullanım hakları bana ait olacaktır.

Tezin kendi orijinal çalışmam olduğunu, başkalarının haklarını ihlal etmediğimi ve tezimin tek yetkili sahibi olduğumu beyan ve taahhüt ederim. Tezimde yer alan telif hakkı bulunan ve sahiplerinden yazılı izin alınarak kullanması zorunlu metinlerin yazılı izin alarak kullandığımı ve istenildiğinde suretlerini Üniversiteye teslim etmeyi taahhüt ederim.

Yükseköğretim Kurulu tarafından yayınlanan ***Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge*** kapsamında tezim aşağıda belirtilen koşullar haricince YÖK Ulusal Tez Merkezi / H. Ü. Kütüphaneleri Açık Erişim Sisteminde erişime açılır.

- □ Enstitü / Fakülte yönetim kurulu kararı ile tezimin erişime açılması mezuniyet tarihimden itibaren 2 yıl ertelenmiştir.
- □ Enstitü / Fakülte yönetim kurulu gerekçeli kararı ile tezimin erişime açılması mezuniyet tarihimden itibaren .. ay ertelenmiştir.
- □ Tezim ile ilgili gizlilik kararı verilmiştir.

23 / 09 / 2019

ALİCAN DÖNMEZ

ÖZET

HAVACILIK UYGULAMALARI İÇİN KISMİ DİNAMİK DONANIM ŞEKİLLENDİRME VE ÜÇLÜ MODÜL YEDEKLEME KULLANILARAK HATA TOLERANSLI FPGA TASARIMI

ALİCAN DÖNMEZ

Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Danışmanı: Prof.Dr. Ali Ziya Alkar
Eylül 2019, 92 sayfa

Bu tez çalışmasında, havacılık uygulamalarında radyasyon kaynaklı tek bit hatalarının bulunması, düzeltilmesi ve hatanın etkisinin maskelenmesi için yöntemler önerilmiştir. Hatanın bulunması için konfigürasyon belleğinin sürekli olarak tasarımın kendi tarafından okunması yöntemi kullanılmıştır. Hatanın düzeltilmesi için parçalı dinamik donanım şekillendirme yöntemi kullanılmıştır. Hatalı olan konfigürasyon belleği çerçevesi yeniden şekillendirilerek düzeltilmiştir. Hatanın maskelenmesi için üçlü modül yedekleme yöntemi kullanılmıştır. Hata düzeltildikten sonra sadece yapısal bütünlüğün değil, durum bütünlüğünün de korunması sağlanmıştır. Kurulan mimari ARINC-429 protokolü üzerinde, hata yerleştirilerek test edilmiştir. Elde edilen sonuçlarda en kötü durumda yaklaşık 67ms içerisinde oluşan tek bit hatasının düzeltildiği görülmüştür. Düzeltilemeyecek kritik hatalar yaratılamamış fakat istatiksel hesabı yapılmıştır. Önerilen sistemin kullanılmadığı duruma göre hata ihtimalinin yaklaşık 200 bin kat daha az olduğu görülmüştür. Yapılan hata yerleştirme testinde üçlü yedeklenmiş ARINC-429 modülleri üzerinde oluşan hataların hepsi maskelenebilmiştir. Maskelenemeyecek hataların yaratılması sağlanamamış

fakat istatistiksel hesabı yapılmıştır. Üçlü modül yedekleme yönteminin yaklaşık 3,3 kat daha fazla FPGA alanı kullandığı ölçülmüştür. Sistemde hata bulma ve düzeltme için kullanılan devrenin FPGA alanının yaklaşık iki binde biri olduğu ölçülmüştür. Önerilen sistemin tek bit hatalarına karşı hata toleranslı yapıda olduğu görülmüştür.

Anahtar Kelimeler: Hata Toleranslı FPGA Tasarımı, Parçalı Dinamik Donanım Şekillendirme, Üçlü Modül Yedekleme, Tek Bit Hatası, Üçlü Modül Senkronizasyonu

ABSTRACT

FAULT TOLERANT FPGA DESIGN FOR AEROSPACE APPLICATIONS WITH USING PARTIAL DYNAMIC RECONFIGURATION AND TRIPLE MODULE REDUNDANCY

ALİCAN DÖNMEZ

Master of Science, Department of Electrical and Electronics Engineering
Supervisor: Prof.Dr. Ali Ziya Alkar
September 2019, 92 pages

In this thesis, methods for detecting, correcting, and masking the effect of radiation-induced single-bit errors in aerospace applications are proposed. In order to detect the error, the method of continuous reading of the configuration memory by the design itself was used. Partial dynamic reconfiguration is used to correct the error. The incorrect configuration memory frame has been corrected by reconfiguration. The triple module redundancy method is used to mask the error. After correcting the error, not only the structural integrity, but also the state integrity is maintained. The architecture was tested by injecting an error on ARINC-429 protocol. The results showed that in the worst case, the single bit upset is corrected within approximately 67ms. Critical errors that could not be corrected could not be created, but statistical calculations are made. It is seen that the probability of error is approximately 200 thousand times less than when the proposed system is not used. In the error injection test, all errors occurred on triple redundant ARINC-429 modules could be masked. It was not possible to create errors that could not be masked, but statistical calculations were made. It was measured that the triple module redundancy method uses approximately 3.3 times more FPGA space. It was measured that area of the circuit used for error

detection and correction in the system is approximately one in two thousand of the whole FPGA area. It is seen that the proposed system is fault tolerant against single bit errors.

Keywords: Fault Tolerant FPGA Design, Partial Dynamic Reconfiguration, Triple Module Redundancy, Single Bit Event Upset, Triple Module Synchronization

TEŞEKKÜR

Tez çalışmamın en başından bugüne kadar bilgi, görüş ve yorumlarıyla yolumu aydınlatan danışmanım Sayın Prof.Dr. Ali Ziya Alkar'a değerli katkılarından dolayı teşekkürlerimi sunarım.

Yüksek lisans yapma olanağı sağladığı için ASELSAN'a, tez sürecimdeki desteklerinden dolayı amirlerime ve çalışma arkadaşlarıma teşekkür ederim.

Beni her zaman destekleyen, bugünlere gelmemi sağlayan, varlıklarından güç aldığım anneme, babama ve kardeşlerime çok teşekkür ederim.

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

SİMGELER VE KISALTMALAR

Kısaltmalar

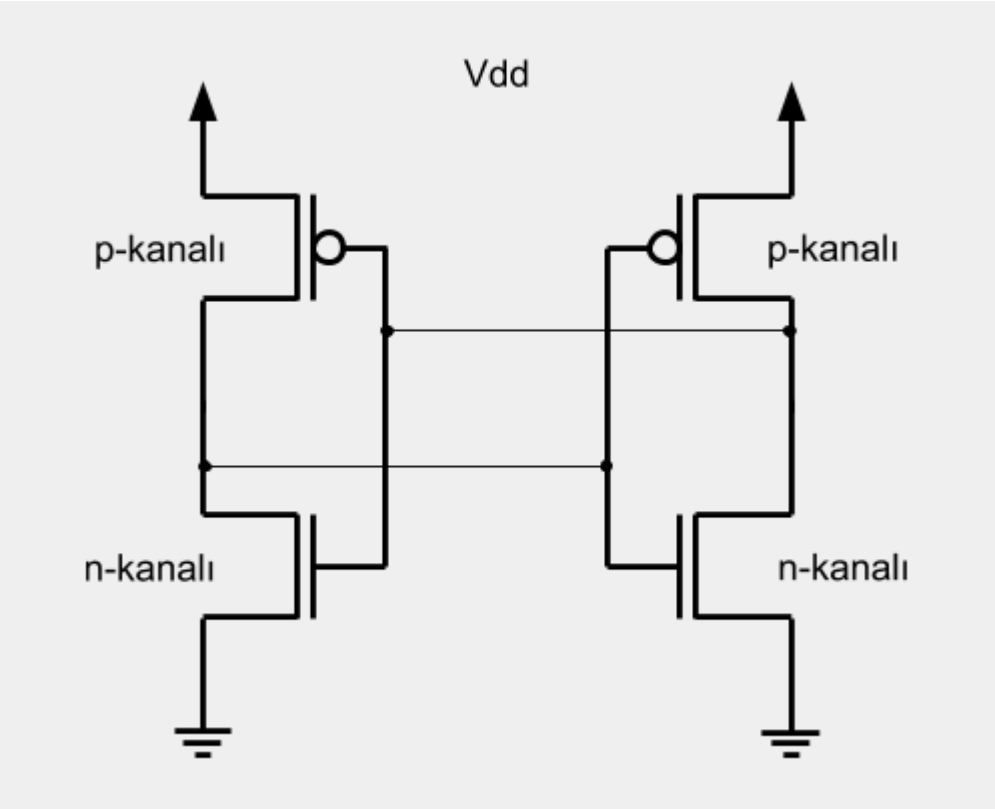
ACK	Alındı Bilgisi (Acknowledge)
ASIC	Uygulamaya Özel Entegre Devre (Application Specific Integrated Circuit)
BCD	İkili Kodlu Ondalık (Binary Coded Decimal)
BIST	Dahili Test (Built-In Self-Test)
BIST DMR	Dahili Testli İkili Modül Yedekli (Built in Self Test With Dual Modul Redundancy)
BNR	İkili Sayı Gösterimi (Binary Number Representation)
CLB	Yapılandırılabilir Mantık Bloğu (Configurable Logic Block)
CRC	Döngüsel Artıklık Kodu (Cyclic Redundancy Code)
CTS	Gönderim Serbest (Clear to Send)
DFF	Çift Flip-Flop (Double Flip-Flop)
DMR	İkili Modül Yedekli (Dual Modul Redundancy)
ECC	Hata Düzeltme Kodu (Error Correction Code)
EFAR	Hata Çerçevesi Adres Kaydı (Error Frame Address Register)
EOT	Gönderim Sonu (End of Transmission)
FAR	Çerçeve Adres Kaydı (Frame Address Register)
FIT	Zaman İçinde Hata (Failure in Time)
FPGA	Proglamlanabilir Mantık Devreleri (Field Programmable Gate Array)
FSM	Sonlu Durum Makinası (Finite State Machine)
GFI	Genel Format Tanımlayıcı (General Format Identifier)
HDL	Donanım Tasarım Dili(Hardware Description Language)
ICAP	Dahili konfigürasyon erişim portu (Internal Configuration Access Port)
IOB	Giriş Çıkış Bloğu (Input Output Block)
LDU	Bağlı Veri Birimleri (Linked Data Units)
LSB	En Az Anlamlı Bit (Least Significant Bit)
LUT	Aramalı Tablo (Look-up Table)

Mb	Megabit
MSB	En Anlamlı Bit (Most Significant Bit)
MTBF	Hatalar Arası Ortalama Zaman (Mean Time Between Failures)
P	Parite (Parity)
PDR	Kısmi Dinamik Donanım Şekillendirme (Partial Dynamic Reconfiguration)
RAM	Rasgele Erişimli Bellek (Random Access Memory)
RR	Yönlendirme Kaynakları (Routing Resources)
RTS	Gönderim İzni (Request to Send)
RZ	Sıfıra-Dönüş (Return to Zero)
RX	Alıcı (Receive)
SEBO	Tek Bit Yanması (Single Event Burnout)
SEL	Tek Bit Takılı Kalması (Single Event Latch Up)
SEU	Tek Bit Değişimi (Single Event Upset)
SET	Tek Bit Geçici Değer (Single Event Transient)
SOC	Yonga Üzerinde Sistem (System on Chip)
SOT	Gönderim Başlangıcı (Start of Transmission)
SRAM	Statik Rasgele Erişimli Bellek (Static Random Access Memory)
SSM	İşaret Durum Matrisi (Sign Status Matrix)
TMR	Üçlü Modül Yedekli (Triple Module Redundant)
Tx	Verici (Transmit)

1. GİRİŞ

Günümüzde programlanabilir mantık devreleri (FPGA) artan performans özellikleri ve düşük maliyetleri ile modern aviyonik sistemlerde yaygın bir şekilde kullanılmaktadır. Flaş tabanlı FPGA'lar statik rasgele erişimli bellek (SRAM) tabanlı FPGA'lara göre radyasyon etkilerine karşı daha dayanıklı fakat performans ve yetenekleri SRAM tabanlı FPGA'lara göre daha kısıtlıdır. Flaş tabanlı FPGA'lar aviyonik sistemlerde başlangıç sıfırlaması, güç açma, sağlık durumu gözleme gibi basit ve kritik görevleri üstlenirken, iletişim ara yüzleri ve sinyal işleme gibi görevlerde SRAM tabanlı FPGA'lar tercih edilmektedir. Daha yüksek performans ve daha düşük güç tüketimi için yeni jenerasyon FPGA'larda silikon üretim boyutları 10nm ölçülere kadar düşmüştür.

FPGA yapısı küçülüp yoğunlaştıkça radyasyon etkilerine karşı daha hassas hale gelmiştir. Kozmik ışınlar ve kozmik ışınların sebebiyle atmosferde açığa çıkan nötronlar ve protonlar, FPGA gibi aygıtlarda hatalara sebep olmaktadır [1, 2]. Bu hataların oluşumu gürültü paylarının küçülmesi ve silikon teknolojisinin küçülmesi ve sıklaşması ile artmaktadır. Radyasyon kaynaklı hatalar SRAM tabanlı FPGA belleklerinde, flaş tabanlı belleklere ve uygulamaya özel entegre devre (ASIC) devrelere göre daha fazla görünmektedir. İyonize radyasyonun SRAM belleği etkilemesinin direk iyonlaşma ve dolaylı iyonlaşma olarak adlandırılan iki temel yolu vardır. Direk iyonlaşma, yüklü parçacığın yarı iletken içerisinden geçerken enerjisini kaybettikçe elektron-delik çifti özgürleştirmesi ile olur. Direk iyonlaşma, kozmik ışınların hataya temel olmasının temel sebebidir. Dolaylı iyonlaşma ise yüklü parçacıklardan ziyade kozmik ışınların atmosferde açığa çıkardığı nötronların ve protonların yarı iletken üzerinde sebep olduğu nükleer reaksiyonların sebebiyle olur. Bu nötronlar ve protonlar kendi başlarına hataya sebep olabilecek enerjide olmasalar bile sebep oldukları nükleer reaksiyonlar sonucunda ortaya çıkan yüklü parçacıklar dolaylı iyonlaşmaya sebep olur. SRAM belleklerin genel yarı iletken yapısı Şekil 1-1'de görülebilir. Yüklü parçacıkların transistör bağlantılarında sebep olduğu voltaj dengesizlikleri ile bit değeri değişebilmektedir [3].



Şekil 1-1 SRAM temel transistör yapısı

Radyasyon kaynaklı SRAM bellek hataları tek bit değişimi (SEU), tek bit yanması (SEBO), tek bit takılı kalması (SEL) ve tek bit geçici değer (SET) olarak modellenebilir.

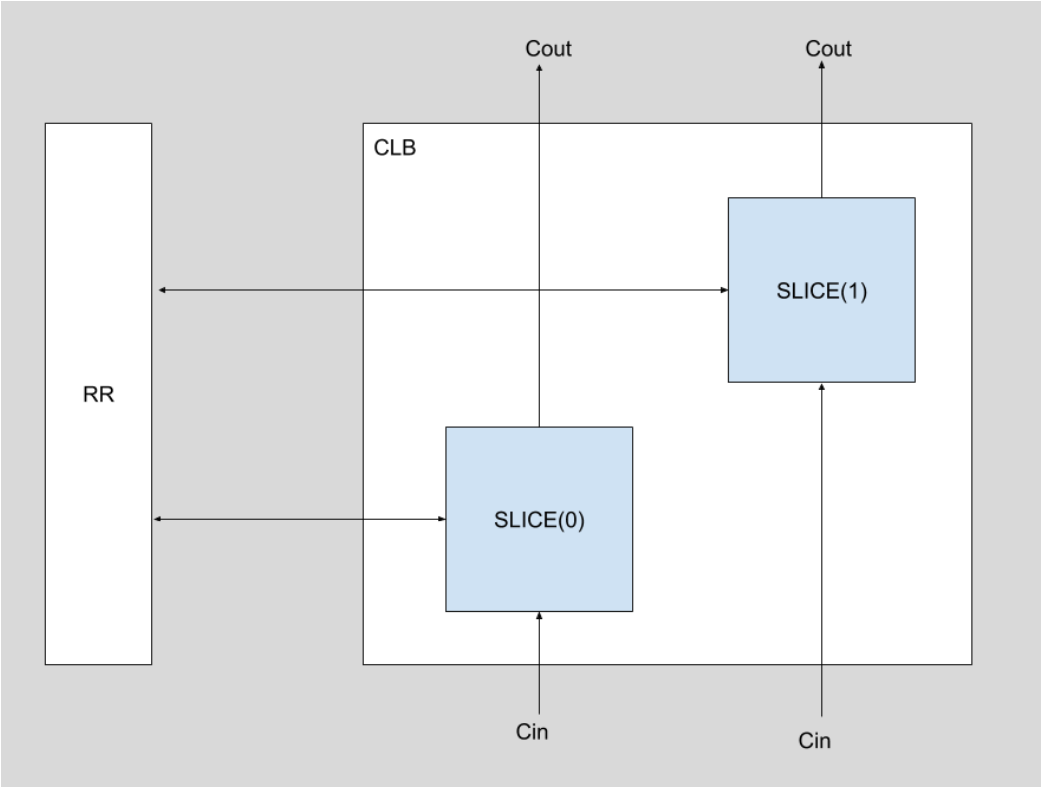
SEL hatası bellek gözesinin bir değere takılıp kalmasıdır. Bu hata türüne karşı bağışıklığı olan FPGA'lar üretilebilmektedir. Tüm Xilinx FPGA'ları üretilirken bu hataya karşı kalifiye edilmektedir. Los Alamos Neutron Science Center (LANSCE)'da yapılan detaylı hızlandırılmış nötron ışıması testleri, Xilinx FPGA'ların bu hataya karşı bağışıklığının olduğunu göstermektedir [4].

Bellek gözesi sinyal seviyesinde meydana gelen anlık değişimlere SET hatası denir. SET hatası oluşan bellek gözesi, kendiliğinden orijinal değerine dönmektedir. FPGA, sinyal hatlarının yüksek kapasitif değerde olması sebebiyle, SET hatalarına karşı dirençlidir [4].

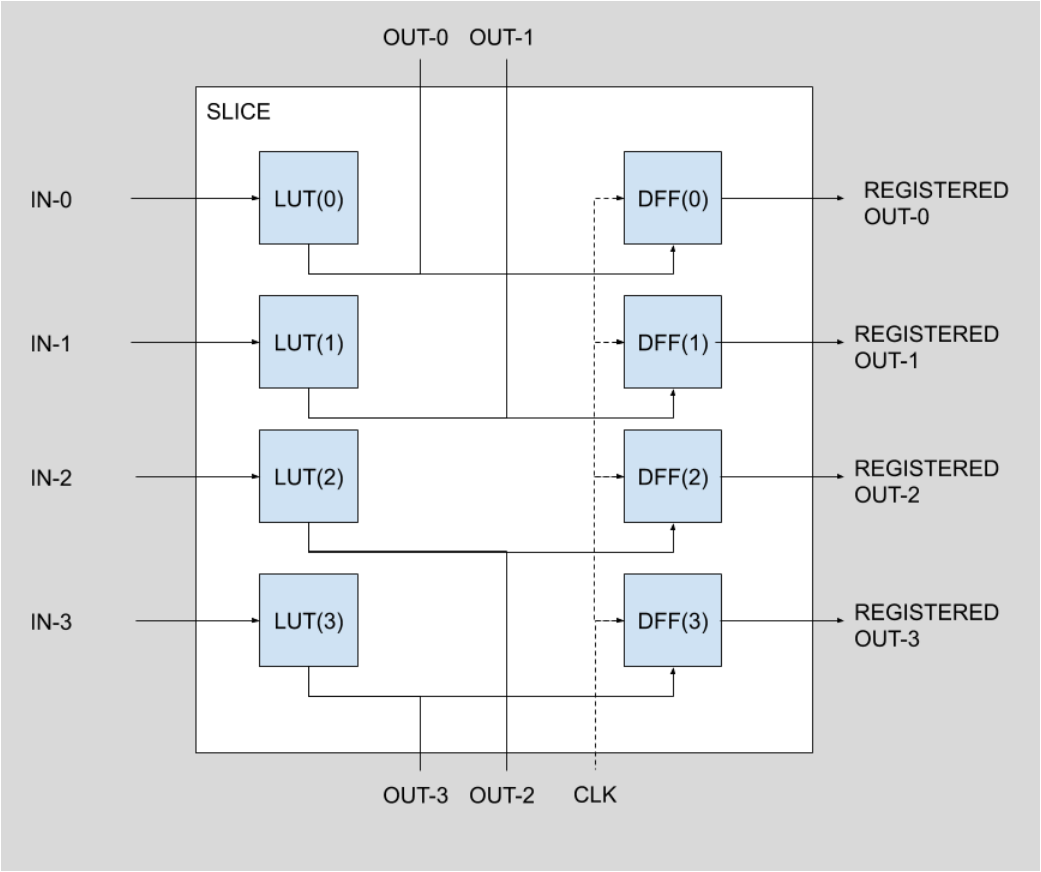
SEBO hataları yarı iletkeni kalıcı olarak etkileyen, düzeltilemeyen hatalardır. Atmosferde radyasyon etkileri kalıcı hatalara sebebiyet verecek seviyelerde değildir [5].

SEU hatası bellek gözesi değerinin tersine dönmesidir ve bellek gözesi tekrardan yazılarak düzeltilebilir. SEU hatası aviyonik sistemlerde kullanılan SRAM tabanlı FPGA'larda en fazla görülen hata çeşididir [3].

FPGA konfigürasyonu SRAM tabanlı FPGA'larda SRAM bellekte tutulur. SRAM tabanlı FPGA aygıtlarında SEU hatası, FPGA konfigürasyonunda değişikliğe sebep olduğundan ciddi problemlere sebep olmaktadır. FPGA'lar temel olarak yapılandırılabilir mantık bloğu (CLB) bloklarından, yönlendirme kaynaklarından (RR) ve giriş çıkış bloğu (IOB) bloklarından oluşan yeniden programlanabilir dizilerdir. CLB'lerin yapısını aramalı tablolar (LUT) ve çift Flip-Flop'lar oluşturur. Bir CLB 2 adet SLICE'den oluşur. Her SLICE'de 4 adet LUT bulunur. LUT altı girişli ve iki çıkışlı bir fonksiyon tablosudur. CLB, SLICE ve LUT sırasıyla Şekil 1-2, Şekil 1-3'de görülebilir. Tasarımdan tasarıma değişen, LUT'lerin ve RR'lerin konfigürasyonları, FPGA konfigürasyon belleğinde tutulur. Konfigürasyon belleğinde konfigürasyon bilgisi harici rasgele erişimli bellek (RAM) verileri de tutulmaktadır [6].



Şekil 1-2 CLB yapısı



Şekil 1-3 SLICE yapısı

CLB'lerin SEU etkisi ile değişmesi, FPGA'nın yapısal bütünlüğünü bozup, gerçekleştirdiği fonksiyonu değiştirmektedir. BLOK RAM kaynaklı hatalar ise yapısal bütünlüğü bozmadan sadece işlevsel verileri bozarak FPGA'nın yanlış çalışmasına sebep olmaktadır [3].

Yapılan çeşitli deneylerde, SEU kaynaklı hataların, sadece uzayda değil, atmosferde de FPGA'ların düzgün çalışmasını etkilediği görülmüştür [4, 7, 8, 9].

Yüksek güvenilirlik gerektiren aviyonik uygulamalarında SEU kaynaklı problemlerin çözümü, hata toleranslı sistem mimarilerini gerekli kılmaktadır. SEU kaynaklı hatalar kalıcı değildir, FPGA'nın tekrar programlanması ile çözülebilmektedir. Xilinx Virtex-6 SRAM tabanlı FPGA'ların tekrar programlanması, kısmi dinamik donanım şekillendirme (PDR) veya bütün şekillendirme olmak üzere iki farklı biçimde yapılabilir. Bütün şekillendirme işlemi esnasında FPGA çalışmaz iken, parçalı yazma işlemi esnasında yazılan blok harici FPGA fonksiyonları çalışmaya devam edebilir. İlk açılışta FPGA'nın bir kere bütün olarak şekillendirilmesi gerekmektedir [10, 11, 12].

SEU kaynaklı hataların maskelenmesinde üçlü modül yedekli (TMR) metodu yaygın olarak kullanılmıştır. TMR metodu, giriş ve çıkışları aynı olan ve aynı fonksiyonu gerçekleştiren birbiriyle aynı üç yedek modülün çıkışlarının bir oylayıcı vasıtasıyla karşılaştırılarak seçilmesi ile kullanılmasına verilen isimdir. Modüllerin birinde hata olması durumunda diğer iki modülün çıkışları birbirine eşit olacağından, oylayıcı hatalı sonucu seçmeyip doğru sonucu seçerek çıkışa hatalı sonucun gitmesini engeller. Hatanın birden fazla modülde meydana geldiği durumlarda TMR sisteminin doğru sonuç vereceği garanti edilemez.

Literatürdeki birçok çalışmada [3, 13, 14, 15, 16, 17, 18] olduğu gibi, TMR metodu PDR ile birlikte kullanılarak hata toleranslı bir mimari kurulabilir. Hata, yedekli modüllerin çıkışlarındaki farklılık gözlemlenerek bulunabilir ve hatalı görülen modül PDR ile yeniden yazılarak hata giderilebilir. Fakat hatanın TMR yapı çıkışı gözlemlenerek bulunması, sadece hatalı çerçevenin değil, genellikle birden fazla çerçeveden oluşan tüm yedek modül baştan şekillendirilmesini gerektirdiği için, hata düzeltme süresini artırmaktadır [3, 13, 14, 15, 16, 17, 18].

FPGA silikon mimarisinde CLB seviyesinde yedekleme sağlanarak ve CLB'lere hata bulma ve düzeltme gibi farklı görevler atanarak sağlanabilir [19, 20].

Fakat tasarıma CLB seviyesinde yaklaşım, tasarım kompleksliğini, tasarlama süresini ve maliyetini artırdığından binlerce CLB'den oluşan aviyonik sistemlerde ölçeklendirme problemi yaratmaktadır. Ayrıca CLB seviyesinde yedekleme sağlamak için kullanılan CLB sayısı, hata toleranslı olmayan tasarımın 7 katına kadar çıkabilmektedir [19].

Dahili test (BIST) kullanılarak hata bulunan ve PDR kullanılarak hatanın düzeltildiği çalışmalar da bulunmaktadır [21, 22, 23].

BIST yapıları aviyonik sistemlerde donanımın durumunu anlık olarak raporlayabildiği için yaygın olarak kullanılmaktadır. Fakat hata toleranslı bir tasarım mimarisi için her fonksiyonun doğruluğunun kontrol edilmesi gerekir. Her fonksiyonun doğruluğunu test eden BIST yapılarının kurulması tasarım kompleksliğini, tasarlama süresini ve maliyetini artırır. Bu yüzden sadece BIST kullanılarak hata bulan toleranslı FPGA tasarımlarının aviyonik sistemlerde kullanılabilirliği düşüktür. Ayrıca gerçekleşen her hata sistemin çalışmasını etkilemez [24].

FPGA yapısal bütünlüğü, konfigürasyon hafızası periyodik olarak yazılarak, hatanın varlığına bakılmaksızın düzeltilebilir [18].

Konfigürasyon hafızasının periyodik olarak yazıldığı metoda veri sürtmesi adı verilmektedir. Veri sürtmesi ile FPGA yapısal bütünlüğü garanti edilebilir. Fakat FPGA'nın durum bütünlüğü garanti edilemez.

Konfigürasyon belleğinde oluşan hatalar, konfigürasyon hafızası tekrardan okunarak gözlemlenebilir. Bulunan hata sadece bozulan bitin bulunduğu çerçeve tekrardan yazılarak düzeltilebilir. Konfigürasyon belleğinin minimum miktarda okuma ve yazma işlemi yapılabilen, 81 adet 32-bitlik kısımlarına çerçeve adı verilmektedir [24]. Sadece konfigürasyon hafızasındaki hatanın düzeltilmesi

FPGA'nın düzgün çalıştığını garantilemez. Gerçekleşen hata, başka modüllerin konfigürasyon yapısını bozmadan yanlış bir duruma geçmesine sebep olabilir. Hata toleransının sağlanabilmesi için durum bütünlüğünün de sağlanması gerekmektedir.

Bu tezde literatürde yer alan çalışmaların eksikleri incelenmiş ve çözüm önerileri sunulmuştur. Hata toleranslı FPGA tasarımında yapısal bütünlüğün yanı sıra durum bütünlüğü de incelenmiştir. Yapısal hatanın bulunma ve düzeltilme sürelerinin kısaltılması hedeflenmiştir. Hata durumunda hatanın FPGA fonksiyonunu etkileme ihtimali minimize edilmeye çalışılmıştır. Teknolojinin hatanın maskelenmesi konusu farklı sistem mimarileri içerisinde incelenmiştir. Kurulan mimarinin hata olasılığı hesaplanmış ve aviyonik sistemlerde en çok kullanılan, kritik verilerin taşındığı iletişim protokolü ARINC-429 üzerinde test edilmiştir. Önerilen tasarım, hata tolerans ihtimali, FPGA alanı, hata düzeltilme süresi başlıkları altında yorumlanmıştır.

Tezin ikinci bölümünde hata toleranslı FPGA tasarımları ile ilgili çalışmalar daha detaylı bir şekilde incelenmiştir. Üçüncü bölümde önerilen hata toleranslı FPGA mimarisi anlatılmıştır. Hatanın maskelenmesi, hatanın tespiti ve hatanın düzeltilmesi konuları işlenmiştir. Tezin dördüncü bölümünde FPGA tasarımının kodlanması detaylı olarak anlatılmıştır. Beşinci bölümde tasarımın test edildiği ortam ve test sonuçları aktarılmış ve performans ve güvenilirlik açısından incelenmiştir. Son olarak altıncı bölümde tez yorumlanmıştır.

2. LİTERATÜR ÖZETİ

Hata toleranslı FPGA tasarımları, hataya yaklaşım ölçeği, hatanın bulunma yöntemi, hata düzeltme işleminin ölçeği, hata düzeltme işleminin kaynağı, yapısal hataların ve durumsal hataların çözümü gibi farklı özellikleri ile incelenebilir. Tezin bu bölümünde literatürde yer alan çalışmalar çeşitli açılardan incelenmiş ve eksikleri tartışılmıştır. İncelenen çalışmalar temel alınarak, hataya yaklaşım ölçeği, hatanın bulunma yöntemi, hata düzeltme işleminin ölçeği, hata düzeltme işleminin kaynağı, yapısal hataların ve durumsal hataların çözümü gibi farklı başlıkların hata toleranslı yapıya etkisi incelenmiştir.

Xilinx Virtex-6 SRAM tabanlı FPGA konfigürasyon bellek içeriği dinamik mantık devreleri ve statik mantık devreleri olmak üzere iki kategoriye ayrılır. Parçalı şekillendirme işlemi, devrenin fonksiyonu değiştirilecekse, yalnızca dinamik mantık devrelerine uygulanabilir, statik mantık devrelerine parçalı şekillendirme yapılamaz. Blokların statik ve dinamik olarak ayrılması, Xilinx FPGA tasarımı, yazılım akışında belirlenir. PLL ve diğer saat değiştirme yapıtaşları, giriş çıkış yapıtaşları, seri alıcı verici blokları dinamik bölgede yer alamaz. Dinamik ve statik bölge arasında çift yönlü sinyaller kullanılamaz. Ayrıca dinamik bloklar tekrardan yazıldıklarında düzgün çalışabilmesi için yeniden başlatma mekanizmasına sahip olmalıdır [10].

Xilinx Virtex-6 FPGA'larda bütün olarak yeniden yazma işlemi sadece FPGA dışı bir kaynakla yapılabilmekte iken, parçalı şekillendirme işlemi hem FPGA dışı bir kaynak tarafından hem de ICAP arayüzü ile FPGA tasarımının kendisi tarafından yapılabilmektedir. Dahili konfigürasyon erişim portu (ICAP) arayüzü okuma ve yazma kanalları 8-bit, 16-bit ve 32-bit olarak seçilebilir. ICAP arayüzü maksimum çalışma frekansı 100MHz'dir [6].

Xilinx Virtex-6 FPGA tasarımı, ISE Design Suite 12.1 ve daha yeni versiyonlarında yapılabilmektedir. Tasarım konsept olarak dinamik ve statik olarak ayrıldıktan sonra dinamik modüllerin ve tüm tasarımın sentezi ayrı ayrı yapılır. Sentez sonrası oluşan NETLIST dosyaları kullanılarak, Plan Ahead

programı aracılığıyla dinamik modüllerin FPGA yerleşimi seçilir. Statik mantık devreleri için yerleşim ve optimizasyon işlemi Plan Ahead tarafından otomatik olarak yapılabilmektedir [11, 12].

CLB seviyesinde SEU tespiti ve hata düzeltimi yapan birçok çalışma vardır. Bu çalışmalardan birinde [19] CLB içerisinde bulunan LUT'lere farklı roller verilerek SEU kaynaklı hata toleranslı FPGA tasarımı yapılmıştır. Hata tespiti için BIST kullanılmış ve hata PDR ile FPGA dışı bir kaynak tarafından düzeltilmiştir. Bu tasarım sadece içerisinde en az 8 LUT bulunan CLB'lere uygulanabilir. Bu roller şunlardır:

1- Orijinal Fonksiyon
   Orijinal fonksiyon rolünün atandığı LUT'dir. CLB'nin yapması hedeflenen asıl işi yerine getirir.
2- Yedek Fonksiyon
   Yedek fonksiyon görevinin atandığı LUT'dir. Orijinal fonksiyon LUT'sinin birebir aynısıdır. Orijinal fonksiyon LUT'sinde bir hata olması durumunda onun yerini alarak hata toleransı sağlanır.
3- Orijinal Fonksiyon Hata Tespiti Kodu
   Orijinal fonksiyon LUT'sinin önceden hesaplanan hata tespit kodunu bulundurur.
4- Yedek Fonksiyon Hata Tespiti Kodu
   Yedek fonksiyon LUT'sinin önceden hesaplanan hata tespit kodunu bulundurur.
5- Hata Denetçisi
   Hata denetçisi LUT'si, fonksiyon LUT'lerinin değerleri ile hata tespit kodu LUT'lerinin değerlerini karşılaştırarak hatayı tespit eder. Hata hem orijinal LUT'de hem de yedek LUT'de bulunuyorsa kalıcı hata olarak sınıflandırır.
6- Fonksiyon Yol Atama
   Fonksiyonlarda meydana gelen hataya göre orijinal fonksiyon ile yedek fonksiyon arasında geçiş yapılmasını sağlar.
7- Kalıcı Hata Raporlama
   Kalıcı hatanın tespit edildiğini raporlayan LUT'dir.
8- Serbest Yol Atayıcı

CLB'de veya komşu CLB'de meydana gelen kalıcı hata sinyaline yön atayan LUT'dir.

Tasarım orijinal fonksiyonda meydana gelebilecek hatayı tespit edip düzeltebilmektedir. Fakat bu tasarımın istenilen toleranslı yapıda olabilmesi için Hata Denetçisi, Fonksiyon Yol Atama, Kalıcı Hata Raporlama ve Serbest Yol Atayıcı LUT'lerinin CMOS seviyesinde radyasyona karşı dayanıklı olması gerektiği belirtilmektedir. Tasarımda CLB'de meydana gelen kalıcı hataların çözümü içinde CLB seviyesinde yedekli yapı önerilmiştir. Kalıcı hatanın bulunması durumunda CLB'nin yerini önceden belirlenen başka bir CLB almaktadır. Bu taşıma işlemi ise PDR kullanılarak yapılmaktadır. PDR işlemi tek bir modül tarafından FGPA dışına UART aracılığıyla mesaj gönderilerek başlatılmaktadır. FPGA dışındaki kaynak bu UART mesajının içeriğine göre hatalı CLB'yi yedek olarak ayrılan CLB üzerine PDR ile taşımaktadır. Bu işlemin 1-2 saniye sürdüğü belirtilmektedir [19].

Literatürde yer alan başka bir CLB seviyesi tasarımda [20], CLB-M ve CLB-S olarak adlandırılan yapıtaşları kullanılmıştır ve tasarım 4 CLB-S'den ve 1 CLB-M'den oluşan yapılara bölünmüştür. CLB-M'ler 3 adet, CLB-S'ler 1 adet CLB'den oluşmaktadır. CLB-M'e komşu olan CLB-S'lerdeki hata CLB-M içerisindeki 3 CLB'nin çıkışları karşılaştırılarak bulunup, hatalı CLB-S, CLB-M'de bulunan CLB'lerden birine PDR metodu kullanılarak taşınabileceği önerilmiştir. Hata bulma metodu olarak TMR yapı çıkışları kullanılmış ve hatanın PDR ile FPGA dışı bir kaynak tarafından yeniden yazılarak çözülebileceği belirtilmiştir [20].

Sentez için kullanılan programlar, bir donanım tasarım dili (HDL) kullanılarak tasarlanan kompleks bir fonksiyonu LUT, RR, DFF ve BLOK RAM seviyesinde bileşenlerine bölerek FPGA üzerinde gerçekler. Böyle bir fonksiyonun tümevarım yöntemi ile LUT, RR, DFF ve BLOK RAM seviyesinde tasarım ve yerleşimi az bileşenli sistemlerde mümkün olmakla birlikte binlerce temel bileşenden oluşan aviyonik sistemlerde ölçeklendirme problemine sebep olmaktadır. Hata toleranslı yapıya CLB seviyesinde yaklaşılarak yapılan tasarımlar [19, 20], daha yüksek çözünürlükteki yaklaşımlara göre, daha fazla FPGA alanı kaplamaktadır. Daha yüksek çözünürlükte yaklaşımlarda yol atama görevini üstlenen blokların kapladıkları alan orijinal fonksiyona göre çok daha az alan kaplayabilmektedir [3].

CLB seviyesinde hata tespiti yapıldığında düzeltme işlemi için konfigürasyon hafızası en az yazma boyutu olan 20 CLB'nin tekrardan yazılması gerekmektedir. CLB seviyesi yedekleme ve düzeltme işleminin kapladığı FPGA alanı açısından dezavantajları bulunmakta olup, düzeltme süresi açısından bir artısı bulunmamaktadır. Önerilen tasarımlarda [19, 20], düzeltme işlemi için FPGA dışı bir kaynağa ihtiyaç duymaktadır. 1-2 saniye olarak belirtilen düzeltme süresi [19] bütün FPGA'nın yeniden şekillendirmesinden daha fazla zaman almaktadır.

Farklı çözünürlükte hata toleranslı TMR yaklaşımlar literatürdeki çalışmada [3], karşılaştırılmıştır. SEU hatası yedek modüllerin çıkışları karşılaştırılarak oylayıcı ile bulunmuş ve hatalı görülen modül FPGA dışı bir kaynak tarafından PDR ile tekrardan yazılmıştır. Ayrıca kalıcı hataların tespiti için tekrardan yazılan modülde hatanın devam edip etmediği kontrol edilmiştir. Sistemin çözünürlüğünün fazla olduğu ve az olduğu yaklaşımlar karşılaştırılmıştır. Yedeklenen modüllerin boyutu küçüldükçe hata düzeltme işleminin hızlanacağı, fakat oylayıcı sayısı artacağı için kullanılan FPGA alanın artacağı belirtilmiştir. Önerilen yapı 10 basamaklı FIR filtre üzerinde denenmiş ve farklı çözünürlükteki yaklaşımlar için sonuçlar verilmiştir. Düşük çözünürlükteki yaklaşımın incelenmesi için 10 basamaklı FIR filtre modülü bütün olarak yedeklenmiştir. Birbiriyle aynı olan 3 modül, bağımsız, tekrardan yazılabilir dinamik alanlara yerleştirilmiş ayrıca oylayıcıda farklı bir alana koyulmuştur. Yüksek çözünürlükteki yaklaşımın incelenmesi için FIR filtrenin her bir basamağını yerine getiren modüller yedeklenmiştir. Her bir basamak için birbiriyle aynı üç modül kullanılmış ve modüllerin çıkışı oylayıcıdan geçirilerek bir sonraki basamağı oluşturan yedeklenmiş modüllere iletilmiştir. Farklı basamakları oluşturan her bir modül ve yedekleri birbirinden farklı tekrardan yazılabilir alanlara yerleştirilmiştir. Yüksek çözünürlüklü TMR yapının, tek bir FIR filtre modülünden 5 kat daha fazla alana ihtiyaç duyduğu gözlemlenmiştir. Düşük çözünürlüklü yapıda ise 4 kat daha fazla alan kaplamıştır. Tüm FPGA'nın tekrardan yazılması için gereken sürenin 364ms olduğu sistemde, yüksek çözünürlüklü, küçük modüllü yaklaşımda bazı basamaklarda kullanılan modüllerin tekrardan yazılması için gereken sürenin 40ms kadar yüksek olabilirken daha küçük olan diğer basamaklarda bu sürenin 17ms gibi rakamlara kadar düştüğü belirtilmiştir. Düşük çözünürlüklü, tüm filtre modülünün yedeklendiği yapıda ise yeniden yazma süresinin 66ms olduğu belirtilmiştir [3].

Karşılaştırma sonucunda, yüksek çözünürlüklü yaklaşımlarda düzeltme süresi kısaldığı, fakat oylayıcı görevi gören blokların kapladığı alanın arttığı açıkça görülmektedir. Bu çalışmada [3], düzeltme işlemi için FPGA dışı kaynak kullanılmıştır. FPGA içi konfigürasyon arayüzü bant genişliği, FPGA dışı konfigürasyon arayüzü bant genişliğine göre çok daha fazladır. PDR için FPGA dışı kaynak kullanımı, hata düzeltme süresinin, uzamasına sebep olmaktadır. Hata süresinin uzamasına sebep olan bir diğer sebep ise, hata tespiti için TMR çıkışlarının gözlenmesidir. Önerilen yapıda [3], oylayıcı çıkışlarının karşılaştırılması ile yapılan hata tespiti ile ancak modül seviyesinde hata bulunabilmekte olup, hatanın gerçekleştiği konfigürasyon çerçevesinin bulunması mümkün değildir. Çalışmada [3], TMR ile düzeltilen blokların senkronizasyonu ve durum bütünlüğü konularına değinilmemiş, sadece yapısal bütünlük konusu işlenmiştir.

Hata toleranslı yapının, sadece mimarinin doğruluğu konusunda değil, kodun o anda bulunduğu durumun doğruluğunu da sağlayacak şekilde tasarlanan çalışmada [17], üçlü yedekli yapı kullanılmıştır. Üç yedeğin bulunduğu konfigürasyon bellek alanı okunup karşılaştırılarak hata durumu aranmıştır. Bunu için yedekler sadece fonksiyonel yapı olarak aynı değil CLB ve RR yerleşimleri ile de aynı olması gerekmektedir. Sentez programları böyle bir özellik sunmamaktadır. Üç modülün kodları itibari ile birbirinin aynı olsa bile yerleşimleri birbirinden farklı olacaktır. Bir modülün bit akış verisini alıp sentezdeki yerleşiminden ayrı boş bir konfigürasyon bellek alanına yazılması o modülün çalışmasını sağlamaz. Konfigürasyon belleği modül içeriği sentez sonrası yerleşimine göre değişir. Çalışmada üç yedek modülün bit akış verisinin birbiriyle tamamen aynı olabilmesi için yön verme kısıtlaması üreten bir program kullanılmış, yerleşim kısıtlamaları el ile yazılmıştır. Bu kısıtlamalar statik hatların modüllerde aynı olmasını sağlayamaz. Bu sebeple statik hatların yeniden yazılabilir modüllerden hiç geçmemesi gerekmektedir. Statik hatların kısıtlanması Xilinx Tasarım Dili kullanılarak sağlanmıştır. Bu kısıtlamalardan yönlendirici ile ilgili olanlar sentez programının anlayacağı fakat okunamayan formattadır. Birinci modülün yerleşimi için üretilen yönlendirme kısıtlamaları ikinci ve üçüncü yedek içinde kullanılmış sadece net isimleri değiştirilmiştir. Bunun sağlanabilmesi için CLB seviyesi yaklaşım gerekmektedir. Birbiriyle tamamen aynı üretilen bu

yedekler konfigürasyon hafızası okunup karşılaştırılmış hatalı bulunan modülün çerçevesi yerine doğru çalışan modül çerçevesi yazılmıştır. Konfigürasyon belleği okuma ve yazma işlemi FPGA dışı bir kaynak ile yapılmıştır. Bu hata toleranslı yapı ile sadece fonksiyonel mimari değil, durum bilgisi de bir modülden diğerine aktarılmıştır [17]. Çalışmada, durum bütünlüğünün sağlanabilmesi için birbiri ile yerleşim kaynakları açısından tamamen aynı 3 modül üretilmesi gerekmektedir. Bunun sağlanabilmesi için modülün tasarımı sırasında CLB seviyesi yaklaşım gerekmemekte iken, modülün sentez sonrası yerleşimi için CLB seviyesi kontrol gerekmektedir. Sentez programları sentezlenen kodu temel bileşenlerine böldükten sonra, sezgisel yaklaşımlar ile FPGA yerleşimlerini yapmaktadır. Bu yerleşimler her sentezde farklı sonuç vermektedir. Yerleşimlerin modüllerde sabitlenebilmesi için, sentez programına CLB seviyesinde yerleşim kısıtlamaları girilmesi gerekmektedir. Yerleşim kısıtlarının, el ile girilmesini gerektiren bir metot, az bileşenli sistemlerde mümkün olmakla birlikte binlerce temel bileşenden oluşan aviyonik sistemlerde ölçeklendirme problemine sebep olmaktadır. Tasarımda [17], hata tespiti için birbiri ile aynı olan 3 modülün ilgili konfigürasyon çerçeveleri karşılaştırılmıştır. Çalışmada [17], FPGA dışı konfigürasyon arayüzü kullanılmıştır. Yapısal ve durum bütünlüğünün korunması için seçilen metodun kullanılabileceği sistem özellikleri ölçeklendirme problemleri nedeni ile kısıtlıdır.

Spesifik bir bileşen için, hata toleranslı yapının hedeflendiği çalışmalar da literatürde yer almaktadır. Bu çalışmalardan birinde [23], FPGA'da tasarlanan mikro işlemcinin hata toleranslı olması hedeflenmiştir. SEU kaynaklı hatanın bulunması için mikro işlemci operasyonunun her evresine BIST yapılar kurulmuştur. Hata bulunduktan sonra işlemci operasyonu düzeltilene kadar durdurulmuştur. Önerilen yapı değiştirilebilir fonksiyonel birimler, yedek fonksiyonel birimler, yeniden yazma kontrolcüsünden oluşmaktadır. Yeniden yazma kontrolcüsü BIST sonucuna göre ilgili değiştirilebilir fonksiyonel birim yerine karşılık gelen yedek fonksiyonel birimin kullanılmasına karar vererek hata toleransını sağlamaktadır. Yedek fonksiyonel birimlerin yazılım ile ve donanım ile gerçekleştirilebileceği belirtilmiş ve birimin kritikliğine göre bu ayrım yapılmıştır. Kritik birimlerin donanım ile yedeklenmesi önerilmiştir. Yedek fonksiyonel birimler için ortak bir dinamik alan ayrılması ve hatalı modülün oraya hata durumunda yazılması önerilmiştir. Bu yapının tasarımı okuyucuya bırakılmıştır. Hata

düzeltildikten sonra yazma kontrolcüsü durdurulan işlemciyi yeniden çalıştırmıştır. Yeniden yazma işlemi FPGA içinden ICAP arayüzü ile yapılmıştır [23]. Kritik bileşenler için ayrılan boş alana, hata durumunda, hatalı bileşenin yeniden yazılabilmesi için, tasarım sırasında bahsi boş alanın her bir kritik birim için tek tek sentezlenip yerleşim yapılması gerekmektedir. Bu yöntem, az sayıda modül için mümkün olmakla birlikte modül sayısının artması ile ölçeklendirme problemi oluşmaktadır. Dinamik olarak ayrılan bir alan ile statik alan arasında bulunan giriş çıkış sinyallerinin birbirinin aynı olması gerekmektedir. Her modülün, aynı alana yazılacak diğer modüllerin giriş çıkış sinyallerini kullanılmasa bile bulunduracak bir tasarım yapılması ile önerilen yapı kurulabilir. Önerilen tasarımda [23], kullanılan hata bulma metodu olan BIST, her bir fonksiyonun doğru çalıştığını kontrol eden devreler kurulması ile mümkündür. Fakat çok bileşenli ve çok fonksiyonlu yapılarda BIST devrelerinin tasarımı ve ortak dinamik alan seçimi, ölçeklendirme problemi oluşturmaktadır. Bir işlemcinin her bir alt fonksiyonu için kurulması gereken BIST devreleri, kaplayacağı FPGA alanı açısından da dezavantajlıdır. Çalışmada [23], yeniden yazma arayüzü olarak FPGA içi ICAP kullanılmış olup, dışarıdan bir kaynağa ihtiyaç duyulmasına gerek kalmamıştır. Fakat ICAP arayüzünü kontrol eden modülde bir hata gerçekleşmesi durumunda, düzgün çalışan FPGA kodunun, yeniden yazılarak bozulabilmesi riski bulunmaktadır. Bu duruma karşı bir önlem önerilmemiştir.

Diğer bir hata toleranslı mikro işlemci tasarımında [16], sistemin hata toleranslı olması için TMR ve PDR kullanılmıştır. TMR Microblaze işlemci için kullanılmış ve Microblaze işlemci yedeklenmiştir. Microblaze işlemci çıkışları yedekleri ile karşılaştırılıp hata bulunmuştur. Hatanın bulunduğu durumda işlemci operasyonu durdurulmuştur ve hatalı modül PDR ile yeniden yazılmıştır. Microblaze işlemcinin kullandığı BLOK RAM'ler için TMR kullanılmamıştır. BLOK RAM hata toleransı için hata düzeltme kodu (ECC) kullanılması önerilmiştir. Yeniden yazılan işlemcinin yedekleri ile senkronizasyonu BLOK RAM ve yazmaç içerikleri kopyalanarak sağlanmıştır. Fakat bunun için yedek işlemcilerin durması gerekmiştir. Önerilen yapı hata toleranslı olmayan yapıya göre 4 kat daha fazla FPGA alanı kullanmıştır. Ayrıca işlemcinin çalışma hızı %30 yavaşlamıştır. Önerilen yapıda yeniden yazma işlemi için FPGA dışı kaynağa ihtiyaç duyulmuştur. Yeniden yazılan işlemcinin 6µs sonra senkronizasyonunun

tamamlandığı ölçülmüştür. Önerilen sistemin matematiksel olarak hata analizi yapılmış ve hatalar arası ortalama zaman (MTBF) değerinin hata toleranslı olmayan sisteme göre 2,713 kat daha fazla olduğu hesaplanmıştır. Hata toleranslı olmayan yapıda bir modülün bit akış veri boyutu 1.143.569 bit iken ECC kullanılan yapıda 1.262.663 bit olduğu görülmüştür. Yeniden yazma hız maksimum 24Mbit olarak verilmiştir. Bu verilen değerler ile Microblaze işlemcinin yeniden yazılma süresi yaklaşık olarak 52μs'dir. Bu ölçümler Virtex-4 FPGA üzerinde yapılmıştır [16]. Microblaze işlemcinin hata toleranslı tasarlandığı yapıda [16], hata TMR kullanılarak çıkışa etkisi maskelenmiştir. Düşük çözünürlükte, işlemci seviyesinde yedekli yapı kurularak statik bölgede yer alacak olan oylayıcı FPGA alanı azaltılmıştır. Fakat hatanın TMR ile bulunması tüm modülün tekrardan yazılmasını gerekli kıldığından, hata düzeltme süresi, sadece hatalı çerçevenin düzeltildiği tasarım mimarisine göre 65 kat fazladır. Ayrıca hata düzeltme metodu olarak FPGA dışı kaynağın seçilmesi, hata düzeltme süresinin artmasının bir diğer sebebidir. BLOK RAM'lerde gerçekleşebilecek SEU kaynaklı hataların çözümü için ECC kullanılmıştır. Yeniden yazılan işlemcinin çalışan işlemcilere senkronizasyonu konularına değinilmiştir. Ayrıca önerilen mimari için yapılan MTBF analizinde, metot uygulandığı taktirde hata ihtimalinin ciddi miktarda azaldığı görülmüştür.

Yonga üzeri sistemin (SOC) hata toleranslı olmasının hedeflendiği çalışmalar da bulunmaktadır [15]. Hata toleranslı yapıda, yazılım ve donanıma çeşitli görevler verilmiştir. Önerilen yapıda TMR ve PDR yazılım kontrolünde gerçekleştirilmiştir. TMR yapının kurulması görevi yazılıma atanmıştır. Böyle bir görevin yerine getirilebilmesi için yeniden yazılabilen modüller için bir kütüphane oluşturulmuştur. Oluşturulan kütüphanedeki her bir fonksiyon için bir adet modül tasarlanmıştır. Bu modülün FPGA'nın neresine yazılırsa yazılsın aynı görevi yerine getireceği söylenmiştir. Böyle bir modül tasarlandığı varsayımından sonra, yazılım ilgili fonksiyon kullanılacağı zaman o modülü yedekleri ile dinamik ayrılan alanda TMR yapı kurulması önerilmiştir. Tamamen yazılım kontrolünde esnek bir yapı kurulmaya çalışılmıştır. Kurulan TMR yapıda oylayıcı çıkışında hata görülmesi durumunda hatalı modül kütüphaneden yeniden yüklenerek hatanın giderilebileceği belirtilmiştir. TMR için ayrılan, yeniden yazılabilir bölge farklı modüller için kullanılabilmekte ve TMR yapı kurulması operasyonu farklı modüller

için sırayla yapılıp hata aranması operasyonu sırayla gerçekleştirilmektedir. TMR yapı her zaman doğru sonuç almak için kurulmamış olup sadece hatanın bulunması için kullanılmıştır. Böyle bir yapının her bir fonksiyon için her zaman yedeği ile çalışan yapıya göre avantajı kapladığı FPGA alanıdır. Tamamen yedekli bir TMR yapıda 3 kattan fazla alan harcanırken bu yapıda harcanmamıştır. TMR için ayrılan alan tüm modüllerde ortak kullanılmıştır. Fakat bu yapı ile her zaman doğru sonuç hedeflenmemiş sadece yapısal bütünlüğün korunması hedeflenmiştir [15]. Tasarımda, TMR hata toleransı sağlamak için değil hatanın bulunması için kullanılmıştır. TMR, konfigürasyon belleğinde oluşan hatanın, modülün yerine getirdiği fonksiyonu etkilememesi, diğer bir deyişle hatanın maskelenmesi için etkili bir yöntemdir. Fakat hatanın tespitini düzeltilebilecek en küçük yazma miktarı olan konfigürasyon çerçevesi ölçeğinde yapamadığından hata tespiti için iyi bir metot değildir. Sistemde TMR hata tespiti için tek bir alan ayrılarak TMR yapının FPGA alanı açısından fazla yer kaplamaması hedeflenmiştir. Fakat bahsi geçen yapıda, önceden ayrılan alanda kontrol edilecek olan modül sayısı arttıkça ölçeklendirme problemi oluşmaktadır. Bir alana birden fazla modülün yerleştirilebileceği bir tasarımın yapılabilmesi için, yerleştirilecek her modülün diğer modüllerin giriş çıkış sinyallerini kullanılmasa bile bulundurması gerekmektedir. Bu yapı birbirinden farklı modül sayısı arttıkça ölçeklendirme problemi doğurmaktadır. Ayrıca yapıda yazılım kontrolüne verilen düzeltme işlemi bant genişliği ICAP'e göre düşük olan FPGA dışı bir kaynak ile gerçekleştiğinden hata düzeltme süresi uzamaktadır. Bu yapı ayrıca yeniden yazılabilir modüllerin saklandığı bir belleğe ihtiyaç duymaktadır.

PDR kullanılan hata toleranslı sistemler için, hata bulma modellerinin incelendiği çalışmada [22], TMR, dahili testli ikili modül yedekleme (BIST DMR), ikili modül yedekleme (DMR) ve BIST yöntemleri karşılaştırılmıştır. Konfigürasyon hafızasının okunarak hata tespiti yapıldığı modüller, konfigürasyon hafızası kaynaklı olmayan hataları bulamayacağından kullanılmamıştır. Sadece SEU kaynaklı değil başka sebeplerden kaynaklanabilecek hatalara da tolerans sağlanmaya çalışılmıştır. Bu hataların kaynağı belirtilmemiştir. BIST yapılı yedeksiz modüller ile hata tespiti, modül içerisindeki her bir fonksiyon için BIST yapısı kurularak hatanın bulunabildiği yapılardır. Hata bulunarak düzeltilebilir fakat hata olduğu esnada düzeltilene kadar sistemin doğru sonuç vermesi

sağlanamaz. DMR yapı ile hata tespiti, birbiri ile aynı fonksiyonu gerçekleştiren iki modülün sonucu karşılaştırılarak hatanın tespit edildiği yapılardır. Hatanın varlığı tespit edilebilir fakat hatanın hangi modülde olduğu tespit edilemez. Hata durumunda iki modülün birden yeniden yazılması gerekmektedir. Hata gerçekleşmesi durumunda, iki modül yeniden yazılana kadar sistemin doğru çalışması sağlanamaz. DMR ve BIST kullanılan modüller ile hata tespiti, hatanın birbiriyle aynı fonksiyonu yerine getiren iki modülün çıkışları gözlemlenerek ve modüllerin BIST sonuçları gözlemlenerek tespit edildiği yapılardır. Hatanın hangi modülde gerçekleştiği bulunabilir ve sadece hatalı modül yeniden yazılarak yapısal bütünlük korunabilir. Hata gerçekleştiğinde doğru çalışan modülün sonucu çıkışa verilerek hata giderilebilir ve sistemin düzgün çalışması sağlanabilir. TMR ile hata tespiti, birbiri ile aynı işi yapan üç modülün çıkışları karşılaştırılarak hatanın tespitinin yapıldığı yapıdır. Hatanın hangi modülde olduğu bulunabilir ve BIST yapılara ihtiyaç duyulmaz. Hata sadece hatalı modül düzeltilerek tolerans sağlanabilir. Hata sistemin çıkışını etkilemez. Karşılaştırılan bu dört yapının kapladıkları alanlar incelenmiştir. Sayaç ve dekoder gibi basit bir fonksiyonu gerçekleştiren bir modülün TMR, BIST DMR, DMR ve BIST yapılarında, Virtex-5 üzerinde kapladıkları alanlar sırasıyla 145 SLICE, 91 SLICE, 86 SLICE ve 44 SLICE olarak verilmiştir. Yedekli modül kullanılan yapılarda, hatanın çıkışta görülmeme ihtimali, SEU sayısına ve yedek modül sayısına göre incelenmiş ve hata ihtimal grafiği verilmiştir. SEU sayısı arttıkça hatanın görülme ihtimali artmış ve yedekli modül sayısı arttıkça hatanın görülme ihtimali azalmıştır [22]. Çalışmada, konfigürasyon hafızasının yeniden okunarak hata tespitinin yapıldığı çalışmalar sadece durum bütünlüğünü bozan hataların tespitinde kullanılamayacağı için tercih edilmediği belirtilmiştir. Fakat durum bütünlüğünün bozulduğu fakat yapısal bütünlüğün bozulmadığı, atmosferde gerçekleşebilen hiçbir hata modeli bulunmamaktadır. Böyle bir hata ancak yarı kararlılık durumlarında gerçekleşebilir. Yarı kararlılık hataları Flip-Flop'ların SETUP ve HOLT zamanlarının ihlal edilmesi ile gerçekleşmektedir. Yarı kararlılık hatalarının çözümü için tasarım esnasında alınabilecek birçok yöntem bulunmakta olup TMR ve diğer hata tespit metotlarına ihtiyaç duyulmadan hata engellenebilmektedir. Yapılan çalışmada [22], dikkat çekici olan nokta, TMR harici kullanılan diğer hiçbir metot ile hatanın maskelenmesinin mümkün olmadığıdır. TMR metodunda diğer metotlara göre hata ihtimalinin daha düşük olduğu açıkça görülmektedir.

Hatanın varlığına bakılmaksızın PDR kullanılarak, belli bir ritimde tüm bloklar yeniden yazılarak hata toleranslı sistemin hedeflendiği çalışmalar da bulunmaktadır. Bu çalışmalardan birinde [18], aynı işi yapan dört farklı modül için dört farklı dinamik alan ayrılmıştır. Her dinamik alan hem fonksiyonu gerçekleştiren modülü bulundurabilmekte hem de boş tutulabilmektedir. TMR yapı kullanılmıştır. Fonksiyon aynı anda üç dinamik alanda birden aktif iken dördüncü dinamik alan boş bırakılmıştır. Hata modüllerin çıkışları oylayıcıdan geçirilerek sistem çıkışına etkisi engellenmiştir. Hatanın varlığına bakılmaksızın, boş olan dinamik alana modül yazılmış ve yedekli modüllerden bir tanesi, boş alana yazma işlemi bittikten sonra silinmiştir. Birinci, ikinci ve üçüncü alanda yer alan modüller aktif çalışmakta iken dördüncü alandaki modül PDR ile yeniden yazılmış, dördüncü alandaki modülün yazılma işlemi bittiğinde birinci alandaki modül silinmiştir. Sırası ile bu işlem belli bir ritimde tüm alanlar için tekrarlanmıştır. Fonksiyon, modüllerin yazma ve silme işlemlerinden etkilenmeden her an işlevini yerine getirmektedir. Yeniden yazma işlemi için FPGA dışı bir kaynak olan, bir bilgisayar kullanılmıştır. Bilgisayarda Tcl kodları ile yeniden yazma işlemi 3 saniye aralıklar ile tekrarlanmıştır. Hatanın bulunmasına gerek kalmadan tüm bloklar yeniden yazılarak, hata toleransı sağlanmıştır [18]. Sistemde, yeniden yazılan modüllerin diğer modüllere senkronizasyonundan ve statik alanlardan bahsedilmemiştir. Modüllerin yazılması için FPGA dışı kaynağa ihtiyaç duyulmuştur. TMR modül sayısı arttıkça, her bir TMR grubu için modüllerin yazılıp silinmesi gerekeceğinden, tüm sistemde ki modüllerin yeniden yazılma ritmi uzayacaktır.

PDR ve TMR kullanılan çalışmalardan birinde [21], hata TMR modül çıkışları karşılaştırılarak bulunmuş ve PDR kullanılarak düzeltilmiştir. Hata modeli olarak SEU harici SEBO ya karşıda önlem alınmaya çalışılmıştır. SEU kaynaklı hataların çözümü için yedekli modüllerin FPGA içinden ICAP arayüzü ile tekrardan yazılması önerilmiştir. SEBO kaynaklı hataların çözümü için ise, her TMR grubu için bir yedek modül alanı daha ayrılmıştır. Kalıcı hatanın görülmediği durumda ayrılan o alan güç harcamaması için boş tutulmuştur. Kalıcı hatanın görüldüğü durumda ise ICAP arayüzü ile boş alana yedek modül yazılarak, hatalı modül yerine TMR grubuna dahil edilmiştir. İkinci bir SEBO kaynaklı hatanın görülme

ihtimali düşük görüldüğünden herhangi bir çözüm gerekli görülmemiştir [21]. PDR ve TMR kullanılan çalışmalardan bir diğerinde [13], FPGA tasarımı iki kısımda incelenmiştir. Bu kısımlar statik modüller, dinamik modüller. Statik modüller hata toleranslı yapıda tasarlanmamıştır. Dinamik modüller TMR yapı ile tasarlanmış ve hata yedekli modül çıkışları karşılaştırılarak bulunmuştur. Hatalı modülü yeniden yazma işlemini yerine getiren kontrolcü modül dinamik kısma konulmuştur. Hatanın belirlendiği durumda kontrolcü FPGA içerisinden ICAP arayüzü ile hatalı modülün tamamını yeniden yazmıştır. Hatalı modülün yeniden yazılabilmesi için gereken parçalı bit akış verisinin flaş veya ROM bellekten alınması önerilmiştir. Yeniden yazılan yedek modüllerin senkronizasyonu için iki metot önerilmiş. Birincisi çalışan modülün durum bilgisinin diğer modüle aktarılmasıdır. İkinci metot ise yeni tur başlamadan önce üç yedek modülün yeniden başlatılmasıdır. İkinci metodun uygulaması daha az ek tasarım gerektirdiği için önerilmiştir. Tasarlanan yeniden yazma kontrolcüsünün kapladığı alan 858 SLICE olarak verilmiştir. ICAP arayüzünden 100 MHZ ile yazma yapıldığı belirtilmiştir [13]. TMR ve PDR ile hata toleranslı sistemin tasarlandığı literatürdeki iki çalışmada [21, 13], TMR metodu hatanın tespiti ve maskelenmesi için kullanılmıştır. Hata tespit edildikten sonra hatalı modülün tamamı ICAP arayüzü ile FPGA dışı bir kaynağa ihtiyaç duyulmadan yeniden yazılmıştır. ICAP arayüzünün kontrol edildiği kontrol modülünde TMR ile çoğaltılıp dinamik alanda yer alması önerilerek, yeniden yazma işleminin sebep olabileceği ciddi hataların önüne geçilmesi hedeflenmiştir. ICAP arayüzüne göre flaş ve ROM belleklerin bant genişliği dar boğaz yaratmaktadır. Hatanın düzeltilme hızı kullanılan ara yüz ICAP bile olsa flaş ve ROM'larda saklanan verinin okunma hızı ile sınırlanacaktır. Ayrıca önerilen iki tasarımda [21, 13], hata tespiti için TMR kullanıldığından, hatalı modülün tümünün baştan yazılması gerekmektedir. Bu da yeniden yazma süresi açısından dezavantaj oluşturmaktadır.

Literatürde yer alan TMR ve PDR kullanılan bir diğer [14] hata toleranslı FPGA tasarımında büyük parçalı TMR kullanılmış ve buna sebep olarak iki farklı bölgenin yol atamasında meydana gelebilecek hatanın ihtimalini azaltmak gösterilmiştir. Fakat büyük parçalı TMR da modül içi oylama olmadığı için ortaya senkronizasyon problemleri çıkmaktadır. Senkronizasyon problemi bozulan büyük parçalı yedek modül yeniden yazıldığında diğer yedekli modüllerden farklı

bir durumda başlayacağı için, yedekli modüllerin düzgün çalışmamasıdır. Bu duruma çözüm olarak devreyi senkronize eden kontrol noktaları konulması önerilmiştir. Yeniden yazılan modül çalışmaya başlamadan önce TMR yapısındaki diğer iki modülün kontrol durumuna gelmesini bekler ve kontrol durumundan sonra diğer modüllerle senkron olarak çalışmaya devam eder. Önerilen yapıda düzgün çalışan modüller çalışmayı hiç durdurmamaktadır. Devre düzgün çalışmaya devam ederken kontrol noktasında yeniden yazılan modül yedeklerine senkron olmaktadır. Sistemde gerçekleşen hatanın tespiti için oylayıcı çıkışı kullanılmıştır. Hatanın düzeltilmesi içinde büyük parçalı yedekli modülün tamamı yeniden yazılmaktadır. Yeniden yazmak için gerekli olan bit akış verilerinin, her yedekli modülün her yedeği için saklanması gereklidir. Önerilen yapı RS-232 verici (TX) ve alıcı (RX) modülleri üzerinde Virtex-2 FPGA ile denenmiştir. Çıkan sonuçlarda senkronizasyon için koyulan kontrol noktası yapısının, sadece TMR yapısına göre ek olarak getirdiği alan ve performans değerlerinin ihmal edilecek derecede küçük olduğu görülmüştür [14]. Kurulan yapıda, TMR sistemdeki yedekli modüller düşük çözünürlükte seçilerek statik alanda yer alacak olan oylayıcıların kapladığı alan azaltılmıştır. Çalışmada sadece yapısal bütünlüğün değil durum bütünlüğün de korunması esas alınmıştır. TMR yapıda yeniden yazılan modülün diğer modüllere senkronizasyonu için kontrol noktaları konulmuştur. Kontrol noktası yönteminin aviyonik sistemlerde senkronizasyon için kullanılabileceği makaleden açıkça görülmektedir. Fakat PDR işleminin, FPGA dışı bir kaynakla gerçekleştirilmesi ve hata tespiti için TMR metodunun kullanılması, hata düzeltme işlemini yavaşlatmaktadır.

Bu tezde temel alınan çalışma [24], incelenen diğer çalışmalarda karşılaşılan birçok eksikliğe çözüm üretmiştir. SEU kaynaklı hata tespiti için konfigürasyon belleğinin yeniden okunması metodu kullanılmıştır. Yeniden okuma ve PDR işlemi FPGA içi bir kaynak olan ve bant genişliği en yüksek konfigürasyon arayüzü olan ICAP kullanılmıştır. Virtex-4 FPGA'larda ve sonra gelen tüm Xilinx serisi FPGA'larda bulunan FRAME_ECC temel bileşeni, konfigürasyon belleğinde meydana gelen hataların tespiti için kullanılmıştır. ICAP arayüzünden konfigürasyon çerçevesi okunduğunda, FRAME_ECC bileşeni o çerçevede meydana gelen hatayı ve çerçeve içerisindeki yerini tespit edebilmektedir. Tespit edilen bu hata yine ICAP arayüzü ile tekrardan yazılarak düzeltilmiştir. Hatanın

düzeltilme işlemi teknolojinin izin verdiği en az sürede gerçekleştirilmiştir. Doğru konfigürasyon çerçevesi, okunan hatalı konfigürasyon çerçevesi düzeltilerek elde edilmiş ve FPGA dışı bir konfigürasyon saklama yöntemine ihtiyaç duyulmamıştır. Ayrıca zaten var olan veri üzerinden hatayı düzelterek, PDR yöntemi hem dinamik hem statik alanlar için kullanılabilmektedir. Hatanın ICAP okuma ve yazma işlemini yapan kontrolcüde meydana gelebileceği durumlar düşünülüp, ICAP kontrolcü bloğunda meydana gelebilecek hataların maskelenmesi için TMR yapı kullanılmıştır. 2-bit ve daha fazla hata olan durumlarda, hatanın FPGA içi kaynaklar ile düzeltilemeyeceği belirtilmiştir. 2-bit ve daha fazla olan hata durumun çözümü için FPGA dışı bir kaynak ile bütün olarak FPGA'nın yeniden şekillendirilmesinin gerektiği belirtilmiştir. Kurulan sistemin MTBF analizi yapılmıştır. Önerilen yapıda hata oranının 1 milyon kat azaldığı görülmüştür [24].

Bu tezde eksiklilerine çözüm sunulan, literatürdeki bu çalışmada [24], ICAP kontrolcüsü için TMR kullanılırken, sistemin diğer bileşenleri için nasıl bir hata maskeleme metodu kullanılacağı konusuna değinilmemiştir. Ayrıca ICAP kontrolcü modülü için önerilen TMR yapıda, yeniden yazılan modülün senkronizasyonu konularına değinilmemiştir. Sistemde statik bölgelerde meydana gelebilecek hataların yapısal olarak nasıl düzeltilebileceği belirtilirken, hata durumunda sistemin FPGA dışı diğer bileşenlerine olan etkisi incelenmemiştir. Kurulan yapı sadece yapısal bütünlüğün korunmasını esas almakta olup, durum bütünlüğü konusu hiç işlenmemiştir.

Literatürde bulunan çalışmaların incelenmesi ile hata toleranslı bir sistemde ortaya çıkabilecek temel eksiklikler şu maddeler ile özetlenebilir:

- Modül tasarımına CLB seviyesinde yaklaşım
- Modüllerin FPGA yerleşimine CLB seviyesinde yaklaşım
- Hata maskelemek için bir metot önerilmemesi
- Yüksek çözünürlükte TMR yapı
- TMR veya BIST yapıların hatanın tespiti için kullanılması
- PDR işleminin FPGA dışı bir kaynak ile yavaş yapılması

- PDR işlemi ICAP arayüzü kullanılarak yapılırken, ICAP kontrolcüsünde meydana gelebilecek hataların ihtimalinin azaltılmaması
- TMR modül senkronizasyonu diğer bir deyişle durum bütünlüğü konularına değinilmemesi

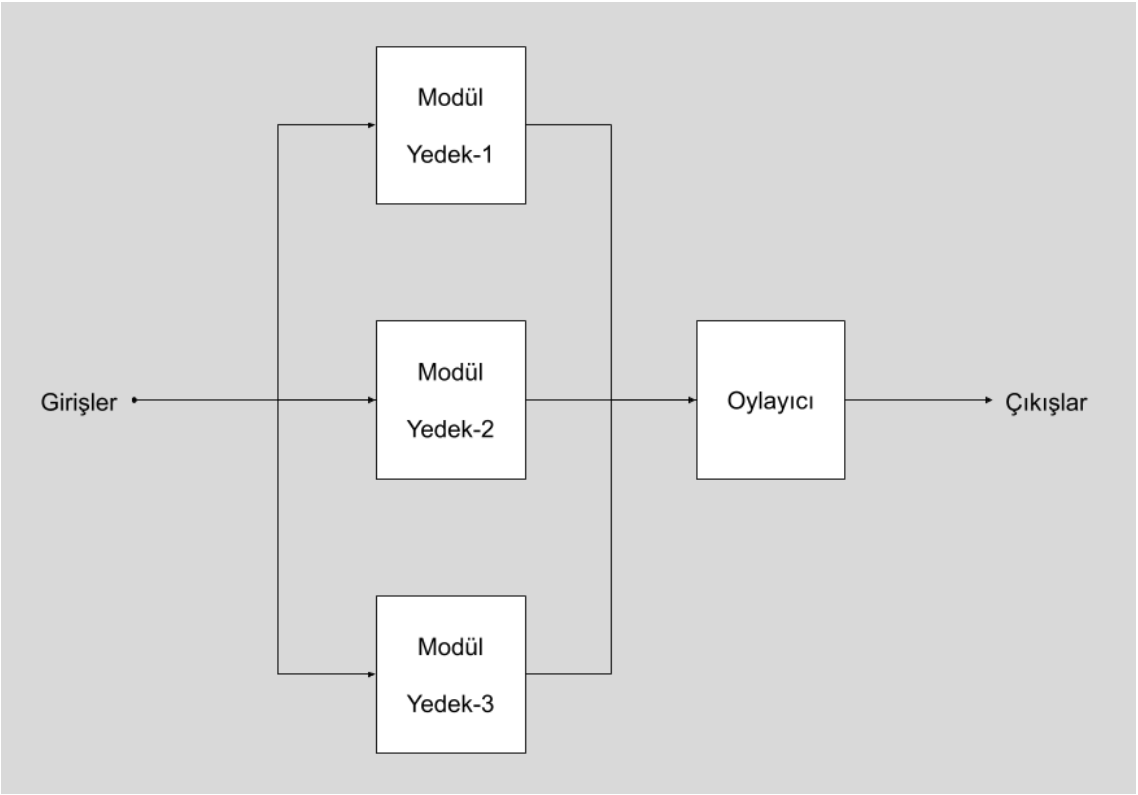
Tezin bir sonraki bölümünde, literatürde bulunan çalışmaları temel alan ve çalışmaların eksik yanlarını tamamlayan hata toleranslı bir mimari önerilmiştir.

3. HATA TOLERANSLI FPGA TASARIMI

SEU kaynaklı hatalara toleranslı bir FPGA tasarımında, hata gerçekleştiğinde hatanın maskelenmesi, hatanın tespit edilmesi ve hatanın düzeltilmesi gereklidir. Tezin bu bölümünde kullanılacak olan hata maskeleme yöntemi, hata tespit yöntemi ve hata düzeltme yöntemi anlatılmıştır. Literatürde incelenen çalışmalar temel alınarak kullanılacak olan yöntemlerin seçilmesinde etkili olan sebeplerden bahsedilecek ve literatüre katkıları anlatılacaktır.

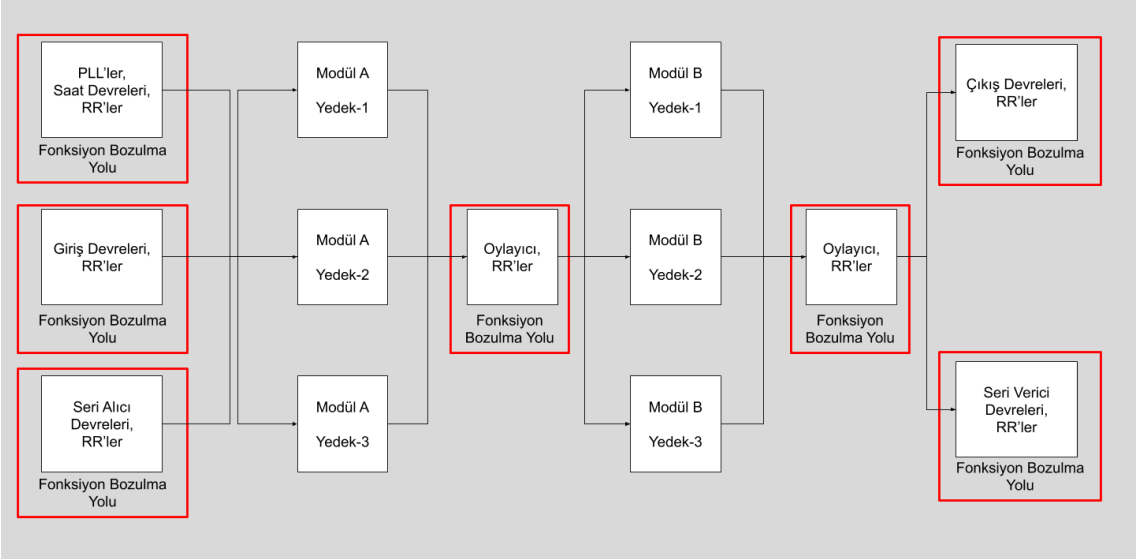
3.1. Hata Maskeleme Yöntemi

Literatürdeki çalışmalar [13, 14, 15, 16, 18, 21, 24] incelendiğinde SEU kaynaklı hataların TMR kullanılarak maskelenebildiği görülmektedir. Standart TMR yapı Şekil 3-1'de görülebilir. TMR modüllerden herhangi birinde meydana gelecek olan hata, diğer modüllerin doğru çalışmaya devam etmesi ile devrenin çalışmasına olumsuz etkisi engellenmekte, oylayıcı çıkışında hata maskelenmektedir. Ayrıca düşük çözünürlükte TMR yapı ile oylayıcıların getireceği ek devre ihtiyacı minimize edilebilir.



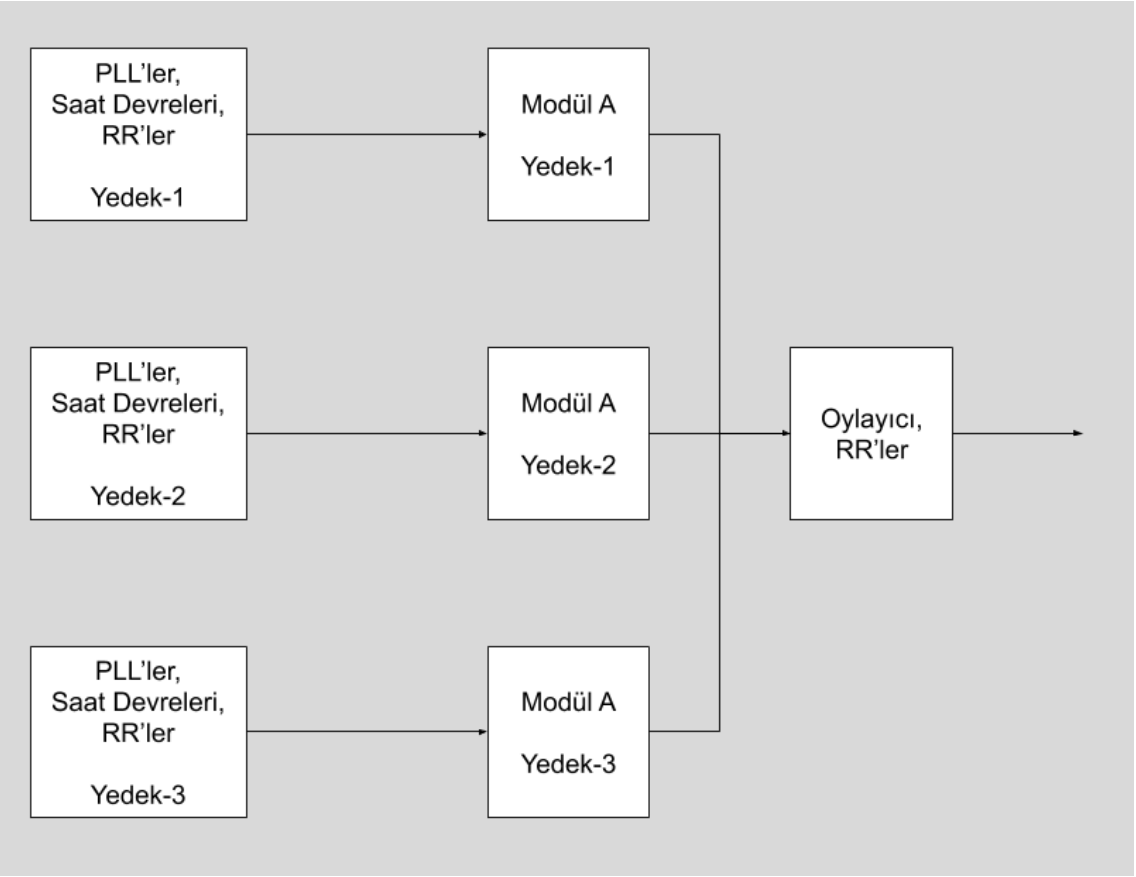
Şekil 3-1 TMR yapı

PLL ve saat devrelerinde, giriş çıkış devrelerinde ve seri alıcı vericilerde meydana gelen hataların maskelenmesi konusuna incelenen çalışmalarda [13]-[24] hiç değinilmemiştir. TMR çıkışı oylayıcıda meydana gelebilecek hatalar ve oylayıcılardan sonra çıkış devrelerinde ve seri vericilerde meydana gelebilecek hatalar, modüllerde gerçekleştirilen fonksiyonların yanlış sonuç vermesine sebep olabilmektedir. TMR modüllerin haricinde hatanın gerçekleşebileceği, fonksiyon bozulma yolları Şekil 3-2'de verilmiştir.



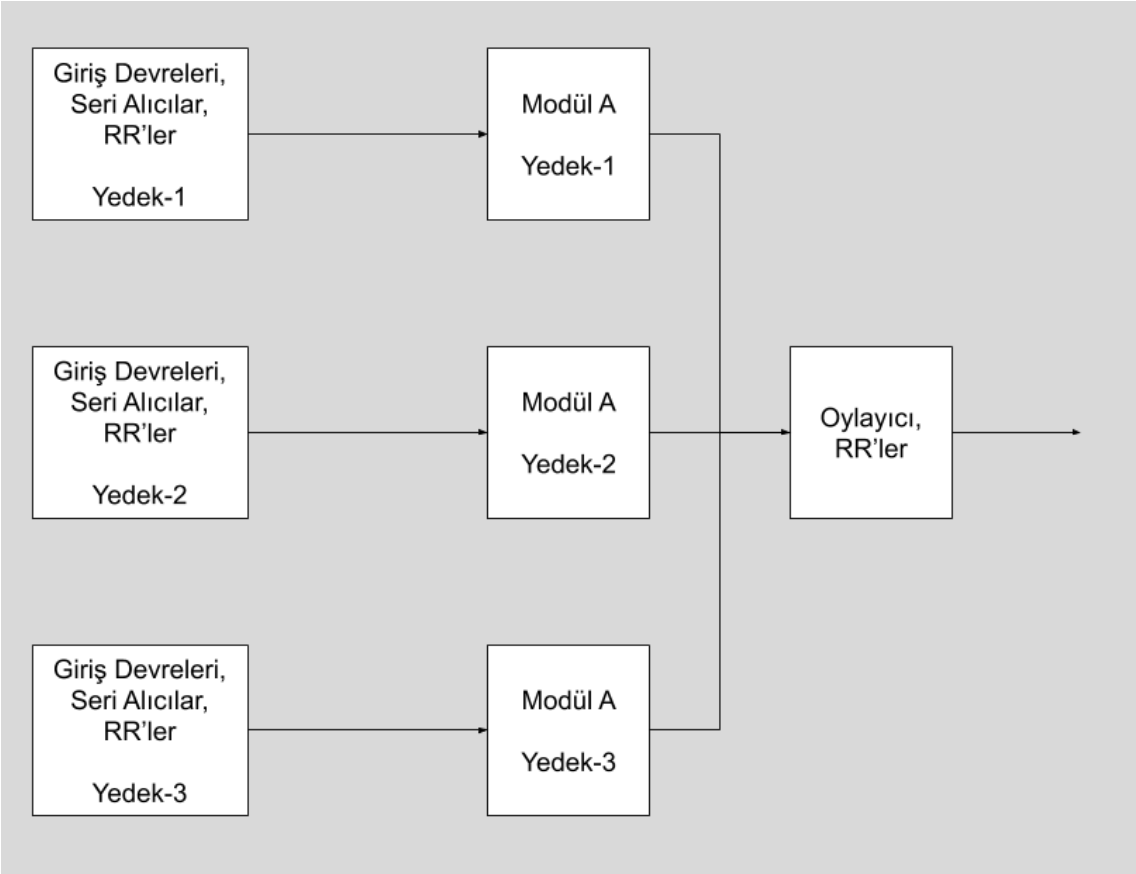
Şekil 3-2 Fonksiyon bozulma yolları

TMR modüllerin haricinde gerçekleşen hatalardan saat ile alakalı olan hataların maskelenmesi için TMR bileşenindeki her bir modülün saat kaynağının farklı PLL ve saat devreleri ile beslenmesi yöntemi kullanılabilir. Önerilen yapı Şekil 3-3'de görülmektedir. Bu yöntem ile de görülen Yedek-1'de, Yedek-2'de ve Yedek-3'de bulunan PLL'lerde, saat devrelerinde ve bu devrelerin diğer modüller ile bağlantısını sağlayan RR'lerde gerçekleşecek SEU sadece bağlandığı TMR modülünü etkileyecek ve hata oylayıcı çıkışında maskelenecektir.



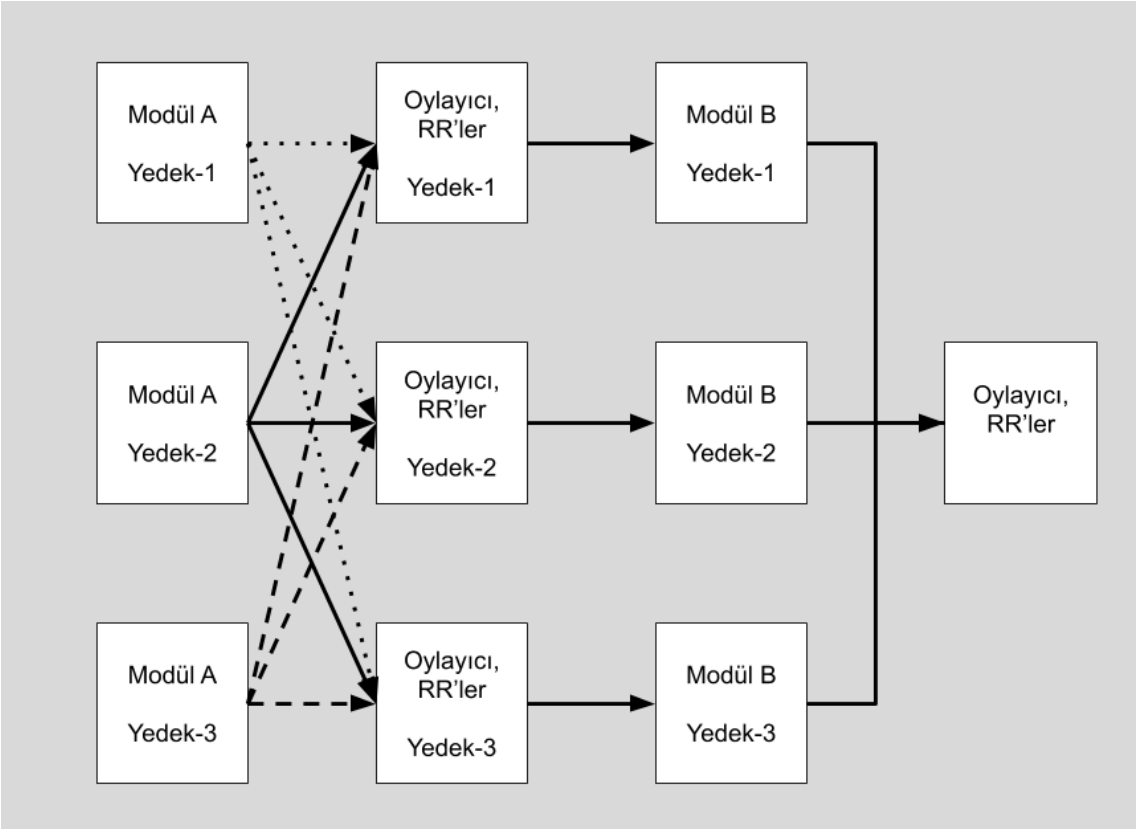
Şekil 3-3 Bağımsız saat devreleri

TMR modüllerin öncesinde bulunan giriş sinyallerinde ve seri alıcılarda gerçekleşen hatalar, her bir modül için ayrı giriş devresi ve seri alıcı kullanılarak maskelenebilir. Bu yapı Şekil 3-4'de verilmiştir. Bu yöntem ile Şekil 3-4'de görülen Yedek-1'de, Yedek-2'de ve Yedek-3'de bulunan giriş devrelerinde, seri alıcılarda ve bu devrelerin diğer modüller ile bağlantısını sağlayan RR'lerde gerçekleşecek SEU sadece bağlandığı TMR modülünü etkileyecek ve hata oylayıcı çıkışında maskelenecektir.



Şekil 3-4 Bağımsız giriş devreleri ve seri alıcılar

TMR oylayıcı çıkışları başka bir TMR yapı için giriş olarak kullanılabilir. Bu durumda oylayıcıda meydana gelebilecek hata, oylayıcı çıkışlarının giriş olarak kullanılacağı TMR yapıdaki her bir modül için ayrı oylayıcı kullanılması ile maskelenebilir. Yedekli oylayıcı mimarisi Şekil 3-5'de verilmiştir. Bu tarz bir yapıya literatürde rastlanmamıştır. Bu yöntem ile Şekil 3-5'de görülen Yedek-1'de, Yedek-2'de ve Yedek-3'de bulunan oylayıcılarda ve bu oylayıcıların diğer modüller ile bağlantısını sağlayan RR'lerde gerçekleşecek SEU sadece bağlandığı TMR modülünü etkileyecek ve hata en dışta bulunan oylayıcı çıkışında maskelenecektir. Bu yapı ile oylayıcı modüllerin, bağlandığı TMR bölgeye alınıp düzeltilmesi sağlanabilir.

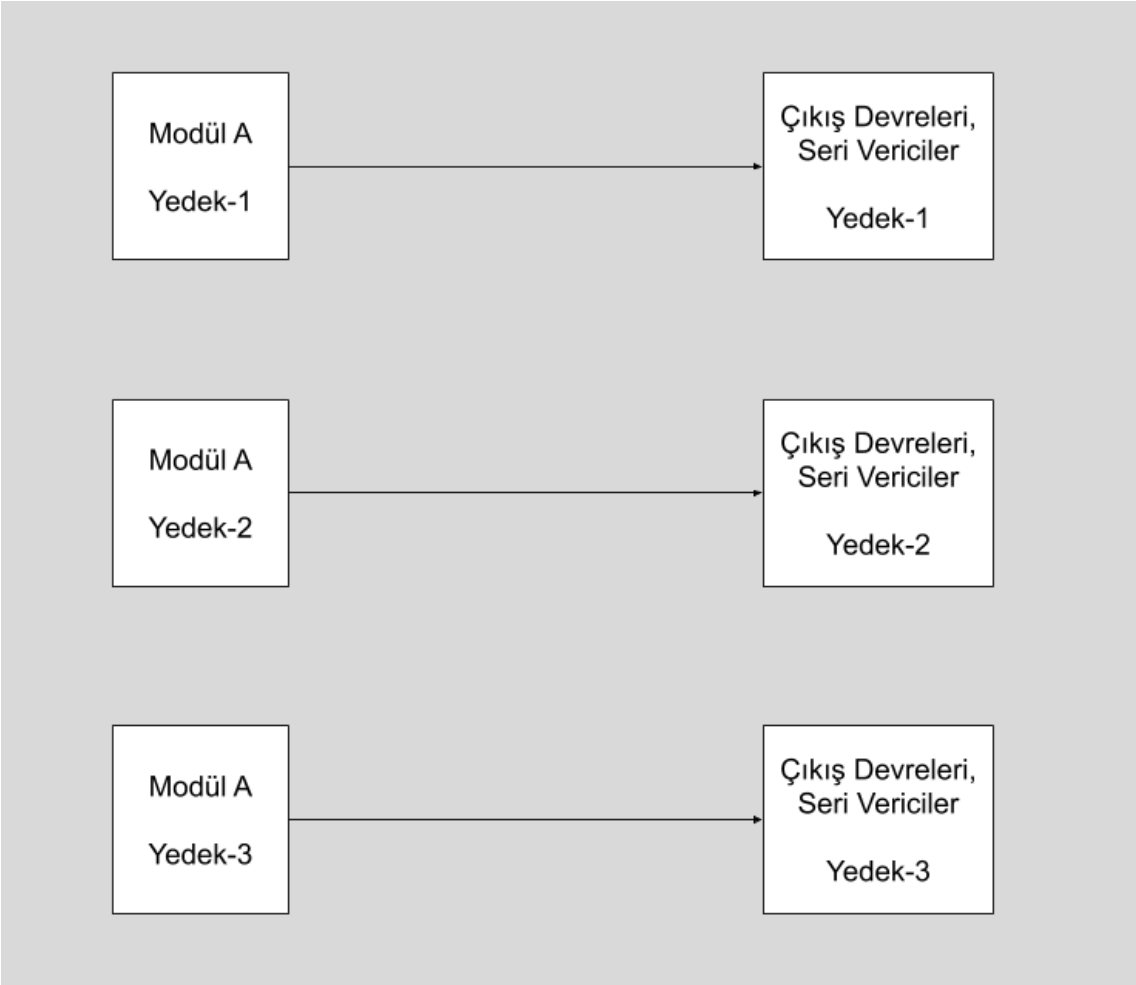


Şekil 3-5 Her modül öncesi ayrı oylayıcılı yapı

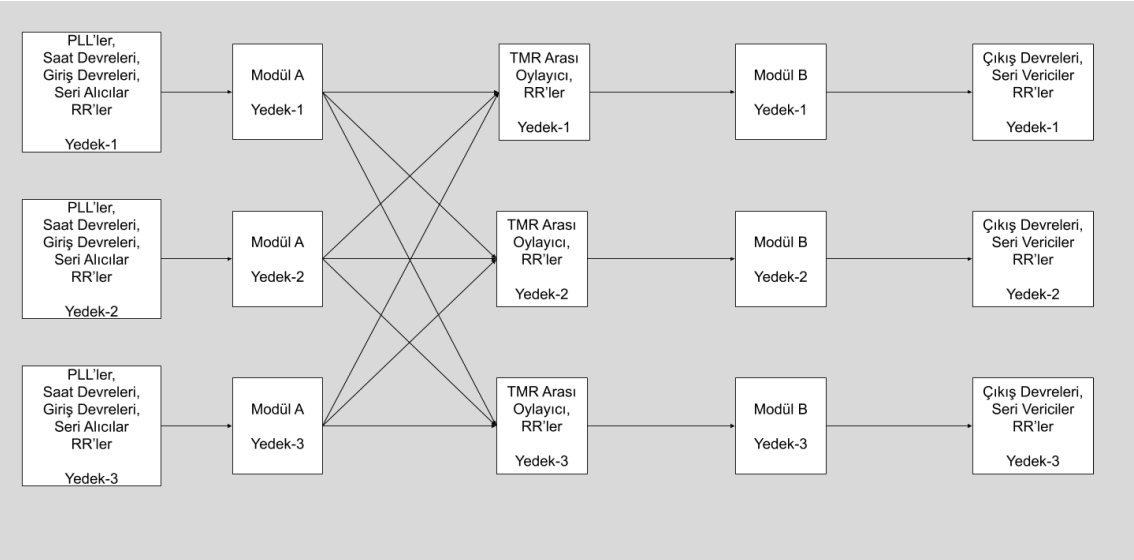
TMR yapıda kullanılan fonksiyon çıkışında, oylayıcılarda, RR'lerde ve çıkış devrelerinde oluşan hatanın maskelenmesi, sadece FPGA dışına çıkan sinyallerin yedeklenmesi ile mümkündür. Bu hataların çözümü için sadece FPGA tasarımında yapılacak önlemler yeterli olmayacaktır. Nasıl bir tasarım yapılırsa yapılsın çıkış devresinde meydana gelebilecek bir SEU hatası, öncesinde yer alan devrenin sonucu önemli olmaksızın çıkışı değiştirecektir. Bu tarz hataların çözümü için çıkış ve seri verici devrelerinin yedeklenmesi yöntemi dışında verinin güvenilirlik durumunu gösteren protokoller kullanılabilir.

Yedekli çıkış, seri verici devreleri ile kurulan yapının dezavantajı sınırlı sayıda olan çıkış devresi ve seri verici devrelerin yedekli olarak kullanılmasıdır. Yedekli çıkış, seri verici devreleri için önerilen yapı Şekil 3-6'de verilmiştir. Bu yöntem ile Şekil 3-6'de görülen, modüllerin bağlandığı Yedek-1'de, Yedek-2'de ve Yedek-3'de bulunan çıkış devrelerinde, seri vericilerde ve bu devrelerin diğer modüllerle bağlantısını sağlayan RR'lerde gerçekleşecek SEU sadece bir çıkışı etkileyecek ve hata FPGA dışında bu üç çıkış kullanılarak maskelenecektir. Her hata kısa

sürede düzeltilebilmektedir. Bu yapı sadece, gerçekleşecek hatalarından kritik olanları için kullanılmalıdır. Gerçekleşen hata kısa bir sürede FPGA tekrardan şekillendirilip düzeltileceği için kritiklik teşkil etmiyor olabilir. Bu metodun her sinyal çıkışı için kullanılması, sınırlı sayıda olan çıkış devrelerini 3 kat daha fazla kullanılacağı için aviyonik sistemler için gerçekçi değildir. Fakat kritik olan çıkışların bu metot ile kullanılması hatayı maskelemektedir.



Şekil 3-6 Bağımsız çıkış devreleri



Şekil 3-7 Hata maskeleme yapısı

Hata maskelemek için kullanılabilecek olan yöntemler ile oluşan yapı Şekil 3-7'de verilmiştir. Bu yöntemler şu maddeler ile özetlenebilir:

- Modüllerde meydana gelebilecek hataların maskelenmesi için TMR yapı kullanılacaktır.
- TMR yapıda bulunan her bir modül için saat kaynakları farklı PLL ve saat devrelerinden sağlanacaktır.
- TMR devrede bulunan modül giriş yolları yedeklenecek ve başka bir TMR yapıdan gelen sinyaller, TMR yapıda bulunan her bir modüle girmeden önce oylanacak ve her bir modül için ayrı oylayıcı kullanılacaktır.
- Çıkışlar kritik sinyaller için yedeklenecektir.

Kurulan mimaride tüm hatalar maskelendiği için, düzeltilmese dahi hata bulunmayan yedekler düzgün sonuç vermeye devam edeceğinden, sistem doğru çalışmaya devam edecektir. Fakat hataların düzeltilmediği durumda, hatalar birikebilir. Yedek modüllerden ikisinde hata oluştuğu taktirde sistemin maskeleme mekanizmasını bozulacaktır. Bu sebeple her hatanın düzeltilmesi gereklidir.

3.2. Hata Tespit Yöntemi

Literatürdeki çalışmalar incelendiğinde, SEU kaynaklı hata tespiti için CLB seviyesi yapılan tasarımlarda [19, 20] ve BIST kullanılan tasarımlarda [22, 23], ölçeklendirme problemi bulunmaktadır. TMR modül çıkışları karşılaştırılarak hatanın bulunduğu tasarımlarda [13, 14, 15, 16, 18, 21, 24] ise hatanın bulunduğu konfigürasyon çerçevesinin tespit edilemediği, ancak modül seviyesinde hata tespiti yapılabildiği görülmüştür. Hata tespiti için konfigürasyon belleğinin yeniden okunması ile hatanın gerçekleştiği bitin konumunun tespit edilebildiği literatürde yer alan çalışmada [24] gösterilmiştir.

Bu tezde hata tespiti için, Legat [24] çalışmasında kullandığı, konfigürasyon verisinin ICAP üzerinden okunarak FRAME_ECC bileşeni ile hatalı bitin konumunun bulunması yöntemi seçilmiştir. Legat çalışmasında [24] kullandığı bu seçimin olumlu yönleri şu maddeler ile özetlenebilir:

- Hata, okunabilen ve yeniden şekillendirilebilen en küçük alan olan konfigürasyon çerçevesi seviyesinde bulunabilmektedir.
- Hatanın bulunabilmesi için CLB seviyesi tasarıma ve CLB seviyesi yerleşim kısıtlarının girilmesine gerek bulunmamaktadır.
- Hatanın bulunabilmesi için her fonksiyon için özel olarak tasarlanması gereken BIST yapılarının kurulmasına gerek duyulmamaktadır.
- Hatalı bölgenin düzeltilebilmesi için gereken bit akış verisi, okunan konfigürasyon çerçevesinde hatalı olan bit düzeltilerek elde edilebilmekte, bit akış verisinin başka bir depolama biriminde saklanmasına gerek duyulmamaktadır.
- Hatanın gerçekleştiği bölgenin sisteme etkisi değerlendirilebilmektedir. Bu değerlendirme olmadan, yapısal bütünlüğü bozan hatanın düzeltilmesi ile durum bütünlüğü sağlanamaz.
- ICAP arayüzü FPGA konfigürasyonun en hızlı okunup yazılabildiği ara yüzdür. ICAP kullanılarak FPGA dışı bir kaynağa ihtiyaç duyulmadan konfigürasyon belleği okunabilmektedir.

Hata tespiti için kullanılacak bloklarda gerçekleşecek olan hataların maskelenmediği tasarımlar, düzgün çalışan sistemin bile yeniden

şekillendirilmesine ve bozulmasına sebep olacaktır. Hata tespitinde kullanılacak bloklar için hata maskeleme yöntemi olarak seçilen yöntemler uygulanacaktır.

3.3. Hata Düzeltme Yöntemi

Literatürdeki çalışmalar incelendiğinde PDR metodunun konfigürasyon belleğinde meydana gelen yapısal bütünlüğü bozan SEU kaynaklı hataların çözümünde etkili olduğu görülmüştür. 2-bit ve daha fazla SEU kaynaklı hataların çözümü için ise FPGA'nın bütün olarak yeniden şekillendirilmesi gerekmektedir. Bütün olarak şekillendirme sadece FPGA dışı bir kaynak tarafından yapılabilmektedir.

Bu tezde yapısal bütünlüğü bozan SEU kaynaklı hataların düzeltilmesi için PDR ile ICAP üzerinden sadece hatalı konfigürasyon çerçevesinin yeniden şekillendirilmesi yöntemi kullanılmıştır. 2-bit ve daha fazla olan hatalarda ise FPGA dışı bir kaynak kullanılarak bütün konfigürasyon şekillendirme yöntemi kullanılabilir. FPGA'nın ilk açılışta bütün şekillendirmesini yapan kaynak, 2-bit ve daha fazla hataların çözümünde kullanılabilir. Bu kaynak FPGA'ya özel olan ve ilk açılışta konfigürasyon şekillendirmesi yapan bir flaş bellek olabilir. Bu tarz bellekler FPGA'nın konfigürasyon durumunu raporlayan çıkış değerlerinin durumuna göre belleğinin şekillendirmesi işlevini yerine getirirler. Konfigürasyon durumunu bildiren çıkış değerlerinin bağlandığı bellek girişleri statik hata durumunda tetiklenerek bütün konfigürasyon işlemi başlatılabilir. FPGA'nın bütün şekillendirilmesinin yapıldığı diğer kaynak bir CPLD veya flaş tabanlı bir FPGA olabilir. CPLD ve flaş tabanlı FPGA'lar her açılışta şekillendirilmeye ihtiyaç duymazlar. Bu akıllı aygıtlar, 2-bit ve daha fazla olan hata durumu bildirilerek, bütün şekillendirme işlemi tetiklenebilir.

PDR yapısal bütünlüğünü bozan hataların çözümü için etkili bir yöntem olsa da durum bütünlüğünün korunması için PDR kullanılan çalışmalar [13]-[24] incelendiğinde, sunulan çözümlerde ölçeklendirme problemi bulunmaktadır. Hata maskeleme yöntemi olarak TMR seçildiğinden dinamik bölgede meydana gelen bir SEU hatasının başka modüllerin çalışmasını etkileme ihtimali bulunmamaktadır. Ancak TMR yapı içerisinde bulunan hatalı modül PDR ile yeniden şekillendirildiğinde yapısal bütünlüğün sağlanmasına rağmen, durum

bütünlüğü için aynı şey söylenemez. Durum bütünlüğünün sağlanması için yeniden yazılan modülün diğer yedekleri ile senkronize edilmesi gerekmektedir. Literatürdeki çalışmalar incelendiğinde, Azambuja [14] önerdiği, kontrol noktası metodunun etkili olduğu görülmektedir. Kontrol noktası metodunun uygulanabilmesi için modüllerin özel olarak tasarlanması gerekmektedir. Fakat bu özel tasarım modül seviyesinde yapılacağından, ek tasarım yükü azdır.

FPGA konfigürasyon belleğinde bulunan, yapısal hataya sebep olmayacak, BLOK RAM'lerde gerçekleşen SEU hatalarının çözümü için, bu tezde ECC modülü kullanılacaktır.

Hata düzeltmede kullanılacak bloklarda gerçekleşecek olan hataların maskelenmediği tasarımlar, düzgün çalışan sistemin bile yeniden şekillendirilmesine ve bozulmasına sebep olacaktır. Bu tezde hata düzeltmede kullanılacak bloklar için hata maskeleme yöntemi olarak seçilen yöntemler uygulanacaktır.

4. ÖNERİLEN YAPININ FPGA ÜZERİNDE UYGULAMASI

Tezin bu bölümünde, seçilen hata maskeleme, hata düzeltme ve hata tespit yöntemlerinin Xilinx Virtex-6 FPGA'da uygulanışı anlatılmıştır. Kurulan yapıda kullanılan blokların tasarımı açıklanmıştır. Önerilen hata maskeleme, hata tespit ve hata düzeltme yöntemlerinin analiz edilebilmesi için, aviyonik sistemlerde en yaygın kullanılan ARINC-429 iletişim protokolünün kullanıldığı bir sistem seçilmiştir. Tezin bu bölümünde kullanılan her bir modülün görevleri ve tüm modüllerin TMR yapısı açıklanmıştır.

4.1. ICAP_VIRTEX6

ICAP_VIRTEX6 devresi, FPGA içerisinde yer alan, FPGA tasarımının kendisi tarafından konfigürasyon belleğinin okunup yazılmasını sağlayan Xilinx Virtex-6 temel fonksiyonlardandır. ICAP_VIRTEX6 devresi, programlanarak değiştirilemez. Dolayısı ile SEU hatalarının gerçekleşmediği bir yapıdır. ICAP_VIRTEX6 devresi en fazla 100Mhz frekansında çalışabilmektedir. ICAP_VIRTEX6 devresinin giriş ve çıkış sinyalleri ve seçilebilir öznitelikleri Çizelge 4-1 ve Çizelge 4-2'de verilmiştir. I giriş sinyali ve O çıkış sinyali 32-bit, 16-bit ve 8-bit olarak seçilebilmektedir.

Çizelge 4-1 ICAP_VIRTEX6 giriş çıkış sinyalleri

Sinyal İsmi	**Sinyal Tipi**	**Açıklama**
CLK	Giriş	ICAP arayüzü saat girişi
CSB	Giriş	0 iken aktif ICAP arayüzü seçici sinyali
RDWRB	Giriş	1 = Okuma, 0 = Yazma
I [31:0]	Giriş	ICAP yazma veri yolu
O [31:0]	Çıkış	ICAP okuma veri yolu
BUSY	Çıkış	1 iken aktif veri yolu meşgul sinyali. Yazma işlemi esnasında her zaman 0.

Çizelge 4-2 ICAP_VIRTEX6 öznitelikleri

Öznitelik İsmi	**Öznitelik Tipi**	**Varsayılan Ayarlar**	**Açıklama**
DEVICE_ID	HEX	<ID Code> (32'h04244093)	Önceden programlanmış ID kodunu belirler.
ICAP_WIDTH	ENUM	X8, X16, X32 (X8)	ICAP veri yolu genişliğini belirler.
SIM_CFG_FILE_NAME	String	NONE, <String> (NONE)	Simülasyon modeli tarafından kullanılacak RBT dosyasını belirler.

ICAP arayüzünden yazma işleminde, ICAP_VIRTEX6 devresi tüm sinyalleri saatin kalkan kenarında örnekler ve sürer. ICAP_VIRTEX6 devresi RDWRB ve CSB sinyallerinin '0' sürüldüğü ilk kalkan kenarda I [31:0] sinyalini örneklemeye başlar ve örneklenen değeri konfigürasyon belleğinin ilgili çerçevesine yazar. Minimum yazma miktarı 2592 bit olan bir konfigürasyon belleği çerçevesidir.

ICAP arayüzünden okuma işleminde, ICAP_VIRTEX6 devresi tüm sinyalleri saatin kalkan kenarında örnekler ve sürer. ICAP_VIRTEX6 devresi RDWRB sinyalinin 1 ve CSB sinyalinin 0 sürüldüğü ilk kalkan kenarda veri okuma işlemini başlatır. ICAP_VIRTEX6 devresi çerçeve verisini O [31:0] sinyali ile gönderir, gönderilecek olan verinin hazır olduğunu göstermek için BUSY sinyalini '0' sürer. Minimum okuma miktarı 2592 bit olan bir konfigürasyon belleği çerçevesidir.

ICAP modülü bu tezde olabilecek en yüksek bant genişliğinde kullanılmıştır. 100Mhz saat frekansı ile beslenmiş ve giriş çıkış veri yolu 32-bit olarak seçilmiştir.

4.2. FRAME_ECC_VIRTEX6

FRAME_ECC_VIRTEX6, ICAP_VIRTEX6'de olduğu gibi, FPGA içerisinde yer alan, programlanarak değiştirilemeyen, dolayısı ile SEU hatalarının gerçekleşemeyeceği, Xilinx Virtex-6 temel FPGA fonksiyonudur. FRAME_ECC_VIRTEX6 bloğu çıkış sinyalleri ve öznitelikleri Çizelge 4-3 ve Çizelge 4-4'de verilmiştir.

Çizelge 4-3 FRAME_ECC_VIRTEX6 çıkış sinyalleri

Sinyal İsmi	**Sinyal Tipi**	**Açıklama**
SYNDROMEVALID	Çıkış	SYNDROME ve diğer çıkış sinyallerinin örneklenmesi için kullanılır. Her bir çerçeve okunduktan sonra 1 saat çevrimi 1 değerini alır.
ECCERROR	Çıkış	Eğer çerçevede hata varsa 1, hata yoksa 0 değerini alır. SYNDROMEVALID 1 iken örneklenir.
SYNDROME [12:0]	Çıkış	Çerçevede gerçekleşen hatanın türünü ve yerini bulmada kullanılır. SYNDROMEVALID 1 iken örneklenir. Hata yok: [12] = 0, [11:0] = 0 1-bit hata: [12] = 1, [11:0] = hatalı bitin yeri 2-bit hata: [12] = 0, [11:0] = geçersiz değer Hamming kodu hatası: [12] =1, [11:0] = 0
CRCERROR	Çıkış	READBACK CRC yöntemi kullanıldığında döngüsel artıklık kodu (CRC) hatası olduğunu gösterir. SYNDROMEVALID 1 iken örneklenir.
FAR [23:0]	Çıkış	En son okunan Konfigürasyon belleği çerçevesinin adresini gösterir. SYNDROMEVALID 1 iken örneklenir.

SYNWORD [6:0]	Çıkış	ECCERRORSINGLE 1 iken geçerlidir. SYNDROME bitleri kullanılarak oluşturulmuştur. Hatanın çerçevede kaçıncı 32-bitde gerçekleştiği bilgisini verir. SYNDROMEVALID 1 iken örneklenir.
SYNBIT [4:0]	Çıkış	ECCERRORSINGLE 1 iken geçerlidir. SYNDROME bitleri kullanılarak oluşturulmuştur. Hatanın 32-bitin hangi bitinde gerçekleştiği bilgisini verir. SYNDROMEVALID 1 iken örneklenir.
ECCERRORSINGLE	Çıkış	1-bitlik hata durumunda 1 değerini alır. SYNDROMEVALID 1 iken örneklenir.

Çizelge 4-4 FRAME_ECC_VIRTEX6 öznitelikleri

Öznitelik İsmi	**Öznitelik Tipi**	**Varsayılan Ayarlar**	**Açıklama**
FARSRC	String	EFAR, FAR (EFAR)	FAR [23:0] çıkış sinyalinin okunan çerçevenin adresini mi yoksa hatalı bitin adresini mi göstereceğinin belirlenmesinde kullanılır.
FRAME_RBT_IN_FILENAME	String	NONE, <String> (NONE)	Simülasyon modeli tarafından kullanılacak RBT dosyasını belirler.

FRAME_ECC_VIRTEX6 devresi konfigürasyon belleğinde gerçekleşen 1 ve 2 bitlik hataları tespit eder. Sentez programı ile konfigürasyon belleği bit akış verisini yaratırken her konfigürasyon belleği çerçevesi için 13-bitlik Hamming kodu oluşturur. Konfigürasyon belleği okunurken FRAME_ECC_VIRTEX6 devresi bu 13-bitlik Hamming kodunu kullanarak, okunan konfigürasyon belleği çerçevesi için sendrom değeri hesaplar. Eğer SEU hatası sonucu konfigürasyon belleği çerçevesinde bir hata gerçekleşmemişse SYNDROME [12:0] çıkış değerini 0 sürülür. Eğer 1-bitlik SEU hatası gerçekleşmişse SYNDROME [12] değeri 1 sürülür ve hatanın gerçekleştiği bitin yeri SYNDROME [11:0] çıkışında verilir. Eğer 2-bitlik bir hata gerçekleşmişse SYNDROME [12] değeri 0 olur, geri kalan bitler ise 0'a eşit olmayan bir değer alır. Eğer 2-bitden fazla hata varsa ECCERROR çıkışı 1 olur ve hatanın yeri belirsizdir. Konfigürasyon belleğinin hangi çerçevesi okunuyorsa FRAME_ECC_VIRTEX6 devresi o çerçeve için sonuç verir. FRAME_ECC_VIRTEX6 devresinin çalışması için konfigürasyon belleğinin okunması gerekmektedir. Konfigürasyon çerçevesi okunması bittiğinde, SYNDROMEVALID sinyali saatin yükselen kenarında 1'e çıkar. Tüm FRAME_ECC_VIRTEX6 çıkış sinyallerinin SYNDROMEVALID sinyalinin 1'e çıkmadan önceki hali esas alınmalıdır.

4.3. SEU Kontrol Modülü

SEU kontrol modülü hatayı bulma ve hatayı düzeltme fonksiyonlarının gerçekleştirildiği modüldür. SEU kontrol modülü sinyalleri Çizelge 4-5'de verilmiştir.

Çizelge 4-5 SEU kontrol modülü giriş çıkış sinyalleri

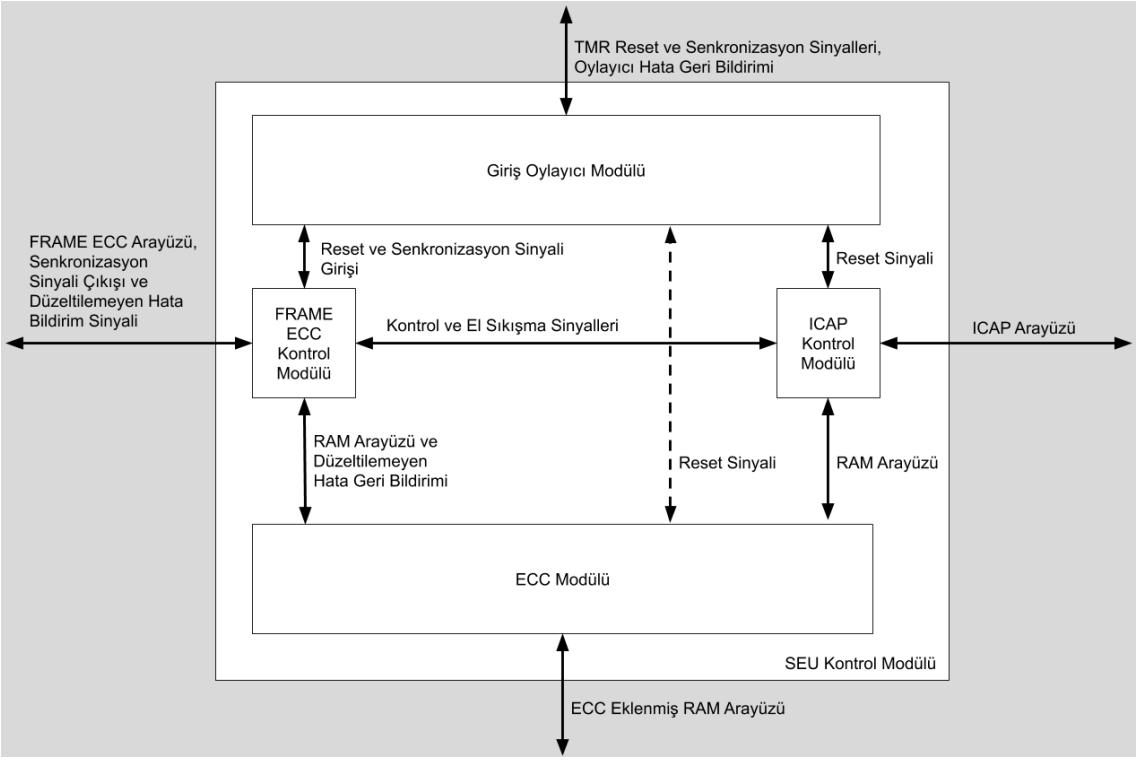
Sinyal Adı	**Sinyal Tipi**	**Açıklama**
CLK	Giriş	ICAP arayüzü ile aynı frekanstaki saatin 180 derece kaydırılmış halidir.

RESET	Giriş	SEU kontrol modülü sıfırlama girişidir.
SYNC_IN	Giriş	Senkronizasyon için kullanılan giriş sinyalidir.
SYNC_OUT	Çıkış	Senkronizasyon için kullanılan çıkış sinyalidir.
LAST_FRAME_ADDRESS_IN	Giriş	Senkronizasyon için kullanılan, sırada okunacak olan konfigürasyon belleği adresi giriş sinyalidir.
LAST_FRAME_ADDRESS_OUT	Çıkış	Senkronizasyon için kullanılan, sırada okunacak olan konfigürasyon belleği adresi çıkış sinyalidir.
BUSY	Giriş	ICAP arayüzünden gelen BUSY sinyalidir.
O [31:0]	Giriş	ICAP arayüzünden gelen O sinyalidir.
CSB	Çıkış	ICAP arayüzüne bağlanan CSB sinyalidir.
RDWRB	Çıkış	ICAP arayüzüne bağlanan RDWRB sinyalidir.
I [31:0]	Çıkış	ICAP arayüzüne bağlanan I sinyalidir.
SYNDROME [12:0]	Giriş	FRAME_ECC arayüzünden gelen SYNDROME sinyalidir.

SYNDROMEVALID	Giriş	FRAME_ECC arayüzünden gelen SYNDROMEVALID sinyalidir.
ECCERROR	Giriş	FRAME_ECC arayüzünden gelen ECCERROR sinyalidir.
RAM_DATA_IN_A [38:0]	Giriş	2 girişli BLOK RAM arayüzünün A portundan gelen veri çıkış sinyalidir.
RAM_DATA_OUT_A [38:0]	Çıkış	2 girişli BLOK RAM arayüzünün A portuna bağlanan veri çıkış sinyalidir.
RAM_WR_EN_A	Çıkış	2 girişli BLOK RAM arayüzünün A portuna bağlanan yazma kontrol sinyalidir.
RAM _ADDR_A	Çıkış	2 girişli BLOK RAM arayüzünün A portuna bağlanan adres kontrol sinyalidir.
RAM_DATA_IN_B [38:0]	Giriş	2 girişli BLOK RAM arayüzünün B portundan gelen veri çıkış sinyalidir.
RAM_DATA_OUT_B [38:0]	Çıkış	2 girişli BLOK RAM arayüzünün B portuna bağlanan veri çıkış sinyalidir.

RAM_WR_EN_B	Çıkış	2 girişli BLOK RAM arayüzünün B portuna bağlanan yazma kontrol sinyalidir.
RAM _ADDR_B	Çıkış	2 girişli BLOK RAM arayüzünün B portuna bağlanan adres kontrol sinyalidir.
REQUEST_FULL_RECONFIGURATION	Çıkış	Düzeltilemeyen hataları FPGA dışına bildirmekte kullanılan sinyaldir.
MODULE_NUMBER	Çıkış	Oylayıcıda hata görülmesi ile sıfırlanması gereken modülün yerini bildirmekte kullanılır.

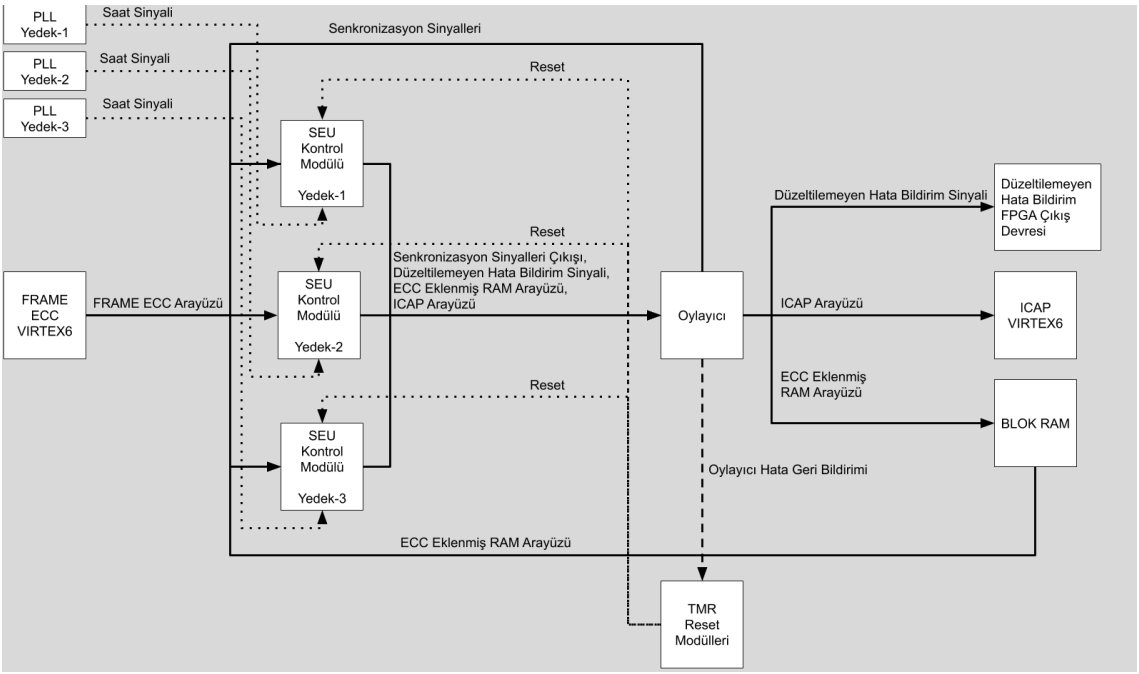
SEU kontrol modülü hatayı tespit edebilmek için ICAP_VIRTEX6 devresini kontrol ederek konfigürasyon hafızasını okur. Okunan konfigürasyon çerçevesini BLOK RAM'e kaydeder. ICAP_VIRTEX6'den okunan her bir çerçeveden sonra FRAME_ECC_VIRTEX6 devre çıkışını kontrol eder. FRAME_ECC_VIRTEX6 modülü kontrolü sonrasında hata yoksa bir sonraki çerçevenin okuma işlemini başlatır. Hata olması durumunda BLOK RAM'de tutulan konfigürasyon çerçevesinin ilgili bitini düzeltir ve ICAP üzerinden tekrardan yazar. BLOK RAM'e yazma işlemi Hamming kodu üretilerek, üretilen Hamming kodu ile birlikte yapılmıştır. BLOK RAM'den veriler okunurken Hamming kodu ile birlikte okunmuş ve RAM'de gerçekleşen SEU hataları düzeltilmiştir. Tasarlanan SEU kontrol modülünün alt blok mimarisi Şekil 4-1'de verilmiştir.



Şekil 4-1 SEU kontrol modülü mimarisi

ICAP_VIRTEX6 devresinin kontrol işlemi, alt bloklardan ICAP kontrol modülü tarafından gerçekleştirilir. FRAME_ECC_VIRTEX6 devre çıkışları kullanılarak hatanın gözlemlenmesi, düzeltilmesi ve ICAP kontrol modülünün kontrol işlemi FRAME_ECC kontrol modülü alt bloğu tarafından gerçekleştirilir. BLOK RAM'e yazılan veriler için Hamming kodu üretilmesi ve BLOK RAM okunurken gerçekleşen hataların düzeltilmesi işlemi ECC modülü tarafından gerçekleştirilmektedir.

SEU kontrol modülü kendi bit akış verisinde gerçekleşen hatalara karşılık TMR yapıda kullanılmıştır. PDR ile düzeltilen SEU kontrol modülünün diğer yedekleri ile senkronizasyonu diğer modüllerin başlattığı çerçevenin adresi ve okuma işleminin 1 saat çevrimi sonra gerçekleşeceği bilgisini veren sinyal kullanılarak yapılmıştır. Bu adres bilgisi oylayıcıdan geçerek diğer modüllere bağlanır. SEU kontrol modülünün TMR yapıda kullanılışı Şekil 4-2'de verilmiştir.



Şekil 4-2 SEU kontrol modülün TMR yapıda kullanılışı

ICAP_VIRTEX6 ve FRAME_ECC_VIRTEX6 devreleri birden fazla olamayacağından ötürü bu modüller ile bağlantının sağlandığı oylayıcıda ve RR'lerde gerçekleşecek hata yedeklenerek maskelenemeyecektir. Bu oylayıcıda ve RR'lerde gerçekleşecek olan hatanın ihtimali tezin beşinci bölümünde hesaplanmıştır.

4.3.1. ICAP Kontrol Modülü

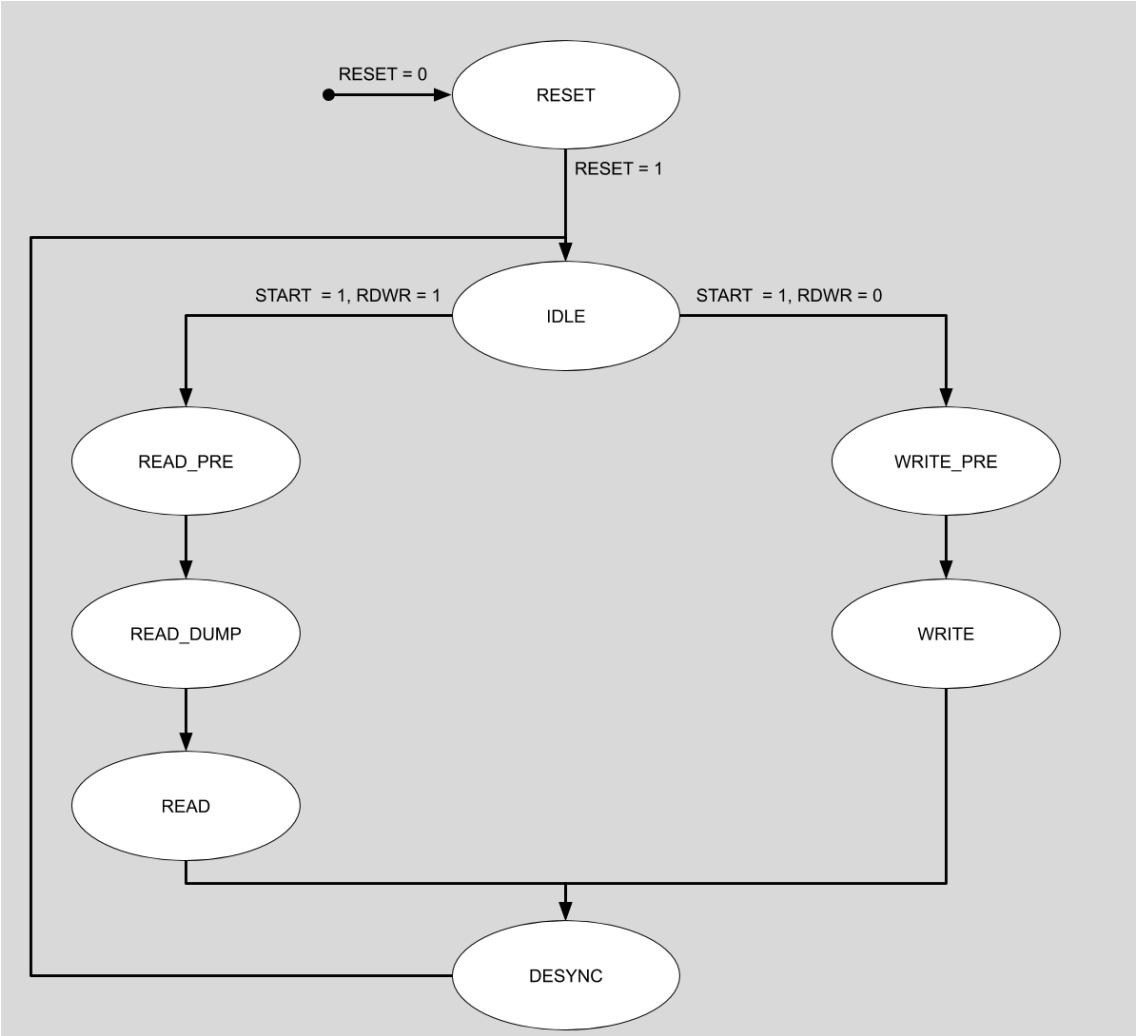
ICAP kontrol modülü SEU kontrol modülünün alt bloklarındandır. ICAP kontrol modülü ICAP_VIRTEX6 devresine yazma ve okuma işlemlerini gerçekleştirir ve okunan konfigürasyon belleği çerçeve verisini BLOK RAM'e yazar ve yazacağı konfigürasyon belleği çerçevesi verilerini BLOK RAM'den okur. ICAP kontrol modülü uydu olarak kullanılmış olup okuma ve yazma işlemleri giriş sinyalleri ile FRAME_ECC_VIRTEX6 kontrol modülü ile kontrol edilmiştir. ICAP kontrol modülü giriş çıkış sinyalleri Çizelge 4-6'da verilmiştir.

Çizelge 4-6 ICAP kontrol modülü giriş çıkış sinyalleri

Sinyal Adı	**Sinyal Tipi**	**Açıklama**
ICAP_CLK	Giriş	ICAP arayüzü ile aynı frekanstaki saatin 180 derece kaydırılmış halidir.
RESET	Giriş	ICAP kontrol modülü sıfırlama girişidir.
BUSY	Giriş	ICAP arayüzünden gelen BUSY sinyalidir.
O [31:0]	Giriş	ICAP arayüzünden gelen O sinyalidir.
CSB	Çıkış	ICAP arayüzüne bağlanan CSB sinyalidir.
RDWRB	Çıkış	ICAP arayüzüne bağlanan RDWRB sinyalidir.
I [31:0]	Çıkış	ICAP arayüzüne bağlanan I sinyalidir.
START	Giriş	Okuma ve yazma işlemlerini başlatan kontrol sinyalidir.
RDWR	Giriş	Başlatılan işlemin Okuma veya yazma olduğunu belirler. Başlatılan işlemin Okuma veya yazma olduğunu belirler. 1 = Okuma 0 = Yazma
FRAME_ADDRESS [31:0]	Giriş	Okuma ve yazma işlemlerinin hangi adresten yapılacağını kontrol eden sinyaldir.

ICAP_CONTROLLER_BUSY	Çıkış	En son başlatılan okuma ve yazma işleminin henüz bitmediğini gösterir. 1 = Meşgul 0 = Boşta
ECC_RAM_DATA_IN_B [31:0]	Giriş	2 girişli BLOK RAM arayüzünün B portundan gelen verinin ECC bloğu ile kontrolü yapılmış halidir.
ECC_RAM_DATA_OUT_B [31:0]	Çıkış	2 girişli BLOK RAM arayüzünün B portuna giden verinin ECC bloğu kodlaması öncesi halidir.
RAM_WR_EN_B	Çıkış	2 girişli BLOK RAM arayüzünün B portuna bağlanan yazma kontrol sinyalidir.
RAM _ADDR_B	Çıkış	2 girişli BLOK RAM arayüzünün B portuna bağlanan adres kontrol sinyalidir.

ICAP kontrol modülünde ICAP_VIRTEX6 devresi ile aynı frekanslı fakat 180 derece faz farklı saat girişleri kullanılmıştır. ICAP_VIRTEX6 devresi sinyalleri saatin yükselen kenarında sürdüğü ve örneklediği için ICAP kontrol modülü tüm sinyalleri saatin alçalan kenarında sürmeli ve örneklemelidir. Yükselen kenarda çalışabilmek için ICAP arayüzü ile 180 derece faz farklı sinyal kullanılmıştır. ICAP kontrol modülü sonlu durum makinası (FSM) diyagramı Şekil 4-3'de görülebilir.



Şekil 4-3 ICAP kontrol modülü FSM diyagramı

ICAP kontrol modülü, SEU kontrol modülünün bütün alt modülleri ile birlikte RESET giriş sinyali ile sıfırlanabilmektedir. RESET sinyali kullanılarak sıfırlanan modülde kullanılan tüm çıkışlar sıfırlanmaktadır. Sıfırlanma durumundan çıkan modül ilk önce IDLE durumuna geçer.

IDLE durumunda START giriş sinyali gözlemlenir. START giriş sinyalinde yükselen kenar görüldüğünde, yapılacak işlemin okuma veya yazma olduğunu belirten RDWR sinyali ile çerçeve adresini belirten FRAME_ADDRESS sinyali örneklenir ve ICAP_CONTROLLER_BUSY sinyali 1 sürülür. Eğer RDWR sinyal değeri 0'sa yazma işlemini başlatmak için WRITE_PRE durumuna gidilir. Eğer RDWR sinyal değeri 1 se okuma işlemini başlatmak için READ_PRE durumuna gidilir.

ICAP_VIRTEX6 devresi ile konfigürasyon belleği çerçevesi okuma ve yazma işleminin başlatılabilmesi ve düzgün bir biçimde bitirilebilmesi için okuma öncesi ve sonrasında ve yazma öncesi ve sonrasında bir dizi verinin ICAP_VIRTEX6 devresine gönderilmesi gereklidir [6, 25, 26, 27]. Bu veri dizileri EK-1'de verilmiştir.

WRITE_PRE durumunda EK-1'de verilen veriler ICAP_VIRTEX6 devresine gönderilir. Bu gönderme işlemi bittikten sonra konfigürasyon belleğine yazılacak verilerin gönderilmesi işlemi için WRITE durumuna gidilir.

WRITE durumunda BLOK RAM'de saklanan konfigürasyon belleği çerçevesi verileri okunup, ICAP_VIRTEX6 devresine gönderilerek konfigürasyon belleği çerçevesi tekrardan yazılır. Yazılacak olan verinin yazma işlemi bittikten sonra bir çerçeve uzunluğunda yapılmalıdır [6]. Yazma işlemi bittiğinde DESYNC durumuna gidilir.

READ_PRE durumunda EK-1'de verilen veriler ICAP_VIRTEX6 devresine gönderilir. Bu gönderme işlemi bittikten sonra konfigürasyon belleğinden okunan değerlerin saklanması işlemi için READ_DUMP durumuna gidilir. ICAP_VIRTEX6 okunacak anlamlı veriden sonra bir çerçeve uzunluğunda işlevsiz veri gönderir [6].

READ_DUMP durumunda ICAP_VIRTEX6 devresinin gönderdiği işlevsiz bir çerçeve uzunluğundaki verinin gönderilme işleminin tamamlanması beklenir. İşlevsiz verinin gönderilmesi tamamlandığında READ durumuna gidilir.

READ durumunda ICAP_VIRTEX6 devresinin gönderdiği konfigürasyon belleği çerçevesi verileri alınır ve BLOK RAM'e yazılır. Okuma işlemi bittiğinde DESYNC durumuna gidilir.
DESYNC durumunda EK-1'de verilen veriler ICAP_VIRTEX6 devresine gönderilir. Bu gönderme işlemi bittikten sonra ICAP_CONTROLLER_BUSY 0 sürülür ve IDLE durumuna dönülür.

4.3.2. FRAME_ECC Kontrol Modülü

FRAME_ECC kontrol modülü SEU kontrol modülünün alt bloklarındandır. FRAME_ECC Kontrol Modülü ile ICAP kontrol modülü kontrol edilerek okuma ve yazma işlemleri başlatılmıştır ve FRAME_ECC_VIRTEX6 devresi çıkışları gözlemlenmiştir. FRAME_ECC_VIRTEX6 çıkışları gözlemlenerek tespit edilen hataların düzeltilmesi için, BLOK RAM'den ilgili 32-bitlik veri okunmuş ve ilgili bitin değili alınarak BLOK RAM'e tekrar yazılmıştır. FRAME_ECC kontrol modülü giriş çıkış sinyalleri Çizelge 4-7'da verilmiştir.

Çizelge 4-7 FRAME_ECC kontrol modülü giriş çıkış sinyalleri

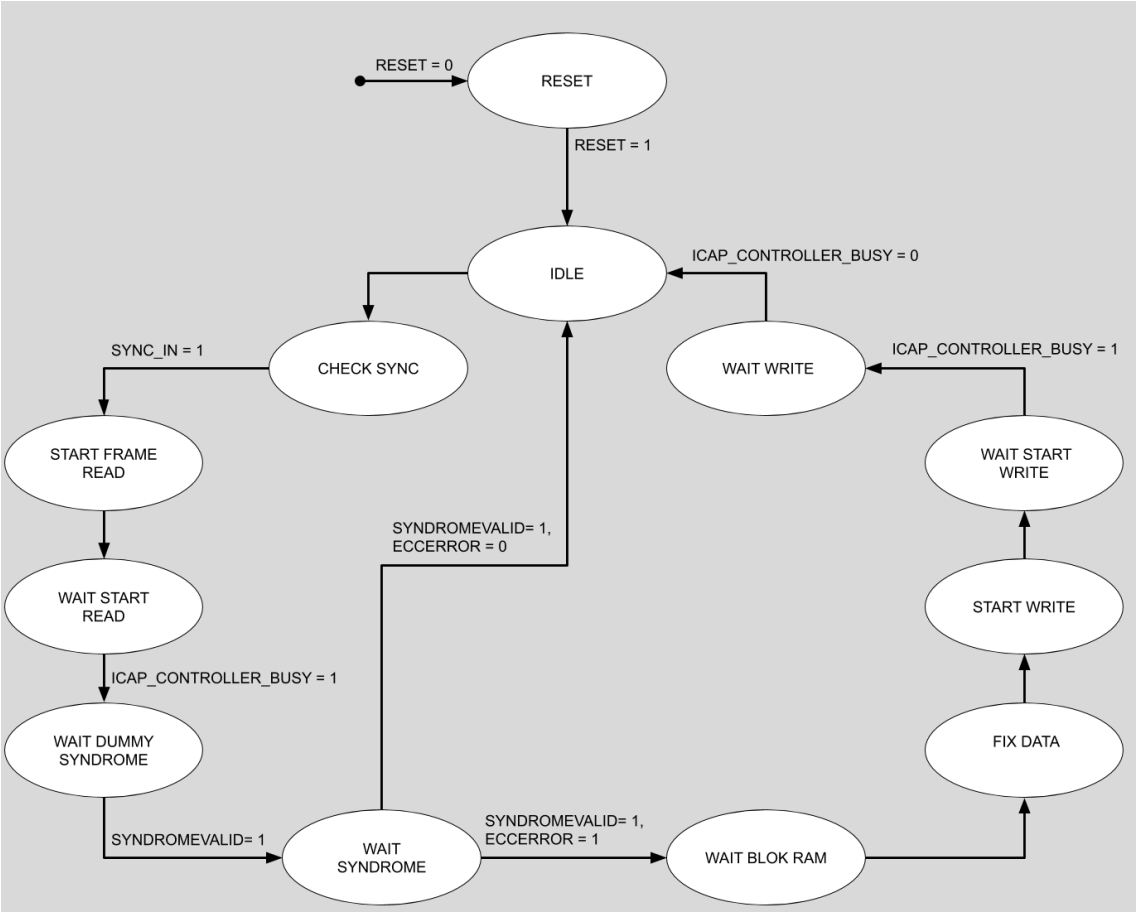
Sinyal Adı	**Sinyal Tipi**	**Açıklama**
ICAP_CLK	Giriş	ICAP arayüzü ile aynı frekanstaki saatin 180 derece kaydırılmış halidir.
RESET	Giriş	ICAP kontrol modülü sıfırlama girişidir.
ICAP_CONTROLLER_BUSY	Giriş	En son başlatılan okuma ve yazma işleminin henüz bitmediğini gösterir. 1 = Meşgul 0 = Boşta
START	Çıkış	Okuma ve yazma işlemlerini başlatan kontrol sinyalidir.

RDWR	Çıkış	Başlatılan işlemin Okuma veya yazma olduğunu belirler. Başlatılan işlemin Okuma veya yazma olduğunu belirler. 1 = Okuma 0 = Yazma
FRAME_ADDRESS [31:0]	Çıkış	Okuma ve yazma işlemlerinin hangi adresten yapılacağını kontrol eden sinyaldir.
ECC_RAM_DATA_IN_A [31:0]	Giriş	2 girişli BLOK RAM arayüzünün A portundan gelen verinin ECC bloğu ile kontrolü yapılmış halidir.
ECC_RAM_DATA_OUT_A [31:0]	Çıkış	2 girişli BLOK RAM arayüzünün A portuna giden verinin ECC bloğu kodlaması öncesi halidir.
RAM_WR_EN_A	Çıkış	2 girişli BLOK RAM arayüzünün A portuna bağlanan yazma kontrol sinyalidir.

RAM _ADDR_A	Çıkış	2 girişli BLOK RAM arayüzünün A portuna bağlanan adres kontrol sinyalidir.
SYNC_IN	Giriş	Senkronizasyon için kullanılan giriş sinyalidir.
SYNC_OUT	Çıkış	Senkronizasyon için kullanılan çıkış sinyalidir.
LAST_FRAME_ADDRESS_IN	Giriş	Senkronizasyon için kullanılan, sırada okunacak olan konfigürasyon belleği adresi giriş sinyalidir.
LAST_FRAME_ADDRESS_OUT	Çıkış	Senkronizasyon için kullanılan, sırada okunacak olan konfigürasyon belleği adresi çıkış sinyalidir.
REQUEST_FULL_RECONFIGURATION	Çıkış	Düzeltilemeyen hataları FPGA dışına bildirmekte kullanılan sinyaldir.
BLOCK_RAM_ERROR	Giriş	BLOK RAM'de 2-bit SEU oluştuğunu ECC modülü tarafından belirtmekte kullanılır.

SYNDROME [12:0]	Giriş	FRAME_ECC arayüzünden gelen SYNDROME sinyalidir.
SYNDROMEVALID	Giriş	FRAME_ECC arayüzünden gelen SYNDROMEVALID sinyalidir.
ECCERROR	Giriş	FRAME_ECC arayüzünden gelen ECCERROR sinyalidir.

FRAME_ECC kontrol modülünde ICAP kontrol modülü ile aynı frekanslı fakat 180 derece faz farklı saat girişleri kullanılmıştır. FRAME_ECC kontrol modülü için kullanılan FSM Şekil 4-4‘de verilmiştir.



Şekil 4-4 FRAME_ECC kontrol modülü FSM diyagramı

FRAME_ECC kontrol modülü, SEU kontrol modülünün bütün alt modülleri ile birlikte RESET giriş sinyali ile sıfırlanabilmektedir. RESET sinyali kullanılarak sıfırlanan modülde kullanılan tüm çıkışlar sıfırlanmaktadır. Sıfırlanma durumundan çıkan modül ilk önce IDLE durumuna geçer.

IDLE durumunda, SYNC_OUT sinyali 1 saat çevrimi süresince 1'e sürülür ve CHECK_SYNC durumuna gidilir.
CHECK_SYNC durumunda SYNC_IN sinyali gözlemlenir. Eğer değeri 1 okunursa LAST_FRAME_ADDRESS_IN sinyali örneklenir, SYNC_OUT sinyali 0'a sürülür ve START_FRAME_READ durumuna gidilir.

START_FRAME_READ durumunda START çıkış sinyali 1, RDWR sinyali 1, ve FRAME_ADDRESS sinyali LAST_FRAME_ADDRESS_IN sinyalinden örneklenen değere sürülür ve ICAP kontrol modülü okuma işlemi için tetiklenmiş olur. Tetikleme işleminden sonra WAIT_START_READ durumuna geçilir.

WAIT_START_READ durumunda ICAP_CONTROLLER_BUSY sinyali gözlemlenir. ICAP kontrol modülünün ICAP_CONTROLLER_BUSY sinyalini 1'e sürerek komutu aldığından emin olunduktan sonra WAIT_DUMMY_SYNDROME durumuna gidilir.

WAIT_DUMMY_SYNDROME durumunda okunmak istenen çerçeveden önce gelen, bir çerçeve uzunluğundaki verinin sonlanması beklenir ve WAIT_SYNDROME durumuna geçilir.

WAIT_SYNDROME durumunda SYNDROMEVALID sinyali gözlemlenir. SYNDROMEVALID sinyalinde yükselen kenar görülmesi ICAP okuma işleminin FRAME_ADDRESS sinyali ile belirtilen adresteki konfigürasyon belleği çerçevesinin okuma işleminin tamamlandığı anlamına gelmektedir. SYNDROMEVALID sinyalinde yükselen kenar görülmesi ile 1 saat periyodu öncesinde örneklenen SYNDROME verisi geçerlilik kazanır. Bu SYNDROME sinyalinin değerine göre yapılacak işlemler belirlenir. Eğer hiç hata görülmediyse LAST_FRAME_ADDRESS_OUT sinyali okunan konfigürasyon çerçevesi adresinin bir fazlasına sürülür ve IDLE durumuna geçilir. Eğer tek bitten daha fazla hata bulunuyorsa START_FULL_RECONFIGURATION sinyali 1 sürülür ve IDLE durumuna geçilir. START_FULL_RECONFIGURATION sinyali FPGA dışı bir kaynak ile bütün şekillendirme işleminin tetiklendiği sinyaldir. Eğer tek bitte hata bulunuyorsa, SYNDROME sinyali ile hatalı olduğu belirtilen bitin yeri tespit edilir ve ilgili 32-bitin BLOK RAM'den okunması için BLOK RAM adresi sürülerek WAIT_BLOK_RAM durumuna geçilir.

BLOK RAM adresi sürüldükten sonra verinin hazır olması 2 saat çevrimi sürmektedir. WAIT_BLOK_RAM durumunda 2 saat çevrimi beklendikten sonra FIX_DATA durumuna geçilir.

FIX_DATA durumunda BLOK RAM'den okunan 32-bitlik verinin SYNBIT sinyalinde belirtilen ilgili biti düzeltilerek aynı adrese tekrar yazılır ve START_WRITE durumuna geçilir.

START_WRITE durumunda ICAP kontrol modülü yazma işlemi için tetiklenir. ICAP kontrol modülü tetiklendikten sonra WAIT_START_WRITE durumuna geçilir.

WAIT_START_WRITE durumunda ICAP_CONTROLLER_BUSY sinyali gözlemlenir. ICAP kontrol modülünün ICAP_CONTROLLER_BUSY sinyalini 1'e sürerek komutu aldığından emin olunduktan sonra WAIT_WRITE durumuna gidilir.

WAIT_WRITE durumunda ICAP_CONTROLLER_BUSY sinyali gözlemlenir ve ICAP kontrol modülünün yazma işlemini bitirmesi beklenir. BLOCK_RAM_ERROR sinyalinde 1 görüldüğünde START_FULL_RECONFIGURATION sinyali 1 sürülerek IDLE durumuna geçilir. BLOCK_RAM_ERROR sinyalinde 1 görülmez ve ICAP_CONTROLLER_BUSY sinyalinde düşen kenar görülür ise IDLE durumuna geçilir.

4.3.3. ECC Modülü

ECC modülü SEU kontrol modülünün alt bloklarındandır. BLOK RAM'e yazılacak veriler için Hamming kodları üretmek ve BLOK RAM'den okunan sinyallerde bulunan 1-bitlik hataları Hamming kodları ile düzeltmek için kullanılmıştır. BLOK RAM'e giden sinyaller bir saat çevrimi bekletilir ve bu 1 saat çevrimi süresince Hamming kodları üretilir. Üretilen 7-bit Hamming kodları ile birlikte 32-bitlik konfigürasyon belleği çerçevesi verisi 39 bit olarak BLOK RAM'e yazılır. BLOK RAM'den gelen sinyaller de bir saat çevrimi bekletilir. Bu sürede gelen sinyal içerisindeki 7-bitlik Hamming kodları ile 32-bitlik konfigürasyon belleği çerçevesi verisi kontrol edilir. Eğer hata yoksa direk okuyan bloğa gönderilir. Eğer hata varsa ve 1-bitse düzeltilerek gönderilir. Eğer hata 1-bitten fazla sayıda görülmüşse BLOCK_RAM_ERROR sinyali 1 sürülür.

4.3.4. SEU Kontrol Giriş Oylayıcı Modülü

SEU Kontrol Giriş Oylayıcı Modülü, SEU Kontrol Modülüne gelen oylanması gereken giriş sinyallerinin oylanması için kullanılmıştır. Oylama işlemi asenkron, saat kullanılmadan yapılmıştır. Senkronizasyon sinyalleri SYNC_IN_1,

SYNC_IN_2, SYNC_IN_3 ve sıfırlama sinyalleri RESET_1, RESET_2 ve RESET_3 oylanmıştır. Oylama sonucunda RESET sinyali ve SYNC_IN sinyalleri üretilmiştir. Hata maskeleme yönteminde önerilen modül öncesi oylayıcı tasarım kolaylığı için modül içine taşınmıştır. Eğer gelen sinyallerden birisinde farklılık varsa ilgili sinyali üreten modülün sıfırlanması için RESET kontrol modülüne gönderilmek üzere MODULE_NUMBER sinyali ilgili modülün numaralandırılmış değerine atanır.

4.4. BLOK RAM

BLOK RAM bloğu XILINX IP GENERATION TOOL programı kullanılarak üretilmiştir. Tek saat girişli ve 2 okuma ve 2 yazma kanallı olarak yaratılmıştır. BLOK RAM veri genişliği 39 bit veri derinliği ise 128 veri olarak seçilmiştir. Bu yapı ile BLOK RAM'e veriler Hamming kodu ile yazılabilecek ve BLOK RAM bir konfigürasyon çerçevesini tutabilecektir. Okuma yazma kanallarının A portu FRAME_ECC kontrol bloğu tarafından kontrol edilmektedir. Diğer okuma yazma kanalı olan B portu ise ICAP kontrol modülü tarafından kontrol edilmektedir. Kanallar birbirinden bağımsız okuma yazma yapabilmektedir.

4.5. RESET Kontrol Modülü

RESET kontrol modülü tüm blokları ayrı ayrı sıfırlamak için kullanılmıştır. Kullanılan tüm PLL bloklarının LOCKED sinyalleri kullanılarak her bloğun kullandığı saat ile senkron sıfırlama sinyalleri üretilmiştir. Blok aracılığı ile asenkron sıfırlama sinyallerinin senkronizasyonu sağlanmıştır. RESET kontrol modülünün bir diğer görevi ise oylayıcı çıkışında senkronizasyon bozulması tespit edildiğinde, bozuk modülün sıfırlanmasıdır. Her modül arası oylayıcı, oylanan sinyallerde farklılık bulunması durumunda, ilgili sinyali üreten modülün numarasını RESET Kontrol modülüne belirtmektedir. Bu sinyal aracılığı ile RESET Kontrol modülü ilgili modülü hata düzelene kadar sıfırlamaktadır. RESET kontrol modülü giriş çıkış sinyalleri Çizelge 4-8'de verilmiştir.

Çizelge 4-8 RESET kontrol modülü giriş çıkış sinyalleri

Sinyal Adı	**Sinyal Tipi**	**Açıklama**
CLK_1	Giriş	Birinci saat gurubunun saat girişidir. Senkron RESET sinyali üretmek için kullanılmıştır.
CLK_2	Giriş	İkinci saat gurubunun saat girişidir. Senkron RESET sinyali üretmek için kullanılmıştır.
CLK_3	Giriş	Üçüncü saat gurubunun saat girişidir. Senkron RESET sinyali üretmek için kullanılmıştır.
CLK_USED	Giriş	RESET Kontrol Modülü devrelerinin kullandığı saat girişidir.
LOCKED_1	Giriş	Birinci saat gurubunun PLL LOCKED sinyali girişidir. PLL'nin saati üretmeye başladığını gösterir.
LOCKED_2	Giriş	İkinci saat gurubunun PLL LOCKED sinyali girişidir. PLL'nin saati üretmeye başladığını gösterir.
LOCKED_3	Giriş	Üçüncü saat gurubunun PLL LOCKED sinyali girişidir. PLL'nin saati üretmeye başladığını gösterir.

MODULE_NUMBER_1 [3:0]	Giriş	SEU modülü YEDEK-1 içerisinde bulunan oylayıcıdan gelen hata bildirimidir. Sıfırdan farklı ise ilgili modülün sıfırlanması gerektiğini bildirir.
MODULE_NUMBER_2 [3:0]	Giriş	SEU modülü YEDEK-2 içerisinde bulunan oylayıcıdan gelen hata bildirimidir. Sıfırdan farklı ise ilgili modülün sıfırlanması gerektiğini bildirir.
MODULE_NUMBER_3 [3:0]	Giriş	SEU modülü YEDEK-3 içerisinde bulunan oylayıcıdan gelen hata bildirimidir. Sıfırdan farklı ise ilgili modülün sıfırlanması gerektiğini bildirir.
MODULE_NUMBER_4 [3:0]	Giriş	ARINC-429 modülü YEDEK-1 içerisinde bulunan oylayıcıdan gelen hata bildirimidir. Sıfırdan farklı ise ilgili modülün sıfırlanması gerektiğini bildirir.
MODULE_NUMBER_5 [3:0]	Giriş	ARINC-429 modülü YEDEK-2 içerisinde bulunan oylayıcıdan gelen hata bildirimidir. Sıfırdan farklı ise ilgili modülün sıfırlanması gerektiğini bildirir.
MODULE_NUMBER_6 [3:0]	Giriş	ARINC-429 modülü YEDEK-3 içerisinde bulunan oylayıcıdan gelen hata bildirimidir. Sıfırdan farklı ise ilgili modülün sıfırlanması gerektiğini bildirir.
RESET_SEU_1	Çıkış	SEU Modülü YEDEK-1'in sıfırlama sinyalidir.

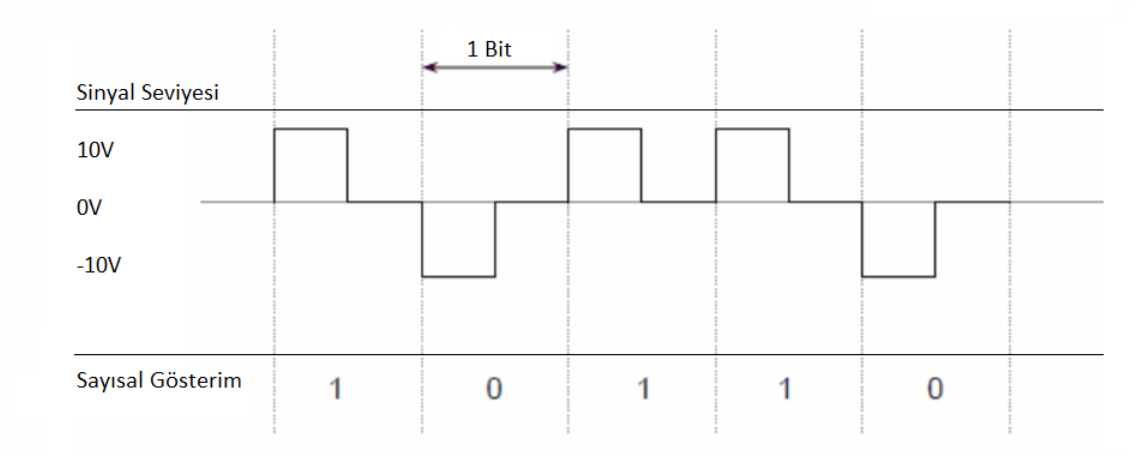
RESET_SEU_2	Çıkış	SEU Modülü YEDEK-2'in sıfırlama sinyalidir.
RESET_SEU_3	Çıkış	SEU Modülü YEDEK-3'in sıfırlama sinyalidir.
RESET_ARINC_429_1	Çıkış	ARINC-429 Modülü YEDEK-1'in sıfırlama sinyalidir.
RESET_ARINC_429_2	Çıkış	ARINC-429 Modülü YEDEK-2'in sıfırlama sinyalidir.
RESET_ARINC_429_3	Çıkış	ARINC-429 Modülü YEDEK-3'in sıfırlama sinyalidir.

4.6. ARINC-429 Modülü

ARINC-429 1978 yılından beri aviyonik sistemlerde yaygın olarak kullanılan veri yolu protokolüdür. ARINC-429 protokolü veri yolunun fiziksel özelliklerini, adresleme mekanizmasını, veri etiketleme metodunu ve veri paket formatını tanımlar [28].

ARINC-429 veri yolunda transferler alıcı ve verici iki ayrı kanal üzerinden çift yönlü yapılabilmektedir. ARINC-429 veri yolu iki cihazı noktadan noktaya bağlamak için kullanılır. Veri yolunda paketler 32-bit den oluşur. Bu 32-bitlik paketin 24-biti veri, 8-biti ise veriyi etiketlemek için kullanılır. Veri 2 farklı frekansta 100kHz ve 12,5kHz de gönderilebilir fakat bir veri yolu için sadece bir frekans kullanılabilir. Hızlar arası geçiş yoktur. Birbiri ardına gönderilen paketler 4 bitlik süre ile ayrılır. Anlamlı bir veri olmasa bile paket transferi devamlı olarak gönderilir. Alıcı paketler arası ayrılan 4 bitlik süreyi kullanarak verinin başlangıç bitini tespit eder ve kaynağa senkronize olur. Ayrı olarak saat sinyali taşınmaz. Transferde her bit Sıfıra-Dönüş (RZ) modülasyonu ile gönderilir [28, 29, 30, 31].

RZ modülasyonu sayısal değeri FPGA tarafından oluşturulmakta fiziksel sinyal seviyesine ise ARINC-429 entegresi tarafından dönüştürülmektedir. RZ modülasyonu ve sayısal değerleri Şekil 4-5'de görülebilir.



Şekil 4-5 RZ modülasyonu ve dijital değerleri

32-bitlik ARINC-429 paketi 5 alandan oluşmaktadır. Şekil 4-6'da paket yapısı verilmiştir [28, 29, 30, 31].

32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
P	SSM		MSB								Data							LSB				SDI		Label								

Şekil 4-6 ARINC-429 paket yapısı

32. bit parite (P) bitidir. ARINC-429 paketleri, tek parite biti kullanılarak, alıcı tarafın verinin bütünlüğünü kontrol etmesi sağlanır [28, 29, 30, 31].
Fakat P biti hatanı yerini tespit edemez ve düzeltemez.

31:30 bit alanı İşaret/Durum Matrisi (SSM) olarak adlandırılır [28, 29, 30, 31]. Pakette kullanılan etikete göre farklı anlamlar alabilir. Bu alanın kullanımı zorunlu değildir. Kullanılmadığı durumda 00 gönderilir. Verinin içeriği ikili kodlu ondalık (BCD) sayı ile ifade ediliyorsa SSM alanının kullanılabileceği anlamlar Çizelge 4-9'de verilmiştir. Veri ikili sayı gösterimi (BNR) ile ifade ediliyorsa SSM alanı içeriği Çizelge 4-10'de belirtilen anlamlarda kullanılabilir. Eğer veri içeriği ayrık bitlerden oluşuyorsa SSM alanı ile ifade edilebilen anlamlar Çizelge 4-11'de verilmiştir.

Çizelge 4-9 BCD sayılar için SSM anlamları

Bit 31:30	Çözümlenen Anlamlar
00	Artı, Kuzey, Doğu, Sağ Yön, Varış Noktası, Yukarı
01	Anlamsız Data
10	Fonksiyonel Test
11	Eksi, Güney, Batı, Sol, Başlangıç Noktası, Aşağı

Çizelge 4-10 BNR sayılar için SSM anlamları

Bit 31:30	Çözümlenen Anlamlar
00	Hata Uyarısı
01	Anlamsız Data
10	Fonksiyonel Test
11	Normal Operasyon

Çizelge 4-11 Ayrık sayılar için SSM anlamları

Bit 31:30	Çözümlenen Anlamlar
00	Normal Operasyon, Doğrulanmış Data
01	Anlamsız Data
10	Fonksiyonel Test
11	Hata Uyarısı

29:11 bit alanı ARINC-429 ile gönderilmek istenen veri için kullanılır ve Data alanı olarak adlandırılır. Bu verini 29. biti en anlamlı bit (MSB) olarak, 11. biti ise en az anlamlı bit (LSB) olarak kullanılır [28, 29, 30, 31].

Veri formatı BNR ise 29. bitin alabileceği anlamlar Çizelge 4-12'de verilmiştir. Verinin formatları Çizelge 4-13'de verilmiştir.

Çizelge 4-12 BNR sayılar için 29. bit anlamları

Bit 29	Çözümlenen Anlamlar
0	Artı, Kuzey, Doğu, Sağ Yön, Varış Noktası, Yukarı
1	Eksi, Güney, Batı, Sol, Başlangıç Noktası, Aşağı

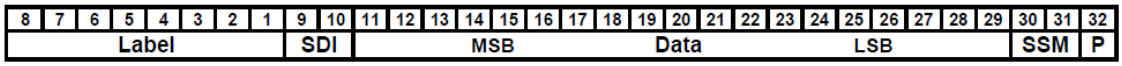
Çizelge 4-13 ARINC-429 veri formatları

Format	Açıklama
BNR	İkiye tümler gösterim biçimi
BCD	ISO alfabesi no-5 nümerik alt kümesi
Ayrık	BCD, BNR kombinasyonu veya ayrık bit gösterimi
Bakım Verisi ve Alındı Bilgisi	İki yönlü haberleşme gerektirir.
Williamsburg/Buckhorn Protokolü	İki yönlü haberleşme gerektirir.

Bakım Verisi ve Alındı bilgisi için iki yönlü haberleşme gerekir ve genellikle Williamsburg/Buckhorn Protokolü kullanılır. Williamsburg/Buckhorn Protokolünde kaynak ve hedef arasında transfer tokalaşma ile başlar. Kaynak Gönderme İzni (RTS) mesajını göndererek başlar. Ardından eğer uygunsa hedef Gönderim Serbest (CTS) mesajını gönderir. Williamsburg/Buckhorn Protokolünde dosyalar, bağlı veri birimleri (LDU) olarak gönderilir. LDU uzunluğu 3 ARINC-429 mesajından 255 ARINC-429 mesajına kadar olabilir. LDU, Transfer Başlangıcı (SOT) ile başlar. SOT mesajı, dosya numarası, LDU sıra numarası ve genel format tanımlayıcı (GFI) alanlarını içerir. SOT mesajını sayısı 253 kadar olabilen ARINC-429 veri sözcükleri takip eder. Veri sözcüklerinin gönderimi tamamlandığında son LDU mesajı olan Transfer Bitişi (EOT) gönderilir. EOT mesajı CRC ve LDU'nun gönderilen dosya içerisindeki konumu alanlarını içerir. Hedef EOT mesajından sonra alındı bilgisi (ACK) mesajını gönderir ve LDU transferi tamamlanır [31].

10:9 bit alanı Kaynak/Hedef Tanımlayıcı (SDI) olarak adlandırılır [31]. Verinin hangi kaynak tarafından veya hangi hedefe gönderilmek istendiğini tanımlar. SDI alanı kullanması zorunlu değildir. Kullanılmadığı durumda veri alanını genişletmek için kullanılabilir.

8:1 bit alanı Etiket (LABEL) olarak adlandırılır ve verinin format bilgisini, cihaz kimliğini veya verinin ne tarz bir bilgi içerdiğini belirtmek için kullanılır [31]. LABEL alanı ARINC-429 paketinde gönderilmesi zorunlu alanlardandır.
ARINC-429 paketinde önce etiket (LABEL) alanı önce MSB bitinden başlanarak gönderilir. LABEL harici paketin diğer bitleri LSB önce olmak üzere gönderilir [31]. Paket gönderim sırası Şekil 4-7'de verilmiştir.



Şekil 4-7 ARINC-429 paket alanları gönderim sırası

Fiziksel katmanda RZ modülasyonu için HI-8596 entegresi, RZ de-modülasyonu için HI8591 entegresi seçilmiştir. ARINC-429 modülü, HI-8596 ve HI8591 entegresi ile birlikte ARINC-429 protokolüne uygun olarak veri gönderip almaktadır [32, 33].

HI-8596 entegresi Tx sinyal sürücü olarak kullanılmaktadır. HI-8596 entegresi bit değerlerini, RZ modülasyonu ile -10V, 0V ve 10V 3 farklı değerde gönderebilmek için 2-bitlik giriş sinyaline ihtiyaç duymaktadır [32]. HI-8596 entegresi 2-bitlik giriş sinyali değerlerine karşılık gelen RZ modülasyonu voltaj seviyeleri Çizelge 4-14'de verilmektedir.

Çizelge 4-14 HI-8596 entegresi RZ voltaj seviyeleri

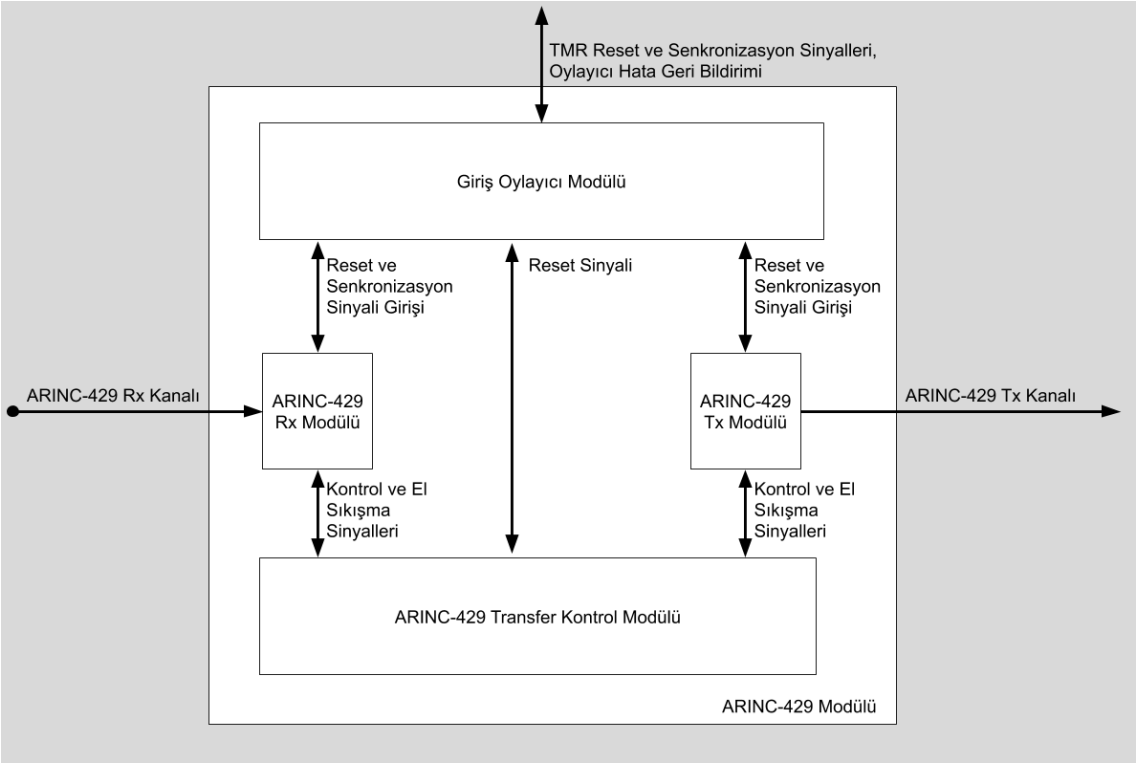
TX_1	TX_0	RZ Voltaj Seviyesi
0	0	0
0	1	-10
1	0	+10
1	1	Yüksek Empedans, Veri hattı kullanımda değil.

Çizelge 4-15 HI-8591 entegresi RZ voltaj seviyeleri

RX_1	RX_0	RZ Voltaj Seviyesi
0	0	0
0	1	-10
1	0	+10
1	1	Yüksek Empedans, Veri hattı kullanımda değil.

HI-8591 entegresi Rx sinyal alıcı olarak kullanılmaktadır. HI-8591 entegresi RZ modülasyondaki -10V, 0V ve 10V, 3 farklı değeri, 2-bitlik çıkış sinyali ile ifade etmektedir [33]. RZ modülasyonu voltaj seviyelerine karşılık gelen 2-bitlik çıkış sinyali değerleri Çizelge 4-15'de verilmektedir.

ARINC-429 modülü ARINC-429 Tx, ARINC-429 Rx, ARINC-429 Transfer Kontrol, ARINC-429 Giriş Oylayıcı alt modüllerinden oluşmaktadır. ARINC-429 blok mimarisi Şekil 4-8'de verilmiştir. ARINC-429 modülü, SEU durumunda sistemin hata toleransı test edilmek için Rx kanalından alınan veriyi, Tx modülünden geri gönderecek şekilde tasarlanmıştır.



Şekil 4-8 ARINC-429 modülü mimarisi

4.6.1. ARINC-429 Tx Modülü

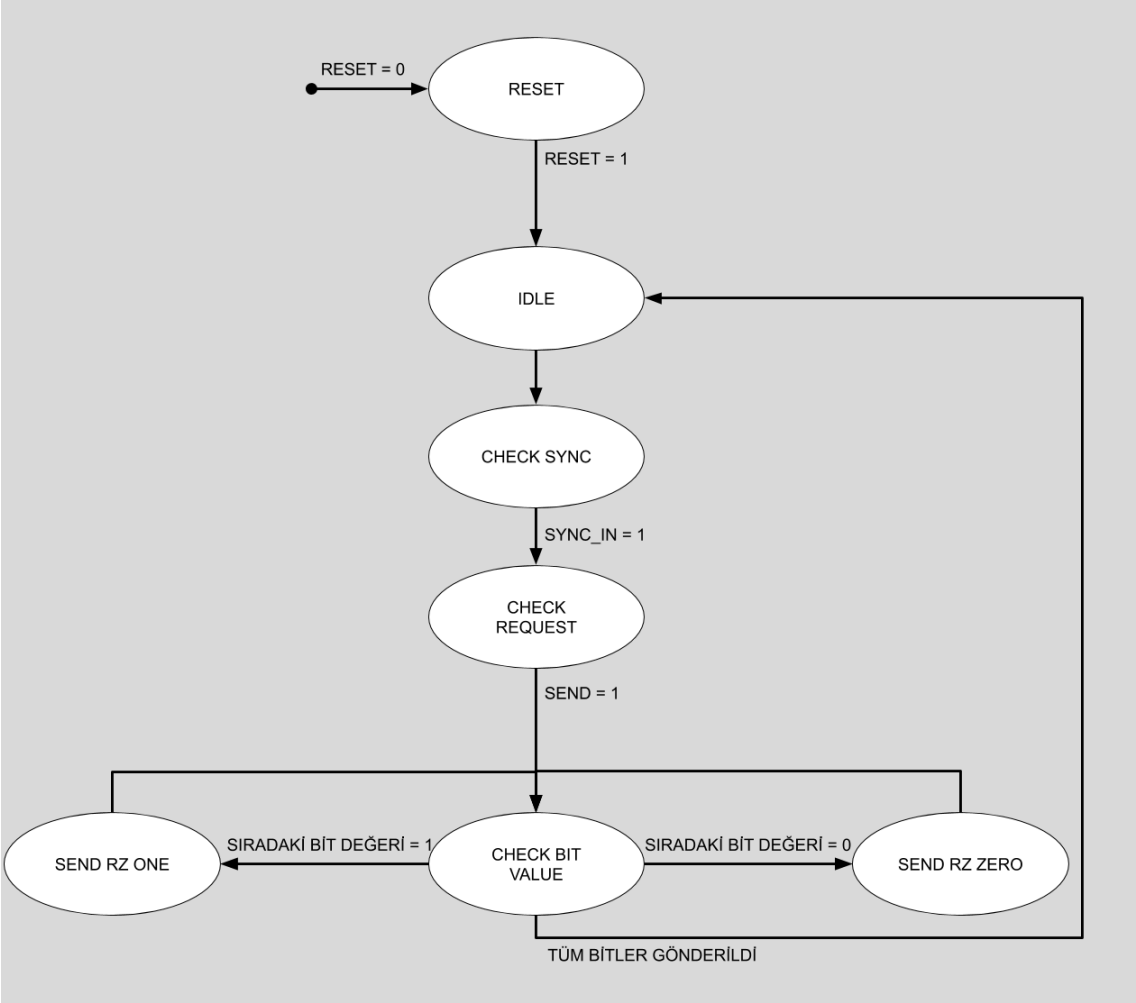
ARINC-429 Tx Modülü, ARINC-429 modülü alt bloklarındandır. ARINC-429 Tx Modülü, ARINC-429 mesajlarını göndermek için kullanılmıştır. ARINC-429 bit gönderim hızı olarak 100kHz seçilmiştir. HI-8596 entegresi hız seçim girişine bağlanan TRANSMIT_SPEED sinyali sabit 0 sürülerek, entegrenin 100KHz frekansına uygun zamanlamada sinyal üretmesi sağlanmıştır. ARINC-429 Transfer Kontrol bloğu tarafından uydu yapıda kullanılmaktadır. ARINC-429 Tx Modülü giriş çıkış sinyalleri Çizelge 4-16'da verilmiştir.

Çizelge 4-16 ARINC-429 Tx Modülü giriş ve çıkış sinyalleri

Sinyal Adı	**Sinyal Tipi**	**Açıklama**
CLK	Giriş	ICAP arayüzü ile aynı frekanstaki saatin 180 derece kaydırılmış halidir.
RESET	Giriş	ARINC-429 Tx Modülü Reset girişidir.
SYNC_TX_IN	Giriş	Senkronizasyon için kullanılan giriş sinyalidir.
SYNC_TX_OUT	Çıkış	Senkronizasyon için kullanılan çıkış sinyalidir.
SEND	Giriş	ARINC-429 Tx Modülü gönderme işlemini başlatan kontrol sinyalidir.
MESSAGE [31:0].	Giriş	Gönderilecek olan veri.
REQUEST_RECEIVED	Çıkış	Gönder komutunun alındığı bilgisini taşıyan el sıkışma sinyalidir.
TX_0	Çıkış	ARINC-429 Tx kanalıdır.
Tx_1	Çıkış	ARINC-429 Tx kanalıdır.
TRANSMIT_SPEED	Çıkış	ARINC-429 Transfer hızını entegreye bildiren sinyaldir.

ARINC-429 Tx Modülü sistemle aynı, 100Mhz saat frekansında çalışmaktadır. 100 kHz bit gönderim hızı, 100Mhz frekanslı saat sinyali sayılarak elde edilmiştir.

ARINC-429 Tx Modülü, tüm ARINC-429 modülleri ile birlikte RESET giriş sinyali ile sıfırlanabilmektedir. Sıfırlanma durumunda 2-bit ARINC-429 çıkış sinyalleri Tx_1 ve TX_0, 11 sürülmüştür. Sıfırlanma durumundan çıkan modül ilk önce IDLE durumuna gider. ARINC-429 Tx Modülünde kullanılan durum diyagramı Şekil 4-9‘da verilmiştir.



Şekil 4-9 ARINC-429 Tx Modülü FSM diyagramı

IDLE durumunda SYNC_TX_OUT sinyali 1'e sürülür ve CHECK_SYNC durumuna gidilir.

CHECK_SYNC durumunda SYNC_TX_IN sinyali kontrol edilir. Eğer 1 okunursa CHECK_REQUEST durumuna gidilir.

CHECK_REQUEST durumunda 4-bit ARINC-429 transferi süresince, paketler arası beklenilmektedir. Bu sürede 2-bit ARINC-429 çıkış sinyalleri Tx_1 ve TX_0 NULL evresinin bir parçası olarak 00 sürülmektedir. CHECK_REQUEST durumunda SEND bayrağı kontrol edilir. Eğer SEND 1 ise MESSAGE sinyali örneklenir ve ARINC429_PACKET sinyaline atanır. MESSAGE sinyalinin örneklendiği bilgisini ARINC-429 Transfer Kontrol bloğuna bildirmek için

REQUEST_RECEIVED sinyalini bir saat çevrimi süresince 1'e çeker. Eğer SEND 0 ise gönderilecek veri yoktur. CHECK_REQUEST durumunda eğer SEND 1 ise 3998 saat çevrimi bekleme süresi bittiğinde CHECK_BIT_VALUE durumuna gidilir.

CHECK_BIT_VALUE durumunda ARINC429_PACKET sinyali bit değeri kontrol edilir. Eğer değer 1 ise SEND_RZ_ONE durumuna gidilir. Eğer değer 0 ise SEND_RZ_ZERO durumuna gidilir. Eğer tüm bitler gönderildiyse IDLE durumuna gidilir. CHECK_BIT_VALUE durumunda 2-bit ARINC-429 çıkış sinyalleri Tx_1 ve TX_0, NULL sinyal seviyesinin bir parçası olarak 0 sürülmektedir.
SEND_RZ_ONE durumunda 1 değeri RZ modülasyonu ile 100kHz bit frekansında gönderilir. Bu işlem için sayaç kullanılır. 100Mhz saat frekanslı sistemde 2-bit ARINC-429 çıkış sinyalleri Tx_1 ve TX_0, 500 saat çevrimi 10 değeri, 499 saat çevrimi 00 değeri gönderilir ve CHECK_BIT_VALUE durumuna gidilir.

SEND_RZ_ZERO durumunda 0 değeri RZ modülasyonu sile 100kHz bit frekansında gönderilir. Bu işlem için sayaç kullanılır. 100Mhz saat frekanslı sistemde 2-bit ARINC-429 çıkış sinyalleri Tx_1 ve TX_0, 500 saat çevrimi 01 değeri, 499 saat çevrimi 00 değeri gönderilir ve CHECK_BIT_VALUE durumuna dönülür.

4.6.2. ARINC-429 Rx Modülü

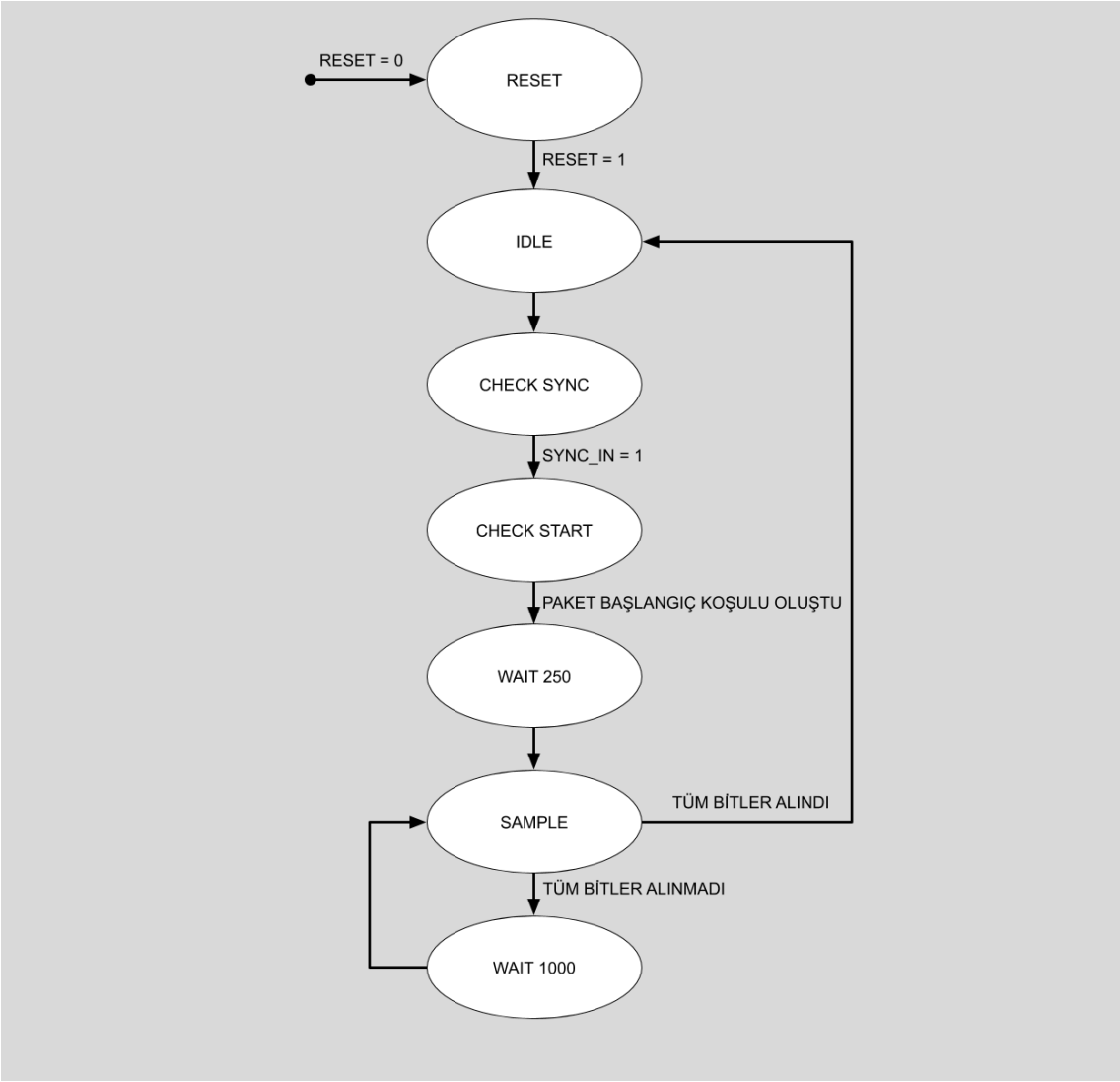
ARINC-429 Rx Modülü, ARINC-429 modülü alt bloklarındandır. ARINC-429 Rx Modülü, ARINC-429 mesajlarını almak için kullanılmıştır. ARINC-429 bit alma hızı olarak 100kHz seçilmiştir. ARINC-429 Transfer Kontrol bloğuna aldığı mesajları iletmektedir. ARINC-429 Rx Modülü giriş çıkış sinyalleri Çizelge 4-17'de verilmiştir.

Çizelge 4-17 ARINC-429 Rx Modülü giriş ve çıkış sinyalleri

Sinyal Adı	**Sinyal Tipi**	**Açıklama**
CLK	Giriş	ICAP arayüzü ile aynı frekanstaki saatin 180 derece kaydırılmış halidir.
RESET	Giriş	ARINC-429 Rx Modülü sıfırlama girişidir.
SYNC_RX_IN	Giriş	Senkronizasyon için kullanılan giriş sinyalidir.
SYNC_RX_OUT	Çıkış	Senkronizasyon için kullanılan çıkış sinyalidir.
DATA_RECEIVED	Çıkış	ARINC-429 Tx Modülü gönderme işlemini başlatan kontrol sinyalidir.
RECEIVED_PACKET [31:0]	Çıkış	Gönderilecek olan veri.
RX_0	Giriş	ARINC-429 Tx kanalıdır.
RX_1	Giriş	ARINC-429 Tx kanalıdır.
TRANSMIT_SPEED	Giriş	ARINC-429 Transfer hızını entegrenin bildirdiği sinyaldir.

ARINC-429 Rx Modülü sistemle aynı, 100Mhz saat frekansında çalışmaktadır. 100 kHz bit alma hızı, 100Mhz frekanslı saat sinyali sayılarak elde edilmiştir.

ARINC-429 Rx Modülü, tüm ARINC-429 modülleri ile birlikte RESET giriş sinyali ile sıfırlanabilmektedir. Sıfırlanma durumundan çıkan modül ilk önce IDLE durumuna gider. ARINC-429 Rx Modülünde kullanılan durum diyagramı Şekil 4-10‘da verilmiştir.



Şekil 4-10 ARINC-429 Rx Modülü FSM diyagramı

IDLE durumunda SYNC_RX_OUT sinyali 1'e sürülür ve CHECK_SYNC durumuna gidilir.

CHECK_SYNC durumunda SYNC_RX_IN sinyali kontrol edilir. Eğer 1 okunursa CHECK_REQUEST durumuna gidilir.

CHECK_START durumunda 2-bit RX_1 ve RX_0 ARINC-429 giriş sinyalleri gözlemlenmektedir. 2-bit RX_1 ve RX_0 ARINC-429 giriş sinyalleri en az 2000 saat çevrimi boyunca 00 görüldükten sonra gelen ilk 10 veya 01 değeri paket başlangıcını göstermektedir. Paket başlangıç koşulu oluştuktan sonra sinyalleri tam orta noktasından örnekleyebilmek için 250 saat çevrimi beklenmek üzere ve

WAIT_250 durumuna geçilir. DATA_RECEIVED sinyali CHECK_ START durumunda 0 sürülür.

WAIT_250 durumunda 250 saat çevrimi süresince beklenir ve SAMPLE durumuna geçilir.

SAMPLE durumunda gelen RX_1 ve RX_0 değeri 10 ise ilgili paket biti 1 olarak kaydedilir ve paketin tüm bitleri alınmadıysa WAIT_1000 durumuna gidilir. Gelen RX_1 ve RX_0 değeri 01 ise ilgili paket biti 0 olarak kaydedilir ve paketin tüm bitleri alınmadıysa WAIT_1000 durumuna gidilir. Diğer durumlar hata durumu olarak görülüp alınan veriler yok sayılarak CHECK_START durumuna dönülür. Eğer paketin tüm bitleri alındıysa DATA_RECEIVED sinyali 1 sürülür, kaydedilen paket değeri RECEIVED_PACKET sinyaline yazılır ve IDLE durumuna dönülür.

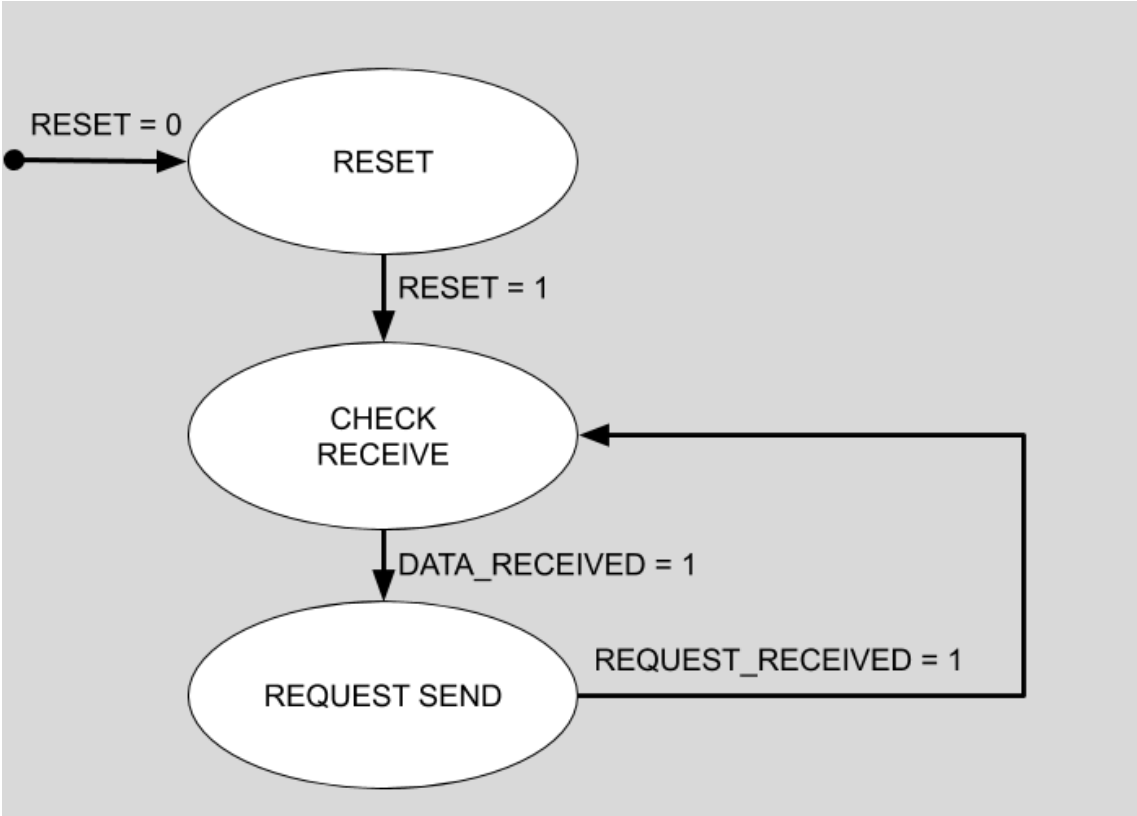
WAIT_1000 durumunda 1000 saat çevrimi süresince beklenir ve SAMPLE durumuna geçilir.

4.6.3. ARINC-429 Transfer Kontrol Modülü

ARINC-429 Transfer Kontrol Modülü, ARINC-429 modülü alt bloklarındandır. ARINC-429 Transfer Kontrol Modülü, ARINC-429 Rx modülü tarafından alınan mesajın, ARINC-429 Tx modülü ile gönderilmesi işlemini kontrol eder.

ARINC-429 Transfer Kontrol Modülü sistemle aynı, 100Mhz saat frekansında çalışmaktadır.

ARINC-429 Transfer Kontrol Modülü, tüm ARINC-429 modülleri ile birlikte RESET giriş sinyali ile sıfırlanabilmektedir. Sıfırlanma durumundan çıkan modül ilk önce CHECK_RECEIVE durumuna gider. ARINC-429 Transfer Kontrol Modülünde kullanılan durum diyagramı Şekil 4-11‘de verilmiştir.



Şekil 4-11 ARINC-429 Kontrol Modülü FSM diyagramı

CHECK_RECEIVE durumunda ARINC-429 Rx modülü tarafından gönderilen DATA_RECEIVED sinyali gözlemlenir. DATA_RECEIVED sinyalinde yükselen kenar görüldüğünde RECEIVED_PACKET sinyali örneklenir ve REQUEST_SEND durumuna geçilir. CHECK_RECEIVE durumunda SEND sinyali 0 sürülür.

REQUEST_SEND durumunda ARINC-429 Tx modülü, alınan veriyi göndermesi için tetiklenir. Tetiklemek için SEND sinyali 1 sürülür, MESSAGE sinyali örneklenen RECEIVED_PACKET değerine sürülür. REQUEST_RECEIVED değeri 1 görüldüğünde CHECK_RECEIVE durumuna geçilir.

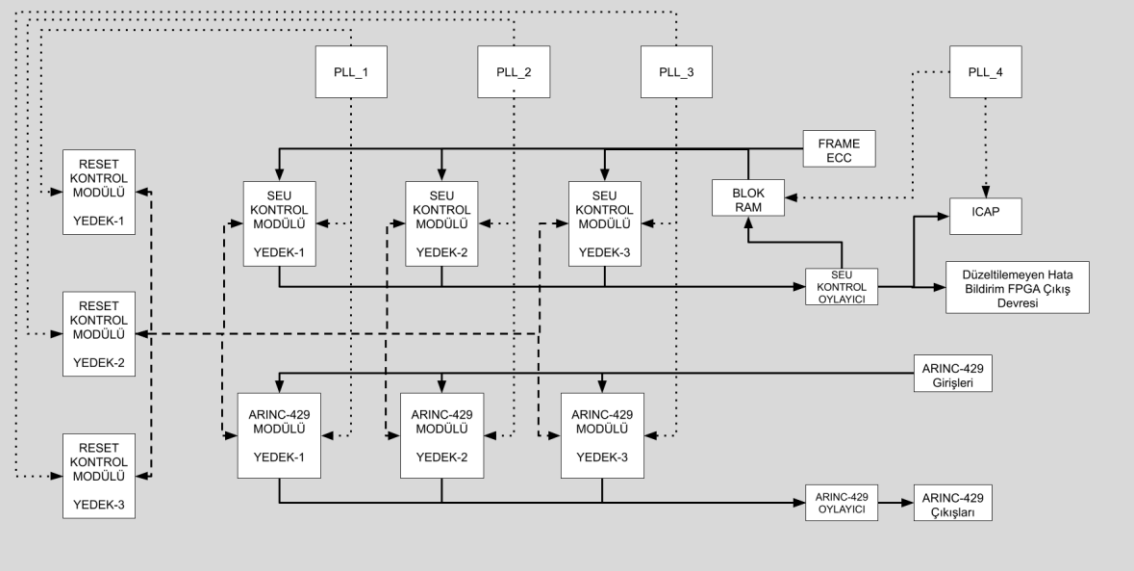
4.6.4. ARINC-429 Giriş Oylayıcı Modülü

ARINC-429 Giriş Oylayıcı Modülü, SEU Kontrol Modülüne gelen oylanması gereken giriş sinyallerinin oylanması için kullanılmıştır. Oylama işlemi asenkron, saat kullanılmadan yapılmıştır. Senkronizasyon sinyalleri SYNC_IN_1, SYNC_IN_2, SYNC_IN_3 ve sıfırlama sinyalleri RESET_1, RESET_2 ve

RESET_3 oylanmıştır. Oylama sonucunda RESET sinyali ve SYNC_IN sinyalleri üretilmiştir. Hata maskeleme yönteminde önerilen modül öncesi oylayıcı tasarım kolaylığı için modül içine taşınmıştır. Eğer gelen sinyallerden birisinde farklılık varsa ilgili sinyali üreten modülün sıfırlanması için RESET kontrol modülüne gönderilmek üzere MODULE_NUMBER sinyali ilgili modülün numaralandırılmış değerine atanır.

4.7. TMR Yapı

Tasarlanan modüller önerilen hata maskeleme yöntemine uygun bir şekilde TMR mimari ile kullanılmıştır. Sistemde yedeklenebilen tüm modüller ve giriş çıkışlar yedeklenmiştir. TMR mimari Şekil 4-12'de görülebilmektedir.



Şekil 4-12 TMR mimari

Tasarlanan tüm bloklar 100Mhz saat ile çalışmaktadır. Oylayıcı blokları saatsiz çalışmaktadır. Çok yüksek hızlarda önerilen yapının kullanılabilmesi için oylayıcı bloklarının da saatle sürülmesi gerekir. Oylayıcı bloklarında yaşanan gecikme ile sentez programı yerleşimi yapamayabilir. Fakat bu tezde kullanılan 100MHz hızda oylayıcıların saat senkron tasarımı gerekmemiştir. Bu bölümde her bir modül tek tek incelenmiştir ve yedek modüllerde gerçekleşebilecek olan hataların maskelenmesi anlatılmıştır.

Tüm modüllerde gerçekleşen hatalar SEU Kontrol Modülleri tarafından tespit edilip PDR ile düzeltilmektedir. Düzeltilen blokların sıfırlanma işlemi RESET Modülleri tarafından gerçekleştirilmektedir. FPGA'da konfigürasyon belleğindeki alanın büyük bir kısmını LUT'lerin kullanılan bitlerinin haricinde kalan bitler tutmaktadır [19]. Bu bitlerin alacağı değer sonucu etkilememektedir. Bir LUT'ye bir fonksiyonu yerine getirirken, içerisinde bulunan tüm bitleri kullanılmaz. Kullanılmayan bitlerine yeni bir fonksiyon da atanamaz. Hataya sebep olmayan bu bitlerin yine de düzeltilmesi gereklidir. Düzeltilmediği taktirde aynı çerçeve içerisinde hataya sebep olabilecek bir bit bozulduğunda, 2-bit hata ECC tarafından düzeltilemediği için probleme sebep olacaktır. Hataya sebep olmayacak bitlerin düzeltilme işleminden sonra sıfırlamaya ihtiyaç yoktur. Oylayıcılarda bir hata ile karşılaşıldığı durumda, hataya sebep olan modül RESET modülü tarafından sıfırlanır. Yedeği olmayan ICAP öncesi oylayıcıda gerçekleşecek olan hataların düzeltilmesi için SEU Kontrol Blokları tarafından bütün şekillendirme işlemi tetiklenmiştir. FRAME_ECC ve ICAP modülü FPGA'da programlanarak değiştirilemeyen kaynaklardandır ve FPGA'da sadece 1 tane vardır. Bu yüzden SEU KONTROL OYLAYICI yedeklenememekte ve gerçekleşecek olan hata maskelenememektedir. ARINC429 OYLAYICI eğer ARINC-429 çıkışları üçe çıkarılırsa yedeklenebilir. ARINC429 OYLAYICI de gerçekleşecek olan hata, bir süre düzgün mesaj gönderilememesine sebep olacaktır. Fakat düzeltilebildiğinden bütün şekillendirme işleminin başlatılmasına ihtiyaç yoktur.

RESET Kontrol Modülleri, sıfırlanan her blok için ayrı sıfırlama sinyali çıkarmaktadır. Sıfırlanan her blok üç yedek RESET Modülünden gelen sıfırlama sinyalini oylayarak kullanmaktadır. RESET Kontrol Modüllerinin çıkışlarının oylanması, çıkışları kullanan blokların içine alınmıştır. Bunun sebebi ise bu oylayıcılarda meydana gelecek olan SEU, girişine bağlandığı bloğun çalışmasını bozacağı için beraber ele alınmıştır. RESET Kontrol Modüllerinden herhangi birinde gerçekleşecek olan hata, diğer yedeklerinin sonucu ile karşılaştırılıp oylanarak kullanılacağı için maskelenmiştir.

SEU kontrol modüllerinin herhangi birinde gerçekleşecek olan hata oylanarak çıkışa verileceği için hata maskelenmiştir. SEU kontrol modülü tüm FPGA de

gerçekleşen SEU hatalarını ICAP arayüzü ile tarayıp, FRAME_ECC_VIRTEX6 ile tespit etmiştir. Tespit edilen hata eğer dinamik alanda ise düzeltip, RESET kontrol modülüne düzeltilen alanı sıfırlanması için bildirmiştir. Düzeltilemeyen hataların FPGA dışı bir kaynak ile düzeltilmesi için FPGA dışına hata bir bit ile bildirilmiştir. SEU kontrol modülü senkronizasyon sinyalleri üretip her işleme başlamadan önce kontrol ederek TMR yapı ile senkronizasyonu sağlanmıştır.

ARINC-429 Modüllerinin herhangi birinde gerçekleşecek olan hata oylanarak çıkışa verileceği için hata maskelenmiştir. ARINC-429 modülü senkronizasyon sinyalleri üretip her işleme başlamadan önce kontrol ederek TMR yapı ile senkronizasyonu sağlanmıştır.

Tasarlanan yapıda ARINC-429 modülünün toleranslı yapısı verilmiştir. Bu yapı farklı veri yolu protokolleri, daha çok sayıda kanal, daha farklı fonksiyonları yerine getiren bloklar için de kullanılabilir.

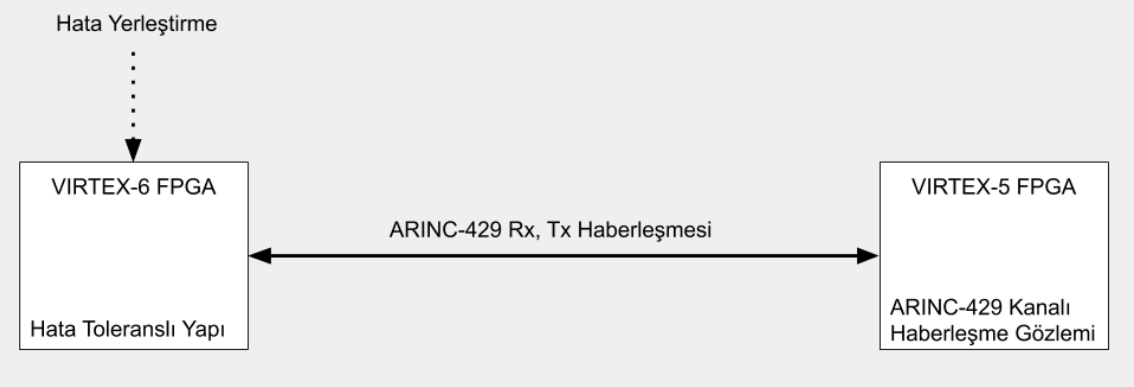
Sistemde kritik hataya sebep olabilecek tek devre SEU KONTROL OYLAYICI modülüdür. Fakat bu modül FPGA de çok az yer kaplayacağı için hata ihtimali çok düşüktür. Sistemin kritik hata ihtimal analizi tezin beşinci bölümünde yapılmıştır. Sistemin çalışmaya devam edip hatasını kendi kendine düzeltebileceği ve çalışmasını durdurup bütün şekillendirme için dışarıdan bir kaynağa ihtiyaç duyacağı hataların ihtimali tezin beşinci bölümünde verilmiştir. Bu analizler önerilen yapının kullanılmadığı durumlar ile karşılaştırılmıştır.

5. SONUÇLAR

Tezin bu bölümünde kurulan yapının performans analizi ve güvenilirlik değerlendirmesi yapılmıştır. Kurulan yapının test ortamı anlatılmıştır. Performans analizi için önerilen yapının kapladığı alan, okuma, yazma ve hata düzeltme süreleri gibi fiziksel özellikleri ölçülmüştür. Bunun haricinde düzeltilemeyen hataların istatistiksel incelemesi yapılmıştır.

5.1. Test Ortamı ve Performans Analizi

Kurulan mimari Xilinx Virtex6 xc6vlx240t-1ff1156 parça numaralı FPGA üzerinde gerçekleştirilmiştir. Dördüncü bölümde anlatılan SEU kontrol modülüne ek olarak bir hata yerleştirme fonksiyonu eklenmiştir. SEU kontrol modülü konfigürasyon belleğini okunurken okuduğu çerçevelerden birinde bir biti değiştirerek yeniden yazması sağlanmıştır. Bozulacak olan çerçevenin yeri, her hatasız okumadan sonra farklı seçilerek, her çerçevede bir bit hata yaratılmıştır. Hata yaratıldıktan sonra modülün hatayı bulup düzeltmesi beklenmiştir. Hata düzeltildikten ve tüm belleğin okunması bittikten sonra yeni bir okuma turunda bir sonraki çerçeve bozulmuştur. Bir çerçevede 20 CLB bulunmaktadır. Çerçeve 81 tane 32 bitlik alanlardan oluşur. Her çerçevenin 32 biti ECC için ayrılmıştır. Geri kalan 80 tane 32 bitlik alanda 20 adet CLB bulunur. Her CLB 128-bit yer tutar. Bir CLB 2 adet SLICE'den oluşur. Her SLICE'de 4 adet LUT bulunur [34]. Xilinx Virtex6 xc6vlx240t-1ff1156 parça numaralı FPGA konfigürasyon belleğinde toplam 28489 adet çerçeve bulunmaktadır [35]. Tüm çerçevelerde en az bir hata oluşturulmuştur. Ayrıca düzeltilemeyen hataların testi için 2-bit bozulmuş ve FPGA dışına düzeltilemeyen hatanın bildirilme fonksiyonu test edilmiştir. Hataların oluşturulduğu ve düzeltildiği sırada ARINC-429 modülü çıkışları Xilinx Virtex5 serisi bir FPGA ile gözlemlenmiştir. Xilinx Virtex5 serisi FPGA ile ARINC-429 verisi test edilen FPGA'ya gönderilmiş ve alınmıştır. Gönderilen veri ile gelen veri karşılaştırılmıştır. Test ortamı Şekil 5-1'de görülebilir.



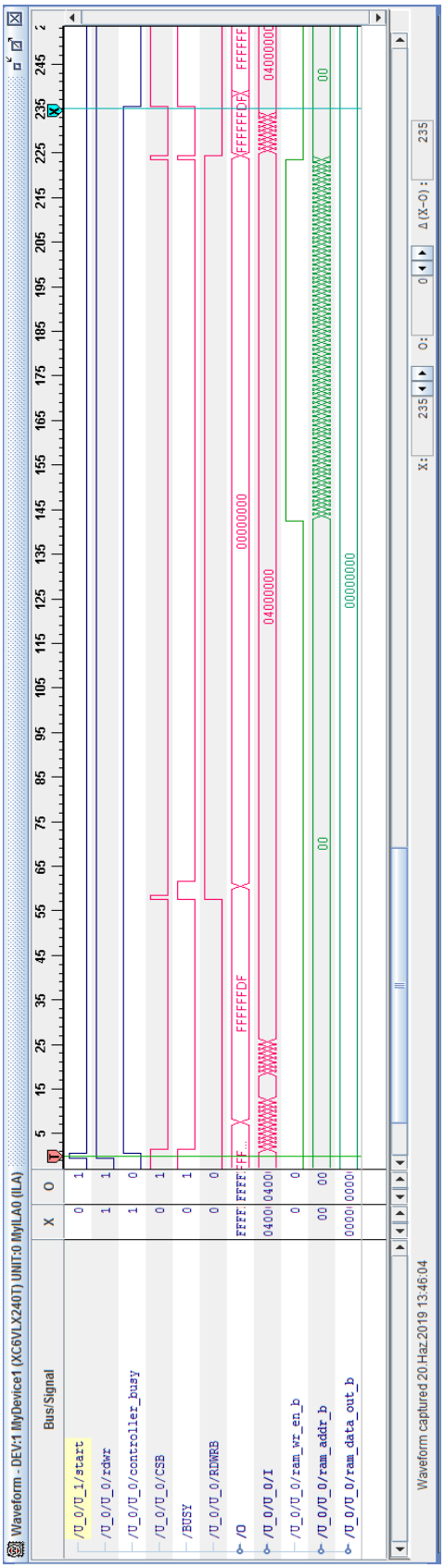
Şekil 5-1 Test Ortamı

Gönderilen veriyi oluşturmak için gereken veriler bir sayaçtan alınmıştır. Test toplam 6 saat boyunca koşturulmuştur ve toplamda 341868 tane hata yerleştirilmiştir. 6 saat süresince ARINC-429 transferinde hata gözlemlenmemiştir. Bu hatalardan 9 tanesi ARINC-429 yedeklerinin çıkışında oylayıcıda gözlemlenmiştir. Benzer şekilde 4 tane hatada SEU Kontrol modülü çıkışında oylayıcıda gözlemlenmiştir. Bu gözlenen hatalar düzeltilebilmiş ve maskelenebilmiştir. Maskelenemeyen hatayı gözlemleyebilmek için kritik öneme sahip olan bitlerin sayısı toplam bit sayısına oranla çok düşüktür. Bu bitlerin konfigürasyon belleğindeki yerleri de bilinmediğinden maskelenemeyen hatalar gözlemlenememiştir. Konfigürasyon belleğindeki tüm bit sayısı olan 73843488 kez hata yerleştirilmesi yapılması ile maskelenemeyen hatalarında gözlemlenebileceği düşünülmektedir. Maskelenemeyen hataların istatistiksel hesabı güvenlik değerlendirmesi kısmında verilmiştir. Test sırasında bir çerçevenin ve tüm konfigürasyon belleğinin okunma ve yazılma süreleri ölçülmüştür. Bu değerler Çizelge 5-1'de görülebilir.

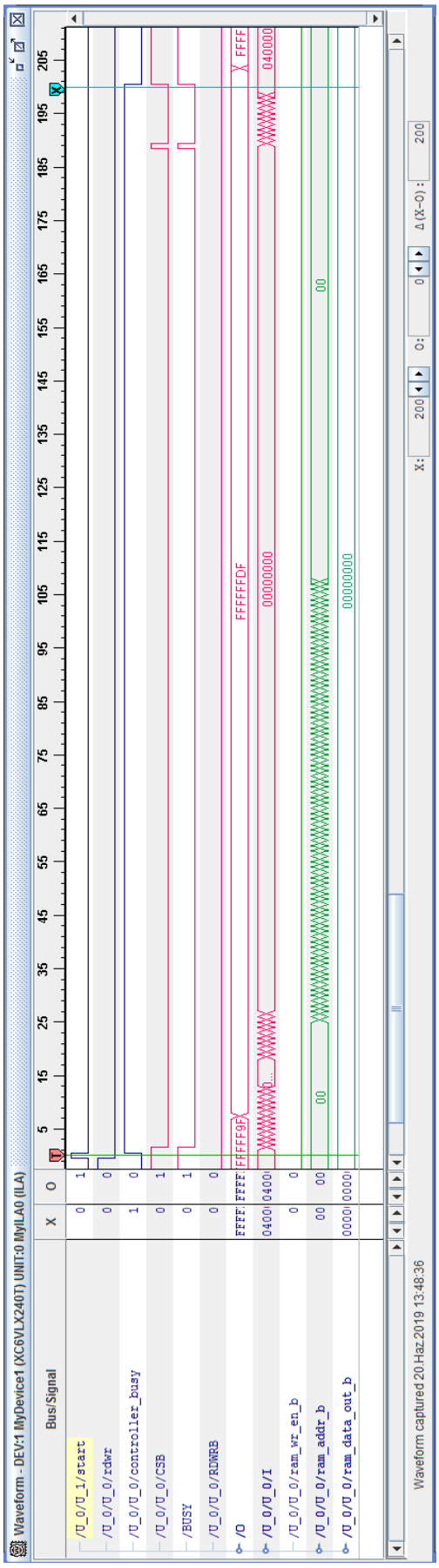
Çizelge 5-1 Konfigürasyon belleği okuma yazma işlemleri süreleri

İşlem	Süre
Bir çerçeve okuma süresi	2350ns
Bir çerçeve yazma süresi	2000ns
Tüm konfigürasyon belleği okuma süresi	66951,500µs
En kötü ihtimal ile SEU düzeltilme süresi	66953,500µs

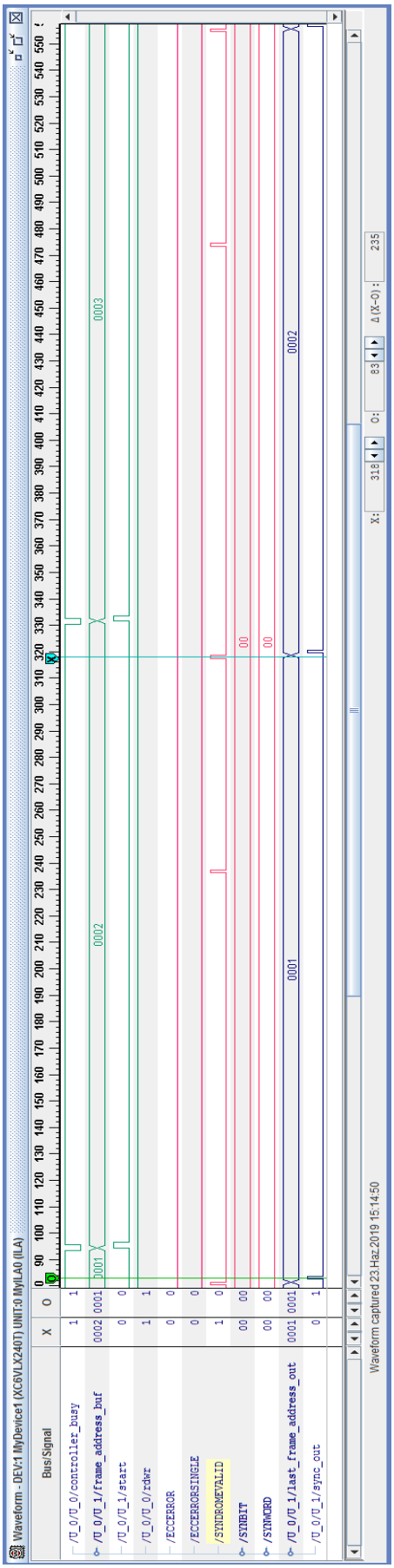
Xilinx FPGA'larda silikon içerisindeki sinyallerin durumunu gözlemleyebilmek gözlem çekirdeği eklenebilmektedir. Bu eklenen gözlem çekirdeği sinyallerin değerlerini anlık olarak alıp JTAG protokolü üzerinden, Xilinx ChipScope programına göndermektedir. Gözlemlenen sinyaller çeşitli anlarda tetiklenerek, tetiklenme anından sonraki bir süre sinyallerin değerleri kaydedilebilmektedir. ChipScope programı kullanılarak bir çerçeve yazma, bir çerçeve okuma, hata bulunmayan ardışık birkaç çerçevenin okunması ve hata yerleştirilmiş çerçeve ve düzeltilmesi esnasında sinyallerin zaman diyagramları alınmıştır. Bu diyagramlar sırasıyla Şekil 5-2, Şekil 5-3, Şekil 5-4 ve Şekil 5-5'de görülebilir.



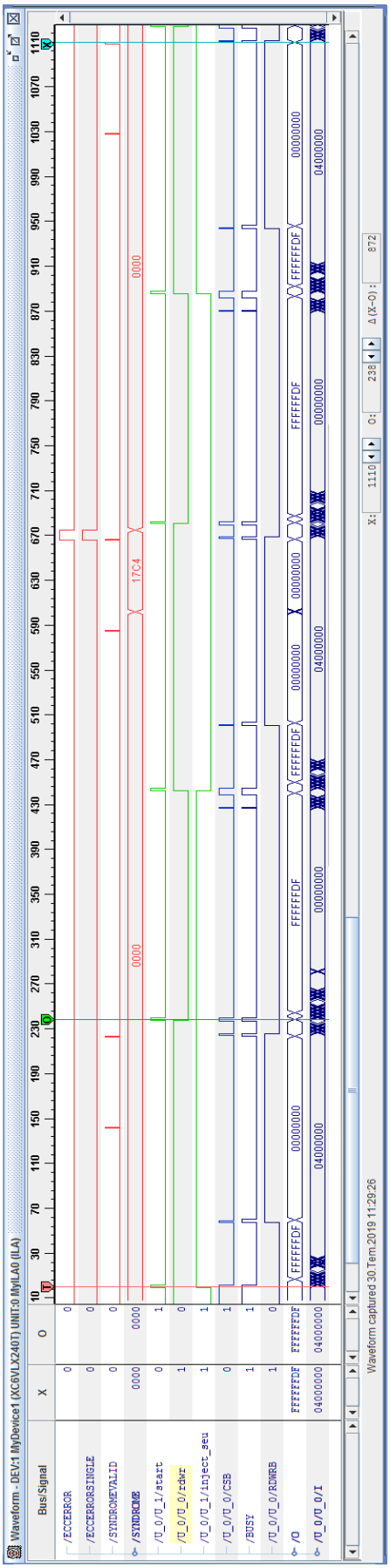
Şekil 5-2 Okuma işlemi Chipscope zaman diyagramı



Şekil 5-3 Yazma işlemi Chipscope zaman diyagramı



Şekil 5-4 Ardışık iki okuma işlemi Chipscope zaman diyagramı



Şekil 5-5 Hata yerleştirme ve düzeltme işlemi Chipscope zaman diyagramı

Zaman diyagramları ve ölçülen süreler dikkate alındığında, tek bir bit hatasının en kötü ihtimal ile bulunma ve düzeltilme süresi yaklaşık 67ms sürmektedir.

TMR yapı kullanılmadan ve TMR yapıda SEU kontrol modüllerinin kapladığı alan, ARINC-429 modülünün kapladığı alan ve OYLAYICI modüllerin kapladığı alanlar Çizelge 5-2'de verilmiştir.

Çizelge 5-2 Modüllerin kapladığı alanlar

Modül	SLICE	LUT	BRAM/FIFO
SEU Kontrol Modülü	171	417	0
SEU Kontrol Modülü TMR Yapı	563	1368	0
BLOK RAM	0	0	2
ARINC-429 Modülü	196	506	0
ARINC-429 Modülü TMR Yapı	635	1675	0
SEU KONTROL OYLAYICI	6	22	0
ARINC-429 OYLAYICI	1	1	0

5.2. İstatistiksel Güvenilirlik Hesaplamaları

Xilinx cihazların güvenilirlik kestirimleri cihaz güvenilirlik raporlarında bulunabilir. SEU hata oranları zaman içinde hata (FIT) ya da MTBF olarak hesaplanmaktadır. FIT 10^9 saatte beklenen hata sayısıdır. MTBF ise iki hata arasındaki ortalama süredir [24].

Literatürdeki çalışmalardan birinde, kurulan sistemdeki hata oranının, tüm cihazın hata oranı ile kritik bitlerin sayısının cihazdaki tüm bitlerin sayısına oranı çarpılarak bulunabildiğini göstermektedir [36].

Kullanılan Virtex6 FPGA'nın atmosferdeki hata oranı 105 FIT/Mb olarak verilmiştir [37].

SEU hata oranı hesaplanırken iki farklı yaklaşım yapılmıştır. Birinci olarak ICAP modülü kontrolü esnasında meydana gelecek kritik düzeltilemeyen ve tüm sistemin bozulup çalışmamasına sebep olacak olan hataların oranı hesaplanmıştır. İkinci olarak ise üçlü çıkış kullanılmadığı takdirde düzeltilebilen fakat maskelenemeyen hataların oranı hesaplanmıştır. Birinci senaryonun hesaplanması için SEU KONTROL OYLAYICI'nın kapladığı alanın tüm sistemin alanına oranı ile Virtex-6'nın atmosferdeki hata oranı çarpılmıştır. İkinci senaryo için ise ARINC-429 OYLAYICI'nın kapladığı alanın TMR sistemde ARINC-429 modüllerinin kapladığı alana oranı ile Virtex-6'nın atmosferdeki hata oranı çarpılmıştır. İkinci senaryoda tüm sistemin alanı ile oylayıcının oranlanmamasının sebebi tasarımda daha fazla modül bulunduğu taktirde oylayıcı sayısının artacağı ve dolayısı ile hata oranında artacağıdır. Tüm tasarımın büyüklüğünden bağımsız olarak en kötü hata oranını bulabilmek için yazılan modüllerin toplam alanı ile gereken toplam çıkış öncesi oylayıcıların alanı oranlanmış ve Virtex-6'nın atmosferdeki hata oranı çarpılmıştır. Değerler Çizelge 5-3 ve Çizelge 5-4'de verilmiştir.

Çizelge 5-3 Kritik Hata ihtimalleri

Durum	Toplam Bit Sayısı	FIT	MTBF
Xilinx Virtex6 xc6vlx240t-1ff1156 Tüm FPGA	73846080 bit	7394 SEU/10^9 h	15.43 Yıl
SEU KONTROL OYLAYICI	384 bit	0.038 SEU/10^9 h	3004085 Yıl

Çizelge 5-4 Maskelenemeyen hata ihtimalleri

Durum	Toplam Bit Sayısı	FIT	MTBF
ARINC-429 OYLAYICI	64 bit	0.006 SEU/10^9 h	19025875 Yıl
ARINC-429 Modülleri tüm FPGA alanını kapladığında ARINC-429 OYLAYICI	40640 bit	4 SEU/10^9 h	28538 Yıl

5.3. Karşılaştırmalı Analiz

Kurulan mimarinin literatürdeki diğer çalışmalar ile karşılaştırması Çizelge 5-5, Çizelge 5-6, Çizelge 5-7 ve Çizelge 5-8'de verilmiştir.

Çizelge 5-5 Genel sistem özellikleri karşılaştırması

	Hata Bulma Süresi	Hata Düzeltme Süresi	Durum Bütünlüğünün Korunması	FPGA Dışı Kaynak İhtiyacı	SEU Toleranssız Tasarıma Göre Ek FPGA Alanı İhtiyacı	Kritik Hata Gerçekleşme İhtimali
Önerilen Mimari	2350ns-67ms	2000ns	Var	Yok	3,3 kat	200 bin kat daha az
H. Baig [19]	Anlık	1s-2s	Var	Var	8 kat	-
F. Lahrach [20]	Anlık	-	Var	Var	1,3 kat	-
C. Bolchini [3]	Anlık	-	Yok	Var	-	-
A. Upegui [17]	-	630ms	Var	Var	-	-
S. Di Carlo [23]	-	-	Var	Yok	-	-
Y. Ichinomiya [16]	Anlık	52µs	Var	Var	4,3 kat	19,8 kat daha az
B. Navas [15]	-	-	Yok	Var	-	-
M. Straka [13]	Anlık	-	Var	Var	2,6 kat	-
S. Ogido [18]	-	3s	Var	Var	4	-
G.I. Alkady [21]	-	-	Yok	Yok	-	-
M. Straka [22]	Anlık	-	-	Yok	-	-
J. Azambuja [14]	Anlık	19,65ms	-	Var	3	-
U. Legat [24]	1,5ms	2100ns	Var	Yok	-	1 milyon kat daha az

Çizelge 5-6 Hata bulma sistemi karşılaştırması

Önerilen Mimari	Konfigürasyon belleğinde gerçekleşen her hata, hatalı bitin yeri ile birlikte minimum 2350ns maksimum 67ms içerisinde bulunabilmektedir.
U. Legat [24]	
H. Baig [19]	Hatalı CLB, yedekli yapıda karşılaştırarak anlık tespit edilebilmektedir.
F. Lahrach [20]	Hatalı CLB, yedekli yapıda karşılaştırarak anlık tespit edilebilmektedir.
Cristiana Bolchini [3]	Hata, TMR modül seviyesinde, TMR çıkışları karşılaştırılarak anlık olarak bulunabilmektedir.
Y. Ichinomiya [16]	
M. Straka [22]	
G.I. Alkady [21]	
M. Straka [13]	
J. Azambuja [14]	
Upegui, A. [17]	Konfigürasyon belleğinde gerçekleşen hataları, TMR modüllerin konfigürasyon belleğindeki yerleri okunup karşılaştırılarak bulunabilmektedir. Hatanın bulunabilme süresi belirtilmemiştir.
B. Navas [15]	Hata, TMR modül seviyesinde, TMR çıkışları karşılaştırılarak bulunabilmektedir. Hatanın bulunabilme süresi belirtilmemiştir.
S. Di Carlo [23]	Hata BIST yapılar kullanılarak anlık olarak bulunabilmektedir. BIST yapılar gerçekleşen her hatayı bulamayabilir.
Seiya Ogido [18]	Hatanın bulunması gerekmemektedir.

Çizelge 5-7 Hata düzeltme sistemi karşılaştırması

Önerilen Mimari	Hata, konfigürasyon belleğinde en küçük yazılabilir bölge olan, bir çerçeve yeniden yazılarak 2000ns sürede düzeltilmiştir. Hatanın düzeltilmesi için ICAP kullanılmış, FPGA dış bir kaynağa ihtiyaç duyulmamıştır. Statik bölge, dinamik bölge farkı bulunmamaktadır.
U. Legat [24]	
H. Baig [19]	Hatalı CLB, CLB seviyesinde tespit edilse dahi minimum bir çerçevelik alan yeniden yazılarak, 1-2s sürede düzeltilmektedir. Hatanın düzeltilmesi için FPGA dış bir kaynak kullanılmıştır.
F. Lahrach [20]	
Cristiana Bolchini [3]	Hatalı TMR modül PDR ile yeniden yazılarak düzeltilmektedir. Düzeltme süresi TMR modülün boyutuna göre değişmektedir. Hatanın düzeltilmesi için FPGA dış bir kaynak kullanılmıştır. Statik bölge hataları çözülememektedir.
Y. Ichinomiya [16]	
M. Straka [22]	
B. Navas [15]	
G.I. Alkady [21]	Hatalı TMR modül PDR ile yeniden yazılarak düzeltilmektedir. Düzeltme süresi TMR modülün boyutuna göre değişmektedir. Hatanın düzeltilmesi için ICAP kullanılmış, FPGA dış bir kaynağa ihtiyaç duyulmamıştır. Statik bölge hataları çözülememektedir.
M. Straka [13]	
J. Azambuja [14]	

Upegui, A. [17]	Konfigürasyon belleğinde birbiri ile birebir aynı olması sağlanan TMR modüllerin hatalı olanı, hatasız olanı okunup, hatalı olan yerine tekrar yazılması ile düzeltilmiştir. Hatanın düzeltilmesi için FPGA dış bir kaynak kullanılmıştır. Statik bölge hataları çözülememektedir.
S. Di Carlo [23]	Hatası bulunan modül PDR ile yeniden yazılarak düzeltilmektedir. Düzeltme süresi modülün boyutuna göre değişmektedir. Hatanın düzeltilmesi için FPGA dış bir kaynak kullanılmıştır. Statik bölge hataları çözülememektedir.
Seiya Ogido [18]	Hatanın varlığına bakılmaksızın, TMR modüller sırayla, belirli bir ritimde tekrardan yazılmaktadır. Hatanın düzeltilmesi için FPGA dış bir kaynak kullanılmıştır. Statik bölge hataları çözülememektedir.

Çizelge 5-8 Hata maskeleme sistemi karşılaştırması

Önerilen Mimari	Çıkış öncesi oylayıcılarda gerçekleşen hatalar hariç, TMR modüllerde, TMR modül arası oylayıcılarda ve saat devrelerinde gerçekleşen hatalar maskelenebilmektedir. Düzeltilen modüllerin senkronizasyonu sağlanarak durum bütünlüğü sağlanmıştır.
H. Baig [19]	Hatalar CLB seviyesi tasarım yapıldığından maskelenebilmektedir.
F. Lahrach [20]	
Cristiana Bolchini [3]	TMR modüllerde gerçekleşen hatalar maskelenebilmiştir. Statik bölgelerde gerçekleşen hataların maskelenmesi ve durum bütünlüğü konularına değinilmemiştir.
M. Straka [22]	
G.I. Alkady [21]	
Seiya Ogido [18]	
B. Navas [15]	
U. Legat [24]	
Y. Ichinomiya [16]	TMR modüllerde gerçekleşen hatalar maskelenebilmiştir. Statik bölgelerde gerçekleşen hataların maskelenmesi konusuna değinilmemiştir. Düzeltilen modüllerin senkronizasyonu sağlanarak durum bütünlüğü sağlanmıştır.
M. Straka [13]	
J. Azambuja [14]	
Upegui, A. [17]	
S. Di Carlo [23]	Hatalar maskelenememektedir. Durum bütünlüğü için işlemci yeniden başlatılmıştır.

6. YORUM

Önerilen çalışmada hata toleranslı sistem mimarisi kurulmuştur. PDR ile düzeltme işlemi 2µs sürmüştür. Bu süre bir çerçevenin yazılma süresidir ve olabilecek en hızlı ve en küçük boyutlu yazma işlemidir. Bir çerçevenin okuma işlemi 2,350µs sürmektedir. Tüm belleğin okuma işlemi 66951,500µs sürmektedir. En kötü ihtimalle hatanın bulunup düzeltilme işlemi 66953,500µs sürmektedir. Sistemde hata bulma ve düzeltme için kullanılan devrenin FPGA alanının yaklaşık 2 binde biri olduğu ölçülmüştür.

67ms boyunca hata düzeltilene kadar sistemin düzgün çalışmaya devam etmesi hedeflenmiştir. Önerilen hata maskeleme yapısı olarak TMR seçilmiştir. TMR yapıda, hatanın maskelenmesi, yapılan hata yerleştirme testi ile denenmiş ve başarılı sonuçlar alınmıştır. Oylayıcıda karşılaşılan hatalar FPGA çıkışında gözlemlenmemiştir. TMR yapıda her bir yedeğin yaklaşık 3,3 katı daha fazla FPGA alanı kullanılmıştır. Bu değer FPGA alanının çok fazla olmadığı tasarımlarda kullanılabilecek seviyededir. TMR yapının FPGA alanına sığmayacağı büyük tasarımlarda fonksiyonların kritikliğine göre değerlendirme yapılıp TMR yapıda kullanılıp kullanılmayacağı değerlendirilmelidir. 67ms bazı fonksiyonlar için hatanın mutlaka maskelenmesi gereken bir süre iken bazı fonksiyonlarda kritiklik teşkil etmeyen bir süre olarak değerlendirilebilir. Örneğin bir video çıkışında gerçekleşecek 67ms'lik hata sadece birkaç video karesinin düzgün çalışmamasına sebep olacaktır. Birkaç video karesinde meydana gelecek olan bu bozulma hatanın maskelenmesini gerektirmeyen bir durum olarak nitelendirilebilir. Aynı şekilde saniyede bir sıcaklık okuması yapan bir devre kritik olarak değerlendirilmeyebilir. Bu durumun tam tersi olarak çok kritik tek bitlik bir sinyalde meydana gelebilecek bir hatanın maskelenmesi için çıkışların üç katına çıkarılması yöntemi kullanılıp FPGA dışında oylanması ile tam olarak maskelenmesi sağlanabilir. Önerilen maskeleme yöntemi daha fazla FPGA alanına ve eğer giriş çıkışlar yedeklenirse üç kata daha fazla giriş çıkışa ihtiyaç duymaktadır. Önerilen hata maskeleme yöntemini fazla kaynak tüketiminden dolayı fonksiyonun kritikliği değerlendirilip ona göre kullanılması tavsiye edilmektedir. Kurulan sistemde hata maskeleme yönteminin bazı fonksiyonlarda

kullanılıp bazı fonksiyonlarda kullanılmadığı bir yapı mümkündür. Hatanın maskelenmediği durumda da oluşan SEU 67ms sonunda çözülecek ve bozulan modül sıfırlandıktan sonra çalışmaya devam edecektir.

Kurulan yapıda SEU KONTROL OYLAYICI'da meydana gelecek olan hata düzeltilemeyecektir. Bu oylayıcıda meydana gelecek olan hata, yapılan yazma işlemlerini bozarak, tüm FPGA'nın bozulmasına ve düzgün çalışmamasına sebep olabilir. Bu durumun olasılık hesabı yapılmıştır. SEU KONTROL OYLAYICI'da bir hata görülme ihtimali yaklaşık 3 milyon yıl olarak hesaplanmıştır. SEU KONTROL OYLAYICI'da meydana gelebilecek olan hatanın ihtimali çok düşük olduğundan bir çözüm önerilmemiştir.

Kurulan yapıda girişler ve çıkışlar yedeklenmezse hatanın maskelenememe ihtimali bulunmaktadır. Giriş ve çıkış devrelerinde veya çıkış öncesi oylayıcıda meydana gelebilecek olan SEU, hatanın maskelenememesine sebep olacaktır. Bu durumun olasılık hesabı yapılmıştır. Tek bir ARINC-429 modülün bulunduğu bir sistemde ARINC-429 OYLAYICI'da hatanın görülme ihtimali yaklaşık 19 milyon yıl olarak hesaplanmıştır. Bu değer çok küçüktür. Fakat yedeklenmeyen çıkışların sayısı arttıkça bu değerde artacaktır. Kullanılan modül sayısından bağımsız olarak, en kötü ihtimalle maskelenemeyen hatanın olasılığını bulabilmek için, ARINC-429 modüllerinin tüm FPGA'nın alanını kapladığı sayıda kullanıldığı durumun olasılık hesabı verilmiştir. Bu değer yaklaşık 29 bin yıl olarak hesaplanmıştır. Bu ihtimal de çok küçüktür. Fakat TMR modüllerin kapladığı alanın oylayıcının kapladığı alana yakın olduğu durumlarda bu hata ihtimali artacaktır. Hata ihtimalinin yüksek olduğu böylesi durumlarda, eğer maskelenmeyen hata kritik hatalara sebep veriyorsa, giriş ve çıkış devreleri yedeklenerek hatanın tamamen maskelenmesi tavsiye edilmektedir.

Önerilen hata toleranslı yapı, hiçbir önlemin alınmadığı yapıya göre, kritik bir hatanın oluşması yönünden 200 bin kat daha güvenlidir.

7. KAYNAKLAR

[1] Xilinx, NSEU Mitigation in Avionics Applications, Xilinx Application Notes XAPP1073 (v1.0) **2010**.

[2] J. B. Ferron, L. Anghel, and R. Leveugle, Analysis of Configuration Bit Criticality in Designs Implemented with SRAM-based FPGAs, in Proc. IEEE Symp. Industrial Electron. Appl. (ISIEA), **2011.**

[3] C. Bolchini, A. Miele, and M. D. Santambrogio, TMR and Partial Dynamic Reconfiguration to Mitigate SEU Faults in FPGAs, 22nd IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, **2007**.

[4] A. Lesea, S. Drimer, J.J. Fabula, C. Carmichael, and P. Alfke, The Rossetta Experiment: Atmospheric Soft Error Rate Testing in Differing Technology FPGAs, IEEE Transactions on Device and Materials Reliability, Vol. 5, No. 3, **2005**.

[5] E. Normand, Single-Event Effects in Avionics, IEEE Transactions on Nuclear Science, Vol. 43, No. 2, **1996.**

[6] Xilinx, Virtex-6 FPGA Configuration, Xilinx User Guides UG360 (v3.9), **2015**.

[7] E. Normand, Correlation of Inflight Neutron Dosimeter and SEU Measurements with Atmospheric Neutron Model, IEEE Transactions Nuclear Science, vol. 48, no. 6, pp. 1996–2003, **2001**.

[8] F. Sturreson, S. Mattson, C. Carmichael, Heavy Ion Characterization of SEU Mitigation Methods for the Virtex FPGA, Radiation Effects on Components and Systems (RADECS) 2001 6 European Conference, pp. 285-291, Sept. **2001**.

[9] D. Hiemstra, V. Kirischian, Single Event Upset Characterization of the Virtex-6 Field Programmable Gate Array Using Proton Irradiation, IEEE REDW, pp. 124-127, **2012**.

[10] Xilinx, Partial Reconfiguration User Guide, Xilinx User Guides UG702 (v14.5), **2013.**

[11] Xilinx, Partial Reconfiguration of Xilinx FPGAs Using ISE Design Suite, Xilinx White Papers, WP374 (v1.2), **2012.**

[12] Xilinx, Partial Reconfiguration Tutorial PlanAhead Design Tool, Xilinx User Guides UG743 (v14.5), **2013.**

[13] M. Straka, J. Kastil, and Z. Kotasek. Generic Partial Dynamic Reconfiguration Controller for Fault Tolerant Designs Based on FPGA, In Proc. of the Int. Conf. on Nordic Microelectronics, (NORCHIP), IEEE, pp. 1-4, **2010**.

[14] J. Azambuja, C. Pilotto, and F. Kastensmidt, Mitigating Soft Errors in SRAM-based FPGAs by Using Large Grain TMR with Selective Partial Reconfiguration, in 2008 European Conference on Radiation and Its Effects on Components and Systems (RADECS), pp. 288-293, **2008**.

[15] B. Navas, J. Oberg, L Sander, The Upset-Fault-Observer: A Concept for Self-Healing Adaptive Fault Tolerance, Adaptive Hardware and Systems (ARS) 2014 NASA/ESA Conference, pp. 89-96, **2014**.

[16] Y. Ichinomiya, S. Tanoue, M. Amagasaki, M. Iida, M. Kuga, T. Sueyoshi, Improving the Robustness of a Softcore Processor Against SEUs by Using TMR and Partial Reconfiguration, IEEE CS, pp. 47-54, **2010**.

[17] A. Upegui, J. Izui, G. Curchod, Fault Mitigation by Means of Dynamic Partial Reconfiguration of Virtex-5 FPGAs, Reconfigurable Computing and FPGAs (ReConFig), **2012**.

[18] S. Ogido, C. Yamada, K. Miyagi, A Study of a Fault-tolerant System Using Dynamic Partial Reconfiguration, 2017 Fifth International Symposium on Computing and Networking, **2017.**

[19] H. Baig, J. A. Lee, An Island-Style-Routing Compatible Fault-Tolerant FPGA Architecture with Self-Repairing Capabilities, 2012 Intl Conf on Field-Programmable Technology, pp. 301-304, **2012**.

[20] F. Lahrach, A. Abdaoui, A. Doumar and E. Chatelet, A Novel SRAM-Based FPGA Architecture for Defect and Fault Tolerance of Configurable Logic Blocks, Design and Diagnostics of Electronic Circuits and Systems,pp. 305-308, **2010**.

[21] G.I. Alkady, N.A. El-Araby, M.B. Abdelhalim, H.H. Amer, A.H. Madian, Dynamic Fault Recovery Using Partial Reconfiguration for Highly Reliable FPGAs, 4th Mediterranean Conference on Embedded Computing MECO, June **2015**.

[22] M. Straka, J. Kastil, and Z. Kotasek, Fault Tolerant Structure for SRAM-

Based FPGA via Partial Dynamic Reconfiguration, in Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on, pp. 365-372, **2010**.

[23] S. Di Carlo, A. Miele, P. Prinetto, A. Trapanese, Microprocessor Fault-Tolerance via on-the-fly Partial Reconfiguration, Proceedings of the 15th IEEE European Test Symposium (ETS) 2010, pp. 201-206, **2010**.

[24] U. Legat, A. Biasizzo, F. Novak, SEU Recovery Mechanism for SRAM-Based FPGAs, IEEE Trans. Nucl. Sci., vol. 59, no. 5, pp. 2562-2571, **2012**.

[25] Xilinx, Virtex FPGA Series Configuration and Readback, Xilinx Application Notes XAPP138 (v2.8) March 11, **2005**.

[26] Xilinx, Correcting Single Event Upsets Through Virtex Partial Configuration, Xilinx Application Notes XAPP216 (v1.0) June 1, **2000**.

[27] Xilinx, Correcting Single-Event Upsets in Virtex-4 FPGA Configuration Memory, Xilinx Application Notes XAPP1088 (v1.0) October 5, **2009**.

[28] Aeronautical Radio, Inc., ARINC Specification 429 Part 1-17, **2004**.

[29] AIM GmbH, ARINC Specification Tutorial, Rev 1.1, **2001**.

[30] Condor Engineering, ARINC 429 Protocol Tutorial (1500-029), Rev. 1.07, **2004.**

[31] Avionics Interface Technologies, ARINC 429 Protocol Tutorial (Doc No. 40100001).

[32] HOLT Integrated Circuits, Single-Rail ARINC 429 Differential Line Driver, HI-8596 Datasheet, **2016**.

[33] HOLT Integrated Circuits, ARINC 429 Line Receiver, HI-8591 Datasheet, **2013**.

[34] Xilinx, Virtex-6 FPGA Configurable Logic Block, Xilinx User Guides UG364 (v1.2) **2012**.

[35] Xilinx, ML605 Hardware User Guide, Xilinx User Guides UG534 (v1.8), **2012**.

[36] R. Velazco, G. Foucard, and P. Peronnard, Combining Results of Accelerated Radiation Tests and Fault Injections to Predict the Error Rate of an Application Implemented in SRAM-based FPGAs, IEEE Trans. Nucl. Sci., vol. 57, no. 6, pp. 3500–3505, **2010**.

[37] Xilinx, Device Reliability Report, Xilinx User Guides UG116, **2018**

EKLER

EK 1- ICAP Komut Dizileri

Okuma öncesi gönderilmesi gereken dizi

Veri Sıra Numarası	Veri
1	FFFFFFFFh
2	000000BBh
3	11220044h
4	FFFFFFFFh
5	AA995566h
6	20000000h
7	30008001h
8	0000000Bh
9	20000000h
10	30008001h
11	00000007h
12	20000000h
13	20000000h
14	20000000h
15	20000000h
16	20000000h
17	20000000h
18	30008001h
19	00000004h
20	20000000h
21	30002001h
22	00000000h
23	28006000h
24	480000A2h
25	20000000h
26	20000000h
27	20000000h

28	20000000h
29	20000000h
30	20000000h
31	20000000h
32	20000000h
33	20000000h
34	20000000h
35	20000000h
36	20000000h
37	20000000h
38	20000000h
39	20000000h
40	20000000h
41	20000000h
42	20000000h
43	20000000h
44	20000000h
45	20000000h
46	20000000h
47	20000000h
48	20000000h
49	20000000h
50	20000000h
51	20000000h
52	20000000h
53	20000000h
54	20000000h
55	20000000h
56	20000000h

Yazma öncesi gönderilmesi gereken dizi

Veri Sıra Numarası	Veri
1	FFFFFFFFh
2	000000BBh
3	11220044h
4	FFFFFFFFh
5	AA995566h
6	20000000h
7	30008001h
8	0000000Bh
9	20000000h
10	30008001h
11	00000007h
12	20000000h
13	20000000h
14	20000000h
15	20000000h
16	20000000h
17	20000000h
18	30018001h
19	44250093h
20	30002001h
21	00000000h
22	30008001h
23	00000001h
24	30004000h
25	500000A2h

Okuma ve yazma sonrası gönderilmesi gereken dizi

Veri Sıra Numarası	Veri
1	20000000h
2	30008001h
3	00000005h
4	20000000h
5	30008001h
6	00000007h
7	20000000h
8	30008001h
9	0000000Dh
10	20000000h
11	20000000h

HACETTEPE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
YÜKSEK LİSANS TEZ ÇALIŞMASI ORJİNALLİK RAPORU

HACETTEPE ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ
ELEKTRİK ve ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI BAŞKANLIĞI'NA

Tarih: 01/10/2019

Tez Başlığı: HAVACILIK UYGULAMALARI İÇİN KISMİ DİNAMİK DONANIM ŞEKİLLENDİRME VE ÜÇLÜ MODÜL YEDEKLEME KULLANILARAK HATA TOLERANSLI FPGA TASARIMI

Yukarıda başlığı gösterilen tez çalışmamın a) Kapak sayfası, b) Giriş, c) Ana bölümler d) Sonuç kısımlarından oluşan toplam 107 sayfalık kısmına ilişkin, 01/10/2019 tarihinde tez danışmanım tarafından *Turnitin* adlı intihal tespit programından aşağıda belirtilen filtrelemeler uygulanarak alınmış olan orijinallik raporuna göre, tezimin benzerlik oranı % 3 'dür.

Uygulanan filtrelemeler:

1- Kaynakça hariç
2- Alıntılar hariç
3- 5 kelimeden daha az örtüşme içeren metin kısımları hariç

Hacettepe Üniversitesi Fen Bilimleri Enstitüsü Tez Çalışması Orjinallik Raporu Alınması ve Kullanılması Uygulama Esasları'nı inceledim ve bu Uygulama Esasları'nda belirtilen azami benzerlik oranlarına göre tez çalışmamın herhangi bir intihal içermediğini; aksinin tespit edileceği muhtemel durumda doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi ve yukarıda vermiş olduğum bilgilerin doğru olduğunu beyan ederim.

Gereğini saygılarımla arz ederim.

Tarih ve İmza

01.10.2019

Adı Soyadı: ALİCAN DÖNMEZ
Öğrenci No: N14324965
Anabilim Dalı: ELEKTRİK ve ELEKTRONİK MÜHENDİSLİĞİ
Programı: ELEKTRİK ve ELEKTRONİK MÜHENDİSLİĞİ
Statüsü: ☒ Y.Lisans ☐ Doktora ☐ Bütünleşik Dr.

DANIŞMAN ONAYI

UYGUNDUR.

PROF. DR. ALİ ZİYA ALKAR

ÖZGEÇMİŞ

Adı Soyadı	: ALİCAN DÖNMEZ
Doğum yeri	: SEYHAN
Doğum tarihi	: 15.09.1990
Medeni hali	: BEKAR
Yazışma adresi	: Eryaman Mah. 346. Sok. Özdeniz Yıldızı Sitesi 1.Blok Kat:2 No:8 Etimesgut Ankara
Telefon	: 0 530 110 24 68
Elektronik posta adresi	: alicandonmez@gmail.com
Yabancı dili	: İngilizce

EĞİTİM DURUMU

Lisans	: Hacettepe Üniversitesi (2008-2013) Elektrik-Elektronik Mühendisliği Bölümü

İş Tecrübesi

Kasım 2015- …	: Aviyonik Donanım Tasarım Mühendisi ASELSAN MGEO, ANKARA
Ekim 2014 – Ekim 2015	: Donanım Tasarım Mühendisi VESTEL ELEKTRONİK, MANİSA