İSTANBUL TEKNİK ÜNİVERSİTESİ

ELEKTRİK-ELEKTRONİK FAKÜLTESİ

GÜVENLİ BİR NFC UYGULAMASININ FPGA ÜZERİNDE GERÇEKLENMESİ

BİTİRME ÖDEVİ

İSMAİL DEMİR

040070359

Bölümü: Elektronik ve Haberleşme Mühendisliği Bölümü

Programı: Elektronik Mühendisliği

Danışmanı: Doc. Dr. S. Berna Örs YALÇIN

MAYIS 2012

ÖNSÖZ

Bitirme projem süresince benden yardımlarını esirgemeyen danışman hocam Sayın S. Berna Örs Yalçın'a ve aileme teşekkürü bir borç bilirim.

İsmail Demir

Mayıs 2012

İÇİNDEKİLER

KISALTMALAR

NFC : Near Field Communication

RF : Radio Frequency

RFID : Radio Frequency Identification

RFCA : RF Collision Avoidance

AES : Advanced Encryption Standard

FPGA : Field Programmable Gate Array

LUT : Look Up Table

VHDL : Very high speed integrated circuit Hardware Description Language

NFCIP-1 : Near Field Communication Interface and Protocol

ASK : Amplitude Shifting Keying

EOF : End of Frame

SOF : Start of Frame

CRC : Cyclic Redundancy Check

DEP : Data Exchange Protocol

PDU : Protocol Data Unit

$\mathbf{T_{IDT}}$: Initial Delay Time

$\mathbf{T_{ADT}}$: Active Delay Time

$\mathbf{T_{RFW}}$: RF Waiting Time

$\mathbf{T_{IRFG}}$: Initial Guard Time

$\mathbf{T_{ARFG}}$: Active Guard Time

ÖZET

Son yıllarda mobil sistemlerin artması ile bu sistemlerde kullanılan kablosuz haberleşme teknolojileri de büyük gelişmeler göstermiştir. Bu teknolojiler arasında Kızılötesi, Bluetooth, ZigBee ve RFID sistemleri gösterilebilir. Son yıllarda ortaya çıkan bir diğer kablosuz haberleşme teknolojisi de NFC (Near Field Communication, Yakın Alan Haberleşmesi) teknolojisidir. ISO 18092 standardında tanımlanmış olan NFC teknolojisi 13.56 Mhz frekansında, 10 cm' ye kadar etkileşim alanı içerisinde ve 106, 212, ve 424 Kbps hızlarında haberleşme olanağı sağlar. Etkileşim alanı kısa olması nedeniyle doğal bir güvenliğe sahiptir. NFC teknolojisini gerçekleyen cihazların iki çalışma modu vardır. Birincisi pasif haberleşme ikincisi aktif haberleşme modudur. Bitirme çalışması kapsamında NFC teknolojisinin aktif modunu gerçekleyen donanım birbirinden farklı görevleri yerine getiren alt modüller içerecek şekilde gerçeklenmiştir. Her ne kadar NFC kısa etkileşim mesafesi dolayısıyla doğal bir güvenliğe sahip olsa da ödeme vb. işlemlerde kullanıldığında bu güvenlik yeterli olmamaktadır. NFC' nin var olan güvenliğini artırmak amacıyla bitirme çalışması kapsamında AES (Advanced Encryption Standard, İleri Şifreleme Standardı) şifreleme standardı kullanılmıştır. Bitirme çalışması kapsamında NFC donanımı ve AES donanımı FPGA(Sahada Programlanabilir Kapı Dizisi) üzerinde gerçeklenmiştir. FPGA üzerinde bulunan MicroBlaze işlemcisi kullanılarak AES ve NFC donanımları kontrol edilmiş ve güvenli bir kablosuz haberleşme sistemi tasarlanmıştır. Bu sistem kullanılarak yapılan testler ve simülasyonlar sonucunda güvenli bir haberleşme sağlanmıştır.

İlerleyen bölümlerde sırasıyla FPGA' lar, NFC teknolojisi ve AES hakkında çalışmanın anlaşılabilmesi için gerekli bilgiler verilmiş, NFC ve AES' i gerçekleyen donanımlar hakkında bilgi verilmiş ve MicroBlaze eklenerek gerçeklenen sistem anlatılmıştır. Sonuç olarak tasarlanan sisteme ilişkin simülasyon ve test sonuçları değerlendirilmiştir.

SUMMARY

1.GİRİŞ

Belli bir amaca yönelik tasarlanan ve bu amacı olabildiğince hızlı ve az güç tüketimiyle gerçekleştiren sistemlere gömülü sistemler denir. Sahada programlanabilir kapı dizileri (FPGA)' ların bulunmasıyla birlikte gömülü sistem tasarımı ve testlerin yapılması oldukça kolaylaşmıştır. FPGA' lar programlanabilir mantık devreleridir. FPGA' nın temel birimi Look Up Table (LUT)' lardır. Bir LUT projede kullanılan FPGA için 16 hücreli bir flip flop dizisi ve bu diziden kod çözme yoluyla 4 bitlik girişin adreslediği hücredeki flip flop içeriğini çıkışına veren mantıksal birimdir. Bu şekilde çalışan LUT dört girişli herhangi bir boole fonksiyonunu gerçekleyebilir. LUT' un çıkışı bir multiplexer sayesinde ister saatle senkron istenirse asenkron alınabilir. Ayrıca LUT'lar arası bağlantılar da programlanabilir yapıdadır. VHDL veya Verilog donanım tanımlama dilleriyle yaptığımız kodlamada kullanılan donanım sentez yazılımı, istenen mantıksal fonksiyonu FPGA üzerinde bulunan LUT 'lara haritalandırır. Donanımın FPGA üzerinde yerleşim ve bağlantı işlemi (Place and Route) yapıldıktan sonra bit uzantılı bir dosya oluşturulur. Bu dosya FPGA'da bulunan LUT'lardaki flip flopların ve bağlantıları yönlendirme için kullanılan multiplexerların kontrol girişlerinde bulunan flip flopların içeriğini bulundurur. Kart üreticisinin sağladığı bir USB kablosu ile bit dosyası FPGA' ya yüklenir. FPGA istenilen fonksiyonu gerçekleştirmeye uygun hale getirilmiş olur.

Proje kapsamında güvenli bir NFC (Near Field Communication, Yakın Alan Haberleşmesi) sistemi AES algoritması kullanılarak FPGA üzerinde tasarlanacaktır.

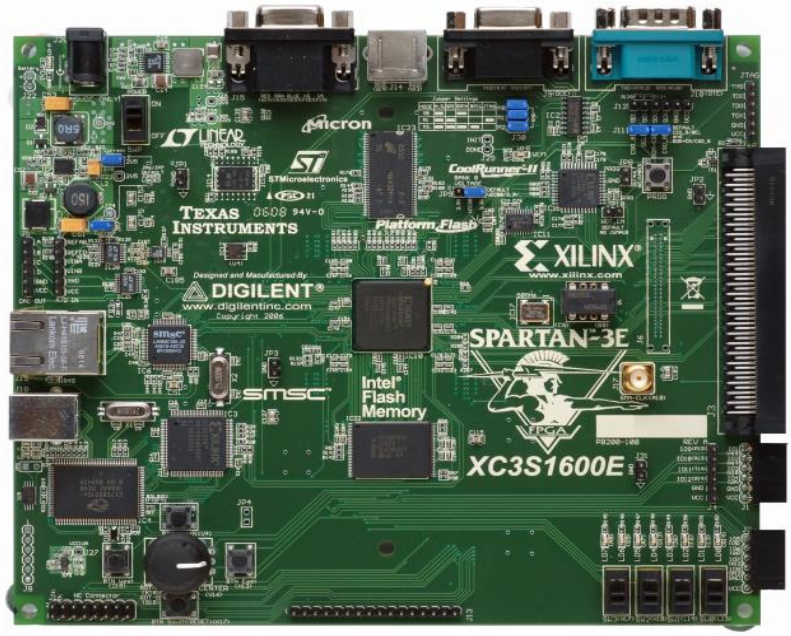
2.PROJEDE KULLANILAN YAZILIM VE DONANIMLAR

2.1.YAZILIMLAR

Proje kapsamında yapılan donanım tasarımlarında VHDL(Very High Speed Integrated Circuits Hardware Description Language, Çok Yüksek Hızlı Tümdevre Donanım Tanımlama Dili) kullanılmıştır. Yazılan donanımların doğrulanması ve sentezlenmesi için XILINX firmasının ISE Design Suite 13.2 yazılımı kullanılmıştır. Bu yazılımın içerdiği üç program; Project Navigator, XPS ve SDK kullanılmıştır. Project Navigator ortamında VHDL ile donanım tasarımı yapılmış ve tasarlanan donanımlar XPS ortamında XILINX XCS1600E Sahada Programlanabilir Kapı Dizisi (FPGA)' nde istenildiği zaman herhangi bir kodlama gerektirmeden oluşturulabilen MicroBlaze işlemcisine çevresel donanım olarak eklenmiştir. SDK ortamında ise tasarımı yapılan bu işlemci ve donanım sistemini kontrol eden yazılım yazılmıştır.

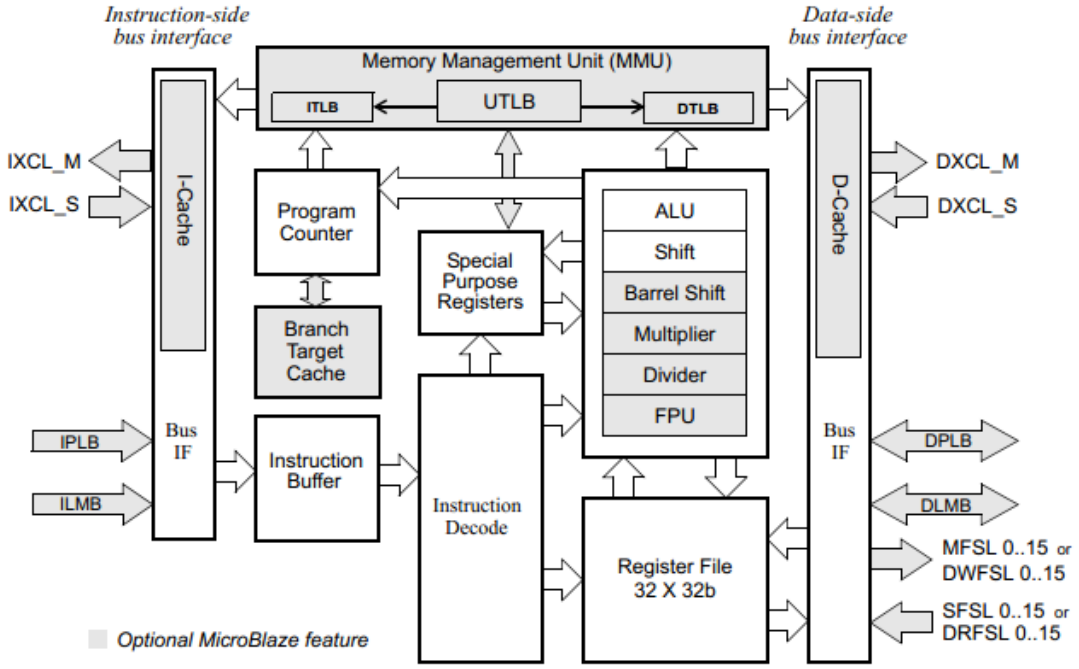
2.2.DONANIMLAR

Gerçeklenen tasarımda XILINX firmasının XCS1600E FPGA' i kullanılmıştır. Kullanılan FPGA' da 64 MB DDR SDRAM, 16 MB NOR Flash bellek, 2x16 LCD ekran, mouse ve klavye girişi, VGA gösterge çıkışı, iki tane RS232 seri haberleşme portu, Ethernet bağlantısı, 8 tane LED, analog dijital ve dijital analog çevirici 4 adet kaydırmalı anahtar ve 1 adet döner buton bulunur [1].



Şekil 2.1.Xilinx XCS1600E FPGA Kartı [1]

.Aynı zamanda bu FPGA üzerinde gömülü olarak bulunmayan fakat istenildiğinde XPS aracılığıyla FPGA üzerinde sentezlenebilen MicroBlaze işlemcisi kullanılmıştır. MicroBlaze 32 bit RISC mimarisine sahiptir. Birbirinden bağımsız 32 bitlik veri ve komut yoluna sahiptir. Soft bir işlemcidir. Şekil 2.2'deki blok diyagramda gri renkli görülen kısımlar seçmelidir ve sistem tasarımcısı tarafından istenirse sisteme eklenir.



Şekil 2.2. MicroBlaze İşlemcisi [2].

3.GÜVENLİ YAKIN ALAN HABERLEŞMESİ SİSTEMİ

Bitirme çalışması boyunca tasarlanması hedeflenen sistem Şekil 3.1' de gösterilen sistemdir. Bu sisteme bakıldığında NFC teknolojisine göre çalışan NFCIP-1 modülü, AES şifreleme ve şifre çözme modülü ile MicroBlaze işlemcisinden oluşmaktadır. Bu sistemi oluşturan NFC ve AES aşağıdaki bölümlerde anlatılmıştır.

3.1.Yakın Alan Haberleşmesi

ISO 18092 ve bu standarda eşdeğer ECMA 340 [3] standardında belirlenmiş olan NFCIP-1 cihazı 13.56 Mhz frekansında ve 10 cm civarında bir alan içerisinde çalışan bir kablosuz haberleşme aygıtıdır. Dinle ve daha sonra cevap ver prensibine göre çalışır. Bu prensibe göre veri gönderen taraf karşı taraftan cevap gelene kadar bekler. Cevap gelince bu cevaba karşılık yeni bir veri paketi yollanır. Bu şekilde sırayla veri gönderimi yapılır. Haberleşmeyi başlatan cihaza initiator karşı tarafa target denir.

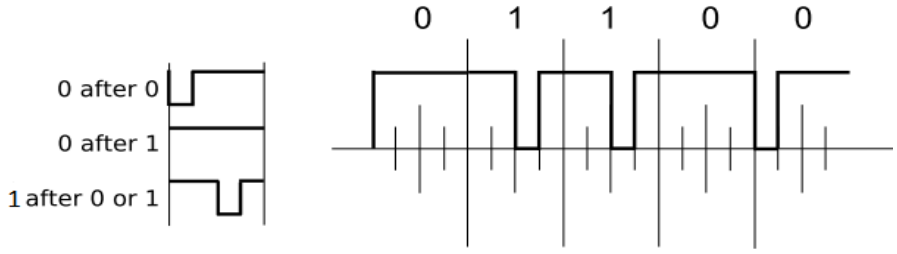
NFCIP-1' in iki çalışma modu bulunmaktadır. Bu çalışma modları pasif ve aktif haberleşme modlarıdır. Pasif haberleşme modunda sadece initiator kendi RF alanını yayınlar. Target initiatorun yayınladığı RF alanından load modulation yöntemiyle enerji elde ederek cevap verir. Aktif haberleşme modunda ise hem initiator hem target kendi RF alanını yayınlar. Her iki çalışma modunda da 106, 212 ve 424 Kbps hızlarının desteklenmesi zorunludur. Çalışma boyunca NFC' nin aktif haberleşme modu tasarlanacağı için sonraki bölümde sadece aktif haberleşme anlatılacaktır.

3.1.1.Aktif Mod Yakın Alan Haberleşmesi

Aktif modda hem initiator hem de target kendi RF alanını yayarak veri alışverişi yapar. Gönderilecek veriye ait bitler hıza göre farklı şekilde kodlama ve modülasyon ile gönderilir. Aynı şekilde bit hızına göre veri formatı da değişir.

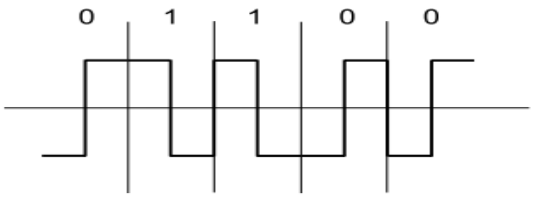
106 Kbps hızında kullanılan modülasyon türü modülasyon derinliği 1 olan genlik kaydırmalı modülasyon (ASK)'dır. Bit kodlaması için Şekil 3.1' de gösterilen ve Modified Miller adı verilen kodlama kullanılır. Modified Miller kodlamasına göre logic-1 bit süresi dörde bölünecek şekilde "1101" dizisi şeklinde kodlanır. Logic-0

ise kendinden önce gönderilen bite göre değişik şekillerde kodlanır. Logic-0'dan önce logic-0 gönderildiyse "0111" şeklinde, logic-1 gönderildiyse "1111" şeklinde kodlanır. Bu kodlamada sof "0111" şeklinde yani logic-0'dan sonra logic-0 gibi kodlanır. eof ise son bite göre kodlanmış bir logic-0'dan sonra "1111" şeklindedir.



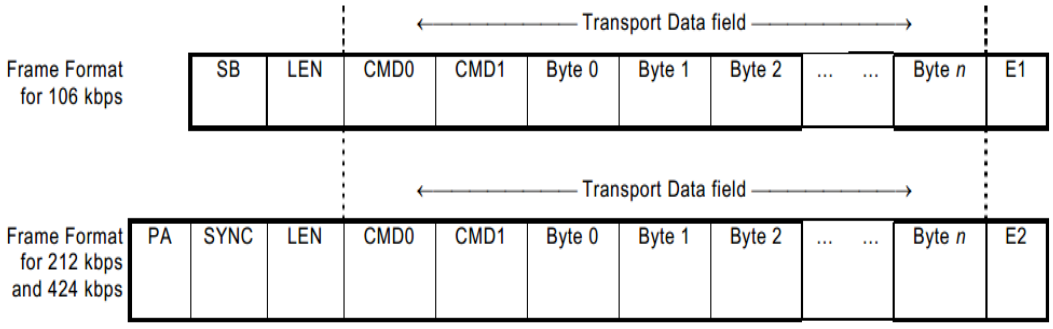
Şekil 3.1. Modified Miller Kodlama [4]. **--- alıntı 24333007 isimli pdften**

106 Kbps dışındaki hızlarda ise 0.08 – 0.3 arası modülasyon derinlikli ASK modülasyonu kullanılır. Bit kodlama için Şekil 3.2' de gösterilen Manchester kodlaması kullanılır. Şekilden de görüldüğü gibi logic-1 için bir bit süresinin yarısı boyunca 1, bit süresinin diğer yarısı boyunca 0 gönderilir. Logic-0 için ise bit süresinin ilk yarısı boyunca 0, ikinci yarısı boyunca 1 gönderilir. Bu kodlama tipinde sof (veri paketi başlangıç karakteri) logic-1 değeridir. eof (veri paketi sonu karakteri) ise bit süresinin her iki yarısı boyunca 1 olması durumudur.



Şekil 3.2. Manchester Kodlama [4].

Şekil 3.3' de 106 Kbps ve diğer hızlar için kullanılan veri paketi formatları gösterilmiştir. Şekilde gösterilen her iki format için de tüm baytlar eşlik biti içerir.



Şekil 3.3.106 Kbps ve diğer hızlar için veri paketi formatları [3].

106 Kbps için kullanılan formatta gösterilen SB başlangıç baytını ifade eder ve değeri hexadecimal F0 dır. LEN ile ifade edilen bayt Transport data field adlı bölgenin uzunluğunu belirtir. Transport data field kısım ile bir sonraki bölümde anlatılacak olan komutlardan birini ve gerekli ise NFC haberleşmesini kullanan uygulamaya ait veriler taşınır. E1 ile gösterilen kısım veri iki baytlık CRC(Cyclic Redundancy Check)'dir. Bu CRC hesaplanırken veri paketinde gösterilen tüm baytlar (eşlik bitleri hariç) işleme katılır ve CRC'nin başlangıç değeri hexadecimal 6363' tür. 212 ve 424 Kbps hızlarında kullanılan formatta gösterilen PA 6 preamble baytıdır ve tamamı 0'dır. SYNC ile gösterilen ise 2 senkronizasyon baytıdır ve değeri hexadecimal B24D'dir. LEN baytı Ttansport data field'ın uzunluğudur. E2 ise 2 baytlık CRC'dir. CRC işlemine PA ve SYNC dışındaki tüm baytlar alınır ve CRC'nin ilk değeri hexadecimal 0000'dır. CRC işlemi aşağıdaki polinoma göre hesaplanır.

$$G(x) = x^{16} + x^{12} + x^5 + 1 \qquad .(3.1)$$

3.1.1.1.Aktif NFC' de Kullanılan Komutlar

Aktif NFC haberleşmesinde Tablo 3.1' de gösterilen komutlar kullanılmaktadır. Tablodan da anlaşılacağı üzere request (istek) her zaman initiatordan gelir. Yani target, initiator istek göndermediği sürece cevap yollamaz ve initiatordan gelecek istekleri bekler.

Tablo 3.1. Aktif NFC Komutları [3].

Mnemonic	Command Bytes		Definition
	CMD0	CMD1	
ATR_REQ	(D4)	(00)	Attribute Request (sent by Initiator)
ATR_RES	(D5)	(01)	Attribute Response (sent by Target)
WUP_REQ	(D4)	(02)	Wakeup Request (sent by Initiator in Active mode only)
WUP_RES	(D5)	(03)	Wakeup Response (sent by Target in Active mode only)
PSL_REQ	(D4)	(04)	Parameter selection Request (sent by Initiator)
PSL_RES	(D5)	(05)	Parameter selection Response (sent by Target)
DEP_REQ	(D4)	(06)	Data Exchange Protocol Request (sent by Initiator)
DEP_RES	(D5)	(07)	Data Exchange Protocol Response (sent by Target)
DSL_REQ	(D4)	(08)	Deselect Request (sent by Initiator)
DSL_RES	(D5)	(09)	Deselect Response (sent by Target)
RLS_REQ	(D4)	(0A)	Release Request (sent by Initiator)
RLS_RES	(D5)	(0B)	Release Response (sent by Target)

3.1.1.2.Haberleşmenin Başlatılması

Haberleşmeyi initiator başlatır. Haberleşmenin başlatılması aşağıdaki adımlar uygulanarak yapılır. Aşağıda geçen komutlar bir sonraki bölümde anlatılacaktır.

- İnitiator devam etmekte olan başka bir haberleşmeyi engellememek için ortamda RF alanı varlığını kontrol eder. $T_{IDT} > 4\ 096 / f_C$, $f_C = 13.56$ Mhz ve $0 \leq N \leq 3$, N rastgele olmak üzere; $T_{IDT} + N \times T_{RFW}$ süresi boyunca ortamda algılanabilir RF alanı belirleyemezse kendi RF alanını modüle edilmemiş taşıyıcı olarak $T_{IRFG} > 5$ms süresi boyunca yayınlar. Bu işleme initial RFCA (initial RF Collision Avoidance, birincil RF girişimi sakınımı) denir. Bundan sonra haberleşmenin başlaması için gönderilmesi gerekli olan atr_req (Attribute Request,) komutu gönderilir.
- Target RF alanını algılar ve atr_req komutunu alır. Response RFCA adı verilen işlemi uygular. $768/fc \leq T_{ADT} \leq 2\ 559/fc$ ve $0 \leq N \leq 3$, N rastgele olmak üzere; $T_{ADT} + N \times T_{RFW}$ süresi boyunca ortamda algılanabilir RF alanı belirleyemezse kendi RF alanını modüle edilmemiş taşıyıcı olarak $T_{ARFG} > 1024/fc$ süresi boyunca yayınlar. Sonra cevap olarak atr_res (Attribute Response,) komutunu gönderir.

Atr_res komutunu alan initiator ikinci bir RFCA uygular. T_{ADT} süresi boyunca ortamda RF alanı belirleyemezse T_{ARFG} süresi boyunca modüle edilmemiş taşıyıcı yayınlar. Artık haberleşme başlatılmıştır. İnitiator kendisini kontrol eden host (bir işlemci olabilir) tan gelecek komutlara göre veri alışverişi yapmaya başlar.

3.1.1.2.1.Atr_req ve Atr_res Komutları

Haberleşmenin başlaması için kullanılan komutlardır. İnitiatorun yolladığı atr_req komutuna target atr_res komutu ile cevap verir. Atr_req komutunun yapısı Şekil 3.4' te gösterilmiştir. CMD0 tüm komutlarda komutun istek mi cevap mı olduğunu belirtir. Hexadecimal D4 değeri komutun istek olduğunu gösterir. Tüm komutlarda CMD1 o komutun hangi komut olduğunu belirtir. Hexadecimal 00 değeri bu komutun atr_req olduğunu belirtir.

CMD 0	CMD 1	Byte 0	...	Byte 9	Byte 10	Byte 11	Byte 12	Byte 13	Byte 14	...	Byte *n*
(D4)	(00)	nfcid3i0	...	nfcid3i9	DIDi	BSi	BRi	PPi	[Gi[0]]	...	[Gi[*n*]]

Şekil 3.4. Atr_req komutu [3].

0’ dan 9’ a kadar olan baytlara nfcid3 denir ve initiatorun haberleşme boyunca kullanacağı adıdır. nfcid3 her haberleşmede rastgele oluşturulur. 10. bayt DID baytıdır ve initiatorun targeta atadığı bir numaradır. 11. ve 12. Baytlar sırasıyla BS ve BR baytlarıdır ve initiatorun desteklediği veri gönderme hızı ile veri alma hızlarını ifade eder. Bu baytların yüksek anlamlı dört biti 0 olmalıdır. Düşük anlamlı dört bitinin aldığı değere X dersek; en büyük veri alma veya gönderme hızı 106×2^X Kbps şeklinde hesaplanır. 13. bayt PP baytıdır ve initiatorun desteklediği ekstra seçenekleri belirtir. PP baytının 5. ve 4. bitinin değerine X dersek; gönderilebilecek ve alınabilecek en büyük veri uzunluğu $64\times2^X - 1$’ dir. 15. bayt ve sonraki baytlar gönderilecekse 2. biti 1 yapılmalıdır.

CMD 0	CMD 1	Byte 0	...	Byte 9	Byte 10	Byte 11	Byte 12	Byte 13	Byte 14	Byte 15	...	Byte *n*
(D5)	(01)	nfcid3t0	...	nfcid3t9	DIDt	BSt	BRt	TO	PPt	[Gt0]]	...	[Gt[*n*]]

Şekil 3.5. Atr_res komutu [3]

Atr_res komutunun yapısı Şekil 3.5’ te gösterilmiştir. atr_req komutu için anlatılanlar atr_res için de aynen geçerlidir. Tek fark atr_res’ in 14. baytıdır ve bu bayt targetin kullandığı zaman aşımı süresini belirtir. Yüksek anlamlı dört biti 0 olmalıdır. Düşük anlamlı dört bitine X dersek; zaman aşımı değeri aşağıdaki formüle göre hesaplanır:

$$\text{Zaman Aşımı} = (256 \times 16 / fc) \times 2^X \qquad (3.2)$$

3.1.1.3.Haberleşmeye İlişkin Parametrelerin Değiştirilmesi

atr_req gönderip atr_res alarak haberleşmeyi başlattıktan sonra istediği zaman veri gönderme ve alma hızı ile en büyük veri paketi uzunluğu parametrelerini istediği zaman değiştirebilir. Bu değişiklikler için initiator psl_req komutunu gönderir ve psl_res komutu cevabını bekler. Parametre değişikliği yapmadan önce hostun atr_res komutu ile alınan ve targetın desteklediği en yüksek veri gönderme ve alma hızı ile veri uzunluğunu gösteren baytlar olan BS, BR ve PP baytlarına bakması gerekir. Bu baytlara uygun bir değişiklik isteği göndermelidir.

3.1.1.3.1.Psl_req ve Psl_res Komutları

İnitiator veri gönderme ve alma hızları ile gönderilebilecek en büyük veri uzunluğunu değiştirmek istediğinde bu komutu kullanır. Komutun yapısı Şekil 3.6’ de gösterilmiştir. CMD0 ve CMD1 komutun psl_req olduğunu gösterir. 0. bayt DID

baytıdır. 1. bayt ile veri gönderme ve alma hızının değiştirilmesi istenen yeni değer belirtilir. İnitiatorun yeni veri alma hızının 106 * $2^{BRS(2:0)}$ olarak ve veri gönderme hızının 106 × $2^{BRS(5:3)}$ olarak değiştirilmesinin istendiğini gösterir. FSL baytı gönderilebilecek en büyük veri uzunluğunun yeni değerinin $64\times2^{FSL(1:0)}$ – 1 olarak değiştirilmek istendiğini belirtir.

CMD 0	CMD 1	Byte 0	Byte 1	Byte 2
(D4)	(04)	DID	BRS	FSL

Şekil 3.6. Psl_req komutu [3]

Target psl_req komutunu alınca BRS ve FSL baytları ile değiştirilmek istenen parametrelerin alacağı yeni değerlere bakar ve eğer desteklediği değişimler ise Şekil 3.7' da gösterilen yapıdaki psl_res komutunu gönderir. Eğer desteklemediği bir değişim var ise herhangi bir cevap göndermez.

CMD0	CMD1	Byte 0
(D5)	(05)	DID

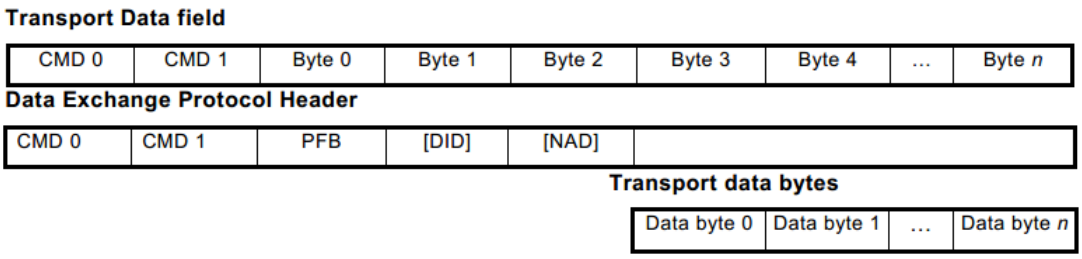
Şekil 3.7. Psl_res komutu [3]

3.1.1.4.DEP

İnitiator atr_req komutu gönderip atr_res komutunu alınca haberleşme başlamış olur ve DEP (Data Exchange Protocol, Veri Değişimi Protokolü)' i kullanarak target ile veri alışverişine başlayabilir. DEP için kullanılan komutlar dep_req ve dep_res komutlarıdır.

3.1.1.4.1.Dep_req ve Dep_res Komutları

Her iki komutta aynı yapıda olup Şekil 3.8' da gösterilmektedir. dep_req ve dep_res komutları ile gönderilen veriye PDU (Protocol Data Unit, Protokol Veri Birimi) denir.



Şekil 3.8. Dep_req ve Dep_res komutları [3]

Transport data field adıyla gösterilen kısım daha önce anlatılmıştı. Transport data bytes ile gösterile kısım gönderilen genel amaçlı verilerdir. CMD0 ve CMD1 komutun dep_req veya dep_res olduğunu belirtir. 1. bayt DID baytıdır. 0. bayt PFB baytıdır ve haberleşme için gerekli kontrol bilgilerini içerir. PFB baytının yüksek anlamlı üç biti PDU tipini belirler. Dört farklı PDU vardır. Bu PDU'lar Tablo 3.2'de gösterilmişlerdir.

Tablo 3.2.PDU Türleri [3]

bit 7	bit 6	bit 5	PFB
0	0	0	Information pdu
0	0	1	NFC-SEC pdu
0	1	0	ACK/NACK pdu
1	0	0	Supervisory pdu
Other settings are RFU			

Information (bilgi) PDU'su genel amaçlı bilgilerin taşınması için kullanılır. NFC-SEC PDU' su korumalı verilerin taşınması için kullanılır. Bu PDU'nun kullanılması için hem initiatorun hem de targetın NFC-SEC desteği olması gerekir. Bu iki PDU için PFB baytının yapısı yüksek anlamlı üç biti Tablo 3.2 de gösterildiği gibi ve diğer bitleri aynı olup dördüncü bit gönderilen verinin bir veri paketine sığıp sığmadığını belirler. Eğer dördüncü bit 1 ise veri bir pakete sığmamış ve devamı gelecek demektir. Bu durumda bu PDU' yu alan taraf Tablo 3.2' de görülen PDU'lardan ACK PDU' yu gönderir. PFB' nin

3.1.1.5.Haberleşmenin Sonlandırılması

İnitiator NFC haberleşmesini sonlandırmak istediğinde dsl_req veya rls_req komutlarından birini kullanır.

3.1.1.5.1.Dsl_req ve Dsl_res Komutları

İnitiator haberleşmeyi wup_req ile tekrar başlatmak üzere bitirmek istediği zaman Şekil 3.9' de gösterilen yapıdaki dsl_req komutunu gönderir.

CMD0	CMD1	Byte 0
(D4)	(08)	[DID]

Şekil 3.9. Dsl_req komutu [3]

dsl_req komutunu alan target bu komuta Şekil 3.10’ de görülen dsl_res komutu ile cevap verir ve haberleşme sona erer.

CMD0	CMD1	Byte 0
(D5)	(09)	[DID]

Şekil 3.10. Dsl_res komutu [3]

3.1.1.5.2.Rls_req ve Rls_res Komutları

İnitiator haberleşmeyi tamamen bitirmek istediği zaman Şekil 3.11’ de görülen rls_req komutunu gönderir.

CMD0	CMD1	Byte 0
(D4)	(08)	[DID]

Şekil 3.11. Rls_req komutu [3]

rls_req komutunu alan target bu komuta Şekil 3.12’ de görülen rls_res komutu ile cevap verir ve haberleşme sona erer.

CMD0	CMD1	Byte 0
(D5)	(09)	[DID]

Şekil 3.12. Rls_res komutu [3]

İnitiator haberleşmeyi tekrar başlatmak isterse önceki bölümlerde anlatılan yöntemle tekrar atr_req göndermelidir.

3.1.1.5.3.Wup_req ve Wup_res Komutları

dsl_req komutunu gönderip dsl_res komutunu alarak targetla haberleşmeyi durdurmuş olan initiator haberleşmeyi tekrar başlatmak istediğinde Şekil 3.13’ da görülen yapıdaki wup_req komutunu yollar. Burada kullanılan nfcid3t haberleşme kesilmeden önce targetın kullandığı nfcid3’ tür. DID ise aynı şekilde önceki haberleşmede kullanılan DID’ dır.

CMD 0	CMD 1	Byte 0	…	Byte 9	Byte 10
(D4)	(02)	nfcid3t0	…	nfcid3t9	DID

Şekil 3.13. Wup_req komutu [3]

Wup_req komutunu alan target nfcid3 ve DID' ı kontrol eder ve önceki haberleşmedeki değerleriyle aynı ise Şekil 3.14' de gösterilen wup_res komutunu gönderir ve haberleşme yeniden başlamış olur.

CMD 0	CMD 1	Byte 0
(D5)	(03)	DID

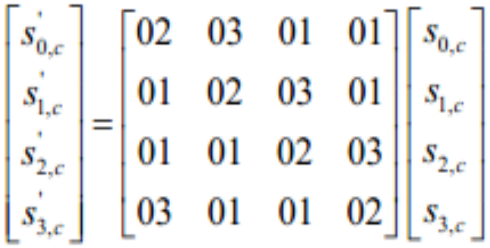
Şekil 3.14. Wup_res komutu [3]

3.2.AES

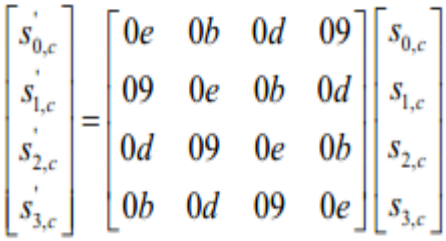
AES (Advanced Encryption Standard, İleri Şifreleme Standardı) 2001 yılında NIST (National Institue of Standards and Technology, Amerikan Ulusal Standartlar ve Teknoloji Enstitüsü) tarafından yayınlanmış şifreleme standardıdır. AES algoritması blok halindeki bilgiyi şifreleyen ve şifresini çözebilen simetrik bir şifre algoritmasıdır. Şifreleme bilgiyi şifreli metin olarak adlandırılan anlaşılmaz bir hale çevirir ve şifre çözme bu anlaşılmaz metni tekrar orijinal haline çevirir. AES algoritması 128, 192 veya 256 bit uzunluklu anahtar kullanarak 128 bitlik veriyi şifreleyip şifre çözebilir.[5].

AES algoritması aynı işlemlerden oluşan ve tur adı verilen 10 adımdan oluşur. Bu turlara geçmeden önce 16 bayttan oluşan ilk durum elde edilir. İlk durum şifrelenmemiş veri ile anahtarın bit bit özel veya işlemine tabi tutulması ile elde edilir. Daha sonra elde edilen ilk durum ile 1. tura geçilir. 10 tur tamamlanınca son tur uygulanır ve son turun çıkışı şifreli veriyi verir. Her turda aşağıdaki işlemler tekrarlanır:

- Anahtar üretici adı verilen bir modül ilk anahtardan yeni bir anahtar türetir.
- Bayt değiştirme : Her adımda durum S-kutusu denilen 4x4 boyutlu bir matrise yerleştirilir ve bu matris kullanılarak baytların yeri değiştirilir. Şifre çözme işleminde ters S-kutusu kullanılır.
- Satır kaydırma : S-kutusunun satırları olarak sola kaydırılır. İlk satır aynen kalır. İkinci satır bir adım, üçüncü satır iki adım ve dördüncü satır üç adım sola kaydırılır. Şifre çözmede aynı adımlar sağa kaydırma ile yapılır.
- Sütun karıştırma : Şekil 3.16' da gösterilen matris çarpımı yapılarak sütunlar karıştırılır. Şifre çözerken ise Şekil 3.17'deki matris çarpımı yapılır.



Şekil 3.15.Sütun Karıştırma [5].



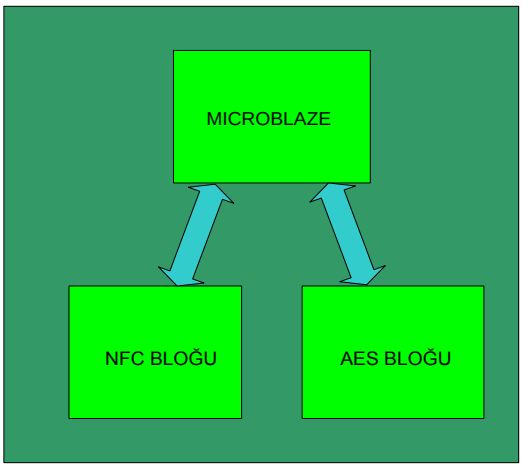
Şekil 3.16. Ters Sütun Karıştırma[5].

- Tur anahtarı ekleme : Her turda üretilen anahtar ile S-kutusu bit bit özel veya işlemine sokulur.

Şifre çözme işleminde satır kaydırma bayt değiştirmeden ve tur anahtarı ekleme işlemi ise sütun karıştırma işleminden önce yapılır.

4.GÜVENLİ NFC SİSTEMİNİN TASARIMI

Sistem Şekil 4.1'de görülen blok diyagramda görülmektedir. Üç alt bloktan oluşmaktadır. MicroBlaze işlemcisi sistemi yönetmektedir ve güvenli NFC sistemini kullanacak uygulamaların bu işlemci üzerinde çalışacağı düşünülmüşdir. Diğer alt bloklar ise NFC protokolü ile AES algoritmasını gerçekleyen bloklardır.



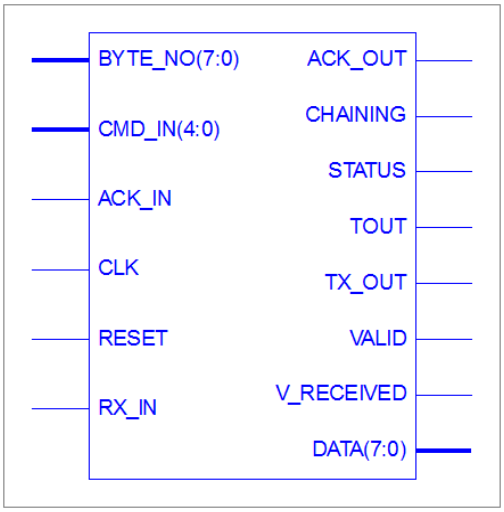
Şekil 4.1.Güvenli NFC sistemi.

4.1.NFC Protokolünü Gerçekleyen Aygıt (NFCIP-1)

NFCIP-1 cihazı ISO/IEC 18092 ve ECMA 340 standartlarında tanımlanmıştır. Çalışma boyunca bu standartlardan ECMA 340 standardına başvurulmuştur. Standardın aktif haberleşme bölümü VHDL dili ile XILINX ISE ortamında tasarlanmıştır. NFCIP modülünün tasarımında ECMA 340 standardını gerçeklemek üzere aşağıdaki modüllerin tasarımına karar verilmiştir.

4.1.1.NFCIP-1

NFC protokolünü gerçekleyen üst modüldür. Blok diyagramı, Giriş-çıkış özellikleri, kullanıcı kontrol komutları ve çalışma prensibi aşağıda verilmiştir.



Şekil 4.1.NFCIP-1 modülü.

Saat girişi, reset girişi, hosttan gelecek olan 8 bitlik çift yönlü veri giriş çıkış portu data, hosttan gelecek 1 bitlik bilgilendirme girişi ack_in, 8 bitlik rastgele erişimli bellek adres portu byte_no, 5 bitlik komut giriş portu cmd_in, NFCIP modülünün initiator modda iken bir target ile, target modda iken bir initiator ile etkileşimde olup olmadığını gösteren status çıkışı, bir veri paketi alındığında, zaman aşımı olduğunda veya bir NFCIP ile etkileşime geçildiğinde hostu uyaran ack_out çıkışı, alınan veri paketinin doğru formatta olduğunu gösteren v_received çıkışı, initiator modda iken alınan cevabın gönderilmiş olan isteğe uygun bir cevap olduğunu gösteren valid çıkışı, DEP protokolü esnasında bir veri paketine sığmayan veri geldiğinde hostu uyaran chaining çıkışı ve bir bitlik RF vericiye giden tx_out çıkışı ile RF alıcıdan gelen rx_in girişi vardır.

4.1.1.1.NFCIP-1 Modülü Kontrol Komutları

NFCIP modülünü kontrol etmek için kullanıcının kullanabileceği komutlar şunlardır:

- Ch_mode komutu : Bu komut ile birlikte data girişinde olarak "01h" yazılınca NFCIP-1 initiator moda, "00h" yazılınca target moda geçer. Bu komut resetten sonra veya devam etmekte olan haberleşme sona erdirildikten sonra kullanılabilir.
- İnitial_rate komutu : Eğer initiator mod seçilmiş ve NFCIP-1 bekliyorsa bu komut ile haberleşmenin yapılacağı ilk hız data girişi kullanılarak belirtilir. data girişi "00h" ise 106 Kbps, "01h" ise 212 Kbps, "02h" ise 424 Kbps hızı

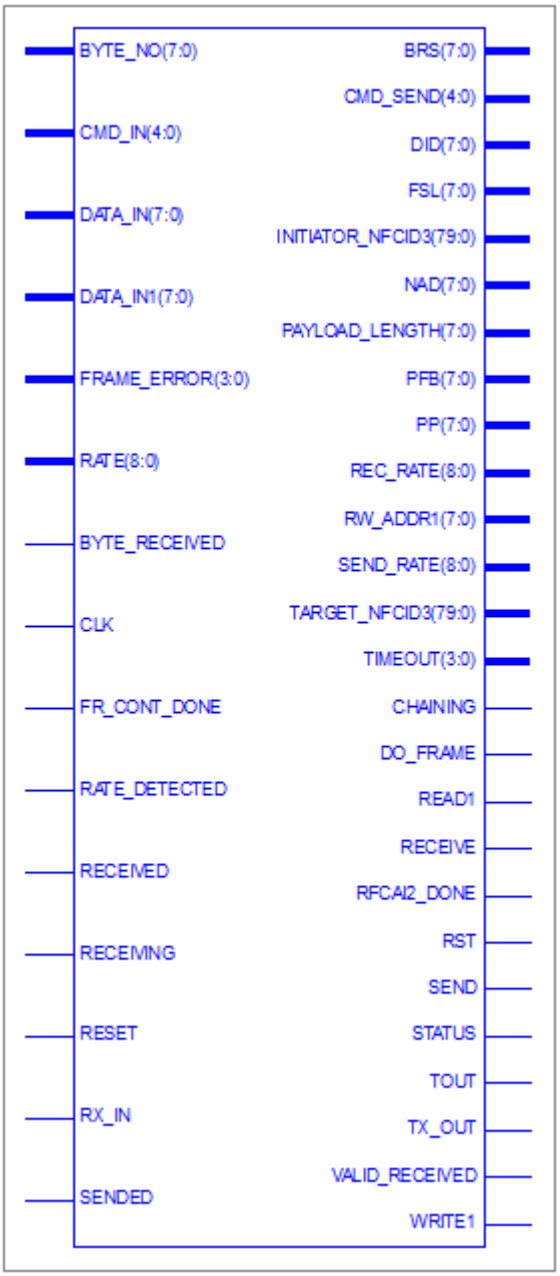
seçilmiş olur. Target modda ilk hız seçme yoktur. İnitiator'un bit hızını ilk alınan pakete bakarak belirlemek gerekir.

- Sstart komutu : Ch_mode komutu ile initiator modu seçildiyse hosttan gelecek komutlar beklenmeye başlanır. Target mod seçildiyse initiator'dan atr_req beklenmeye başlanır.
- Send_atr_req komutu : İnitiator mod seçildiyse target'la haberleşmeyi başlatmak için kullanılması gereken ilk komuttur.
- Send_psl_req komutu : Bu komutu sadece initiator kullanabilir. Target ile haberleşme atr_req gönderip atr_res alma şeklinde başladıktan sonra bu komut istenilen zamanda kullanılarak alma ve gönderme hızı ile bir veri paketinin sahip olabileceği maksimum uzunluk belirlenebilir. Bu komutu kullanmadan önce w_pp_sup_target komutu ile target'ın desteklediği alıcı ve verici bit hızları ile bir veri paketinin sahip olabileceği maksimum uzunluk desteği parametreleri kontrol edilmelidir ve bu parametrelere uygun bir psl_req yollanmalıdır.
- W_pp_sup_target komutu : Bu komut ile data çıkışından bir baytlık PPt(Protocol Parameters used by Target) okunur. PPt bilgisine göre değiştirilebilecek parametreler belirlenir. PPt bilgisini target atr_res ile birlikte gönderir.
- W_brs komutu : Bu komut ile psl_req' in BRS baytı yazılmış olur.
- W_fsl komutu : Bu komut ile psl_req' in FSL baytı yazılmış olur.
- Send_dep_req komutu : Host üzerinde çalışan uygulamanın kullandığı ve gönderip almak istediği genel amaçlı veri paketlerini göndermek için kullanılan komuttur. write_general_info_byte_n ve make_chaining komutları ile beraber kullanılır. Bu komut kullanılınca NFCIP-1 modülü dep_req gönderir.
- Write_general_info_byte_n komutu : Gönderilmek istenen veri bu komut ile byte_no girişinde belirtilen adrese yazılır.
- Make_chaining komutu : Bu komut kullanılarak dep_req' in PFB baytının MI biti 1 yapılır ve gönderilen veri paketinin devamının olduğu ve bir veri paketine sığmadığını belirtir. Bu durumda paketi alan taraf ACK_PDU gönderir ve paketin devamını bekler.

- Read_general_info_length komutu : Alınan paketteki verinin uzunluğu okunur.
- Read_general_info_byte_n komutu : Alınan veri paketinin byte byte okunmasını sağlar. Bu komutla birlikte byte_no girişindeki adresin gösterdiği byte okunur.
- Send_dsl_req komutu : İnitiator bu komutu kullanarak veri alışverişi sonlandırılır. Eğer istenirse daha sonra wup_req yollanarak tekrar veri alışverişi yapılabilir.
- Send_wup_req komutu : dsl_req yollanarak haberleşme kesilmiş olan target ile bu komut kullanılarak tekrar haberleşme başlatılabilir.
- Send_rls_req komutu : Bu komut aracılığıyla haberleşme tamamen kesilmiş olur. Daha sonra wup_req kullanılamaz.
- Read_rec_cmd komutu : Bir veri paketi alındığında alınan komutu okumak için target bu komutu kullanır. Daha sonra alınan komuta uygun bir cevap verilir.
- Send_res komutu : Bu komutu target kullanır. Gelen veri paketine aynı komutla cevap vermek için kullanılır.
- Send_timeout_ext_req komutu : Bu komutu sadece target kullanır. Eğer alınmış olan veri paketinin işlenmesi zaman aşımı süresini aşacaksa target bu komutla gönderilen RTOX baytına bakarak zaman aşımı süresini Zaman $\text{Aşımı}_{\text{yeni}} = \text{Zaman Aşımı}_{\text{eski}} \times \text{RTOX}$ değerine çıkarır. Bu zaman aşımı değeri target veri paketini yolladıktan sonra eski değerine döndürülür.

4.1.1.2.Kontrol Modülü

NFCIP-1 modülünün merkezi kontrol birimidir. NFC protokolünün başlatılması, veri alışverişi, protokolün sonlandırılması, zaman aşımı belirleme, hosttan gelen komutların işlenmesi ve NFCIP-1 modülünün diğer alt birimlerinin kontrol edilmesi işlevlerini yerine getirir.



Şekil 4.3.Kontrol modülü.

Şekil 4.3' de blok diyagramı görülmektedir. Diğer modüllerle olan arayüzü ve kontrol sinyalleri şu şekildedir:

Host arayüzü :

- Reset girişi kullanarak NFCIP-1 modülü ve tüm alt modülleri resetlenebilir.
- Cmd_in girişi hostun komutları yazdığı giriştir.
- Data_in1 girişi kullanılarak komutlar için gerekli ek bilgiler ile genel amaçlı uygulama bilgileri girilir.
- Byte_no girişine host tarafından okunmak istenen veri adresi yazılır.

- Chaining çıkışı ile hosta gelen paket eğer MI biti '1' olarak geldiyse bu paketin devamı gelecek bilgisi verilir.
- Status çıkışı ile hosta herhangi bir NFCIP-1 modülü ile haberleşme halinde olup olunmadığı bilgisi verilir.
- Tout çıkışı zaman aşımı olup olmadığını hosta bildirir.

Veri Paketi Kontrol Modülü arayüzü:

- İnitiator_nfcid3 ve target_nfcid3 çıkışları sırasıyla initator ve target' ın kullandığı nfcid' leridir.
- Cmd_send ve do_frame çıkışları sırasıyla gönderilecek komut ile paket oluştur sinyalidir.
- FSL ve BRS çıkışları psl_req yollanacağı zaman kullanılacak olan FSL ve BRS baytlarıdır.
- PFB ve NAD çıkışı DEP sırasında kullanılan baytlardır. PFB kontrol ve NAD mantıksal adres baytıdır.
- DID çıkışı initiator' a ait DID ve PP çıkışı NFCIP-1' nin desteklediği protokol parametreleridir.
- Frame_error girişi Kontrol Birimine alınmış olan veri paketi ile ilgili hata varsa bu hata bilgilerini verilir.

Haberleşme Modülü arayüzü :

- Sended girişi gönderilmesi istenen veri paketinin gönderildiğini bildirir. sended girişi '1' olunca Kontrol Birimi Haberleşme Birimini receive moduna getirir.
- Receive ve send çıkışları haberleşme modülünü alma ve gönderme modlarına getirmek için kullanılan kontrol sinyalleridir.
- Rate_detected ve rate girişleri ile target atr_res beklerken initiator'un ilk veri gönderme ve alma hızının belirlendiği ve belirlenen bit hızının ne olduğu belirtilir.
- Rfcai2_done çıkışı Haberleşme Modülüne initiator modda iken atr_res komutunu aldıktan sonra gönderilecek komuttan önce yapılması gereken RFCA'nın yapıldığını bildirir.

Rastgele Erişimli Bellek Modülü arayüzü:

Read1 ve write1 çıkışları ile Rastgele Erişimli Bellek modülüne rw_addr1 çıkışının gösterdiği adresten okuma veya yazma yapılır.

Diğer giriş ve çıkışlar :

- Rec_rate ve send_rate çıkışları Haberleşme ve Veri Paketi Kontrol Modüllerine giden ve veri alış ve veri gönderme hızlarını gösteren çıkışlardır.
- Rst çıkışı ile zaman aşımı oluşup da target veya initiator ile haberleşme koparıldığında tüm alt modüller Kontrol Modülü tarafından resetlenir.

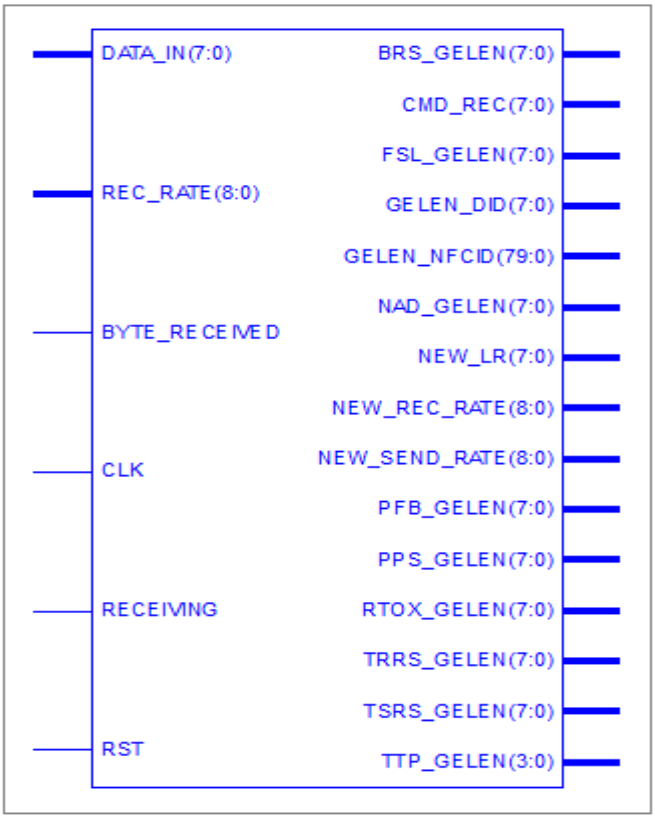
Kontrol Modülü yukarıda anlatılan giriş çıkışlar ve gerçekleştirdikleri işlevler ile tüm diğer alt birimleri kontrol eder. Kontrol modülü kodlama açısından kontrol işlevini yerine getiren altı alt modülden oluşacak şekilde tasarlanmıştır. Kontrol Modülü; kontrol birimi, alınan veri paketi çözücü, RFCA modülü, zaman aşımı belirleyici, alım kontrol modülü ve rastgele sayı üretici modüllerinden oluşur.

4.1.1.2.1.Kontrol Birimi

Kontrol birimi kontrol modülünün diğer alt modüllerinden gelen kontrol bilgileri doğrultusunda haberleşme, veri paketi kontrol modülü, rastgele erişimli bellek ve çift yönlü tampon modüllerini kontrol eder. Aynı zamanda hosttan gelen komutların işlenmesinden de sorumludur.

4.1.1.2.2.Alınan Veri Paketi Çözücü

Bu modül blok diyagramında gösterilen giriş ve çıkışlara sahiptir. İşlevi gelen pakete ait komutu belirleyerek bu komuta ve pakete ait olan ve Kontrol Modülünün kullanacağı bilgileri hazırlamaktır.



Şekil 4.4.Alınan veri paketi çözücü

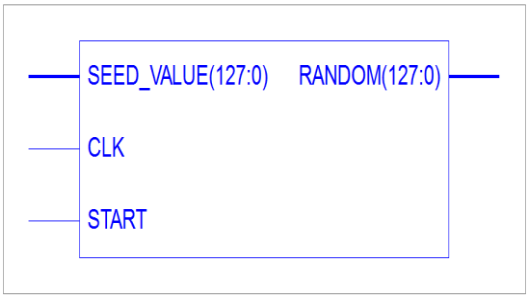
Giriş ve Çıkışları ve Çalışma Şekli :

- Clk girişi saat ve rst girişi Kontrol Modülünden gelen receiving ve byte_received girişleri Haberleşme Modülünden gelmektedir. receiving girişi sof alındıktan sonra '1' olur ve Alınan Veri Paketi Çözücü Modülü alınacak baytları beklemeye başlar. Bundan sonra gelecek olan baytlar beklenmeye başlanır. byte_received girişi '1' oldukça gelen baytlara data_in girişinden bakılarak cmd_rec ve cmd_rec' e bağlı olarak Kontrol Modülünün kullanacağı diğer bilgiler elde edilir. cmd_rec' i bulmak için alış hızını bilmesi gerekmektedir. Bunun için rec_rate girişi kullanılır.
- RTOX çıkışı Zaman Aşımı Belirleyici Modülüne gider. RTOX değeri zaman aşımı değeriyle çapılarak yeni zaman aşımı değeri elde edilir. Bu yeni değer sadece bir veri paketi alınıncaya kadar kullanılır ve sonra tekrar eski zaman aşımı değerine dönülür.
- Cmd_rec gelen veri paketinin içerdiği komuttur.
- Gelen_nfcid çıkışı gelen paket atr_req, atr_res veya wup_req ise paketi gönderen NFCIP-1' e ait nfcid'dir.
- Gelen_did çıkışı gelen paketteki DID baytıdır.

- Brs_gelen çıkışı cmd_rec psl_req olduğunda BRS baytıdır.
- Fsl_gelen çıkışı cmd_rec psl_req olduğunda FSL baytıdır.
- New_send_rate, new_rec_rate ve new_lr çıkışları psl_req'le gelen yeni gönderme ve alma hızları ile maksimum ver paketi uzunluğudur.
- Target_rec_rate_sup_gelen çıkışı gelen paket atr_res ise target' a ait bit alış hızı desteğini gösteren bayttır.
- Target_send_rate_sup_gelen, çıkışı gelen paket atr_res ise target' a ait bit gönderme hızı desteğini gösteren bayttır.
- Target_timeout_parameter_gelen, çıkışı gelen paket atr_res ise target' ın kullandığı zaman aşımı parametresidir.
- Pfb_gelen çıkışı DEP esnasında kullanılan kontrol baytıdır.

4.1.1.2.3.Rastgele Sayı Üreteci

Atr_req ve atr_res ile gönderilen ve rastgele olması gereken 10 baytlık nfcid ve RFCA uygulanırken kullanılan ve rastgele olması gereken 2 bitlik N değerini üreten modüldür.



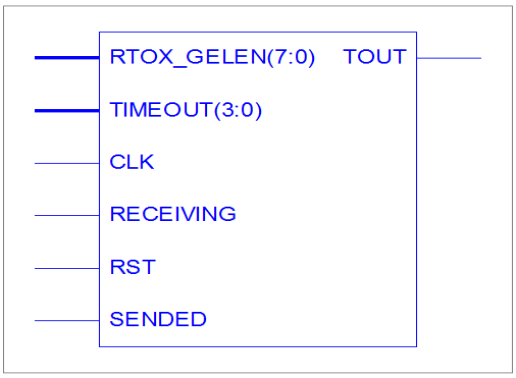
Şekil 4.5.Rastgele sayı üretici.

Giriş Çıkışları ve Çalışma Şekli :

clk girişi saat girişidir. start girişi '1' olunca seed_value girişindeki değeri random çıkışına atar ve rastgele sayı üretimine başlar. random çıkışı rastgele sayı çıkışıdır ve saatin her yükselen kenarında değişir.

4.1.1.2.4.Zaman Aşımı Belirleyici

Zaman aşımı belirleyici blok diyagramı Şekil 4.6' da gösterlmiştir.

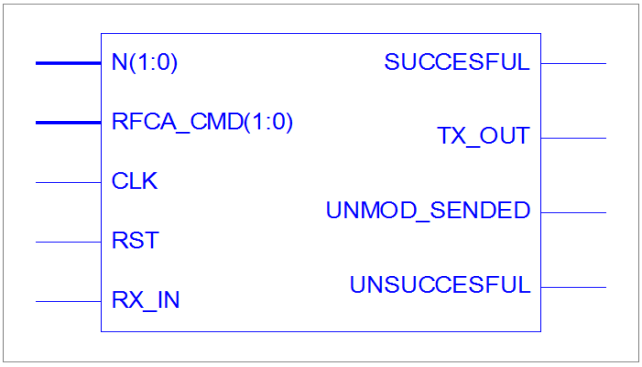


Şekil 4.6.Zaman aşımı belirleyici.

Zaman aşımı oluştuğunda Kontrol Modülünü uyaran modüldür. clk ve rst saat ve reset girişleridir. Haberleşme Modülünden gelen sended girişi '1' olduğunda gönderilmek istenen paket gönderilmiş demektir ve bu andan itibaren receiving girişi '1' olana kadar sayar. Sayma işlemi receiving '1' olmadan önce Kontrol Modülünden gelen timeout girişinden elde edilen zaman aşımı değerine ulaşırsa tout çıkışı '1' olur ve modül ilk durumuna döner. Eğer receiving zaman aşımı süresi dolmadan önce '1' olursa tout çıkışı '0' da kalır ve tekrar ilk duruma dönülür.

4.1.1.2.5.RFCA Modülü

RFCA modülü blok diyagramı Şekil 4.7' da gösterlmiştir.



Şekil 4.7.RFCA yapan modül.

clk ve rst sırasıyla saat ve reset girişleridir. Kontrol Modülü rfca_cmd girişine şu dört komuttan birini yazabilir :

- İdle : Bu komut gelirse modül hiç bir şey yapmadan aşağıdaki komutlardan birini beklemeye devam eder.
- Rfcai1 : Bu komutu initiator kullanır. Bu komut gelince T_IDT + N × T_RFW süresi boyunca RFCA uygular ve rfcai1 başarılı olursa yani rx_in

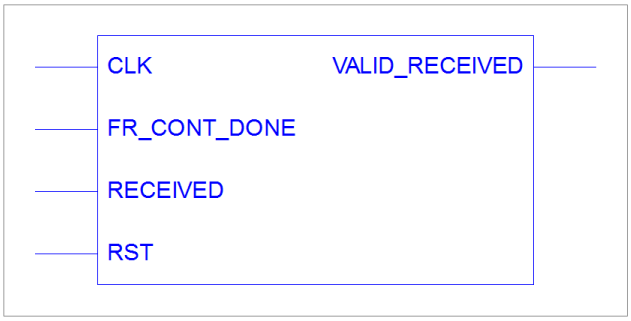
girişinden bu süre boyunca '0' alınırsa T_IRFG süresi boyunca modüle edilmemiş taşıyıcı tx_out çıkışı aracılığıyla yayınlanır. RFCA süresi boyunca 424 Kbps hızı ile örnekleme yapılır. RFCA yapılırken kullanılan 2 bitlik N değeri Rastgele Sayı Üreteci Modülünden gelir. Bu N değerinin rastgele seçilmesinin nedeni aynı anda RFCA yapmaya başlayan modüllerden N değeri büyük olanların RFCA sonucunun başarısız olarak, modüllerden sadece birinin aynı etki alanı içerisinde haberleşme yapabilmesini sağlamaktır.

- Rfcat : Bu komutu target kullanır. Atr_req alındıktan sonra bu komut kullanılır. T_ADT + N × T_RFW süresi boyunca RFCA uygulanır ve başarılı olursa T_ARFG süresi boyunca modüle edilmemiş taşıyıcı yayınlandıktan sonra atr_res yollanır. rfcai1' de olduğu gibi burada da N değerinin kullanılma amacı benzerdir. atr_req' i alan target' lardan sadece N değeri küçük olanın RFCA sonucu başarılı olacaktır.
- Rfcai2 : Bu komutu initiator kullanır. Atr_res alındıktan sonra T_ADT süresi boyunca ikinci bir RFCA uygulanır ve başarılı olursa T_ARFG süresi boyunca modüle edilmemiş taşıyıcı yayınlanır.

Rfcai1, rfcat ve rfcai2 komutları yerine getirilirken eğer RFCA başarılı olursa succesful çıkışı '1' yapılır aksi takdirde unsuccesful çıkışı '1' yapılarak Kontrol Birimi bilgilendirilir. Modüle edilmemiş taşıyıcı gönderme bittiğinde ise unmod_sended çıkışı '1' yapılır ve Kontrol Birimi bilgilendirilir.

4.1.1.2.6.Alım Kontrolü

Alım kontrolü blok diyagramı Şekil 4.8' da gösterlmiştir.



Şekil 4.8.Alım kontrolü

Bu modül bir veri paketi alındığında ve Veri Paketi Kontrol Modülü tarafından gerekli kontroller yapıldıktan sonra Kontrol Modülünü uyarır. clk ve rst saat ve reset

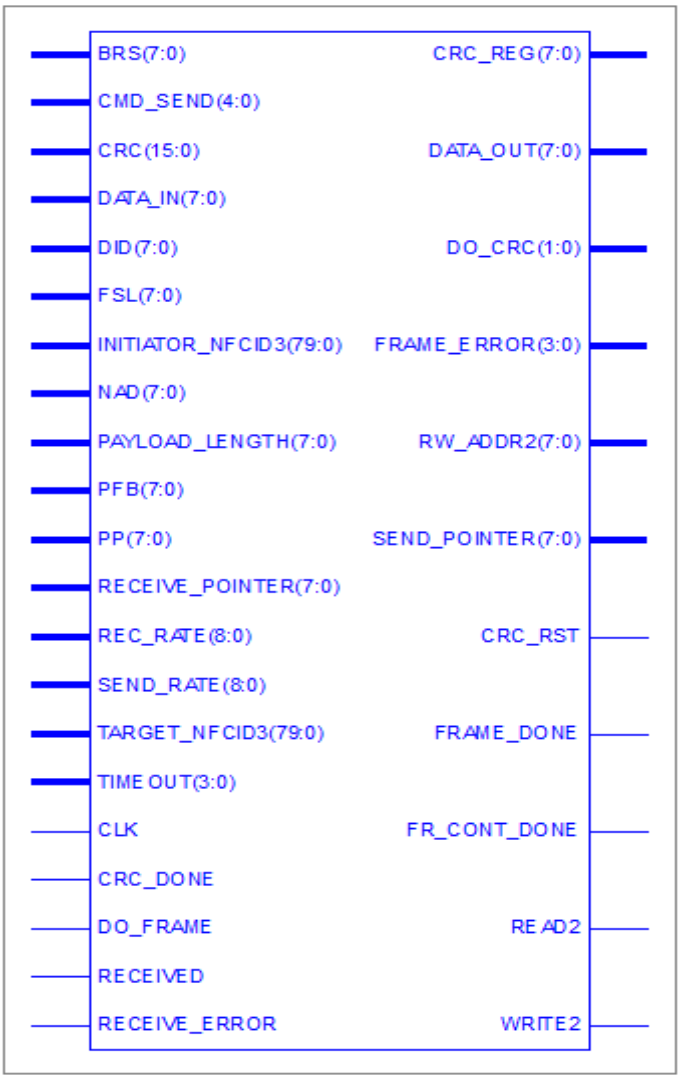
girişleridir. Received girişi haberleşme Modülünden gelir ve bir veri paketi alındığını bildirir. Received girişi '1' olduktan sonra Veri Paketi Kontrol Modülünden gelen fr_cont_done girişinin '1' olması beklenir. Bu giriş '1' olunca valid_received çıkışı '1' yapılarak Kontrol Modülü bilgilendirilir.

4.1.1.3.Veri Paketi Kontrol Modülü

Gönderilecek verinin ve komutun NFC formatına uygun hale getirilmesi ve alınan verinin CRC ve uzunluk kontrollerini yapar. Format Kontrol Birimi ve CRC Modüllerinden oluşur.

4.1.1.3.1.Format Kontrol Modülü

Format kontrolü modülü blok diyagramı Şekil 4.9' da gösterlmiştir.



Şekil 4.9.Format kontrol modülü.

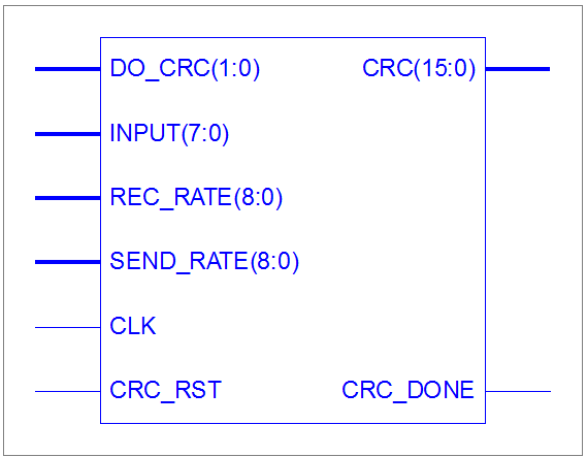
Gönderilecek veri paketinin NFC formatına uygun hale getirilmesi için gerekli başlık bilgileri (başlangıç baytı, senkronizasyon baytları ve uzunluğu gösteren bayt), gönderilen komut ile ilgili baytlar ve sonuna eklenmesi gereken CRC baytlarını ekler. Başlık baytları gönderme hızına göre değiştiği için send_rate girişini eklenecek başlık baytlarını belirlemek için kullanır. Aynı zamanda alınan pakete ilişkin CRC ve uzunluk kontrollerini yapar. rec_rate girişini alınan verinin başlık baytlarını belirlemek için kullanır.

do_frame girişi '1' olunca cmd_send girişinde gördüğü komuta uygun baytları başlık baytı olarak ekler. Komuta ilişkin baytlardan önce Veri paketinin sonuna eklenecek 2 CRC baytını hesaplarken Kontrol Modülünden gelen ve genel amaçlı veri uzunluğunu gösteren paylaod_length girişinden yararlanır. Gönderilecek komuta ilişkin baytlar ve genel amaçlı baytlar CRC işlemine sokularak elde edilen iki bayt write2, rw_addr2 ve data_out çıkışları aracılığıyla Rastgele Erişimli Belleğe yazılır. Bu işlem sona erince frame_done çıkışını '1' yapar ve send_pointer çıkışı gönderilecek verinin bellekteki son adresini gösterir. Bu çıkışları kullanan Haberleşme Modülü gönderilecek veri paketinin hazır olduğunu anlar ve bellekten ne kadar veri göndereceğini bilir.

Bir veri paketi alınması durumunda ise received girişi '1' olur. Bu durumda Haberleşme Modülünden gelen receive_pointer girişine bakarak gelen verinin bellekteki son adresini okumuş olur. Bu adres CRC baytlarını da içeren veri paketinin son adresi olduğu için receive_pointer - 2 adresine kadar CRC işlemini uygular. Bu şekilde CRC baytlarına da CRC işlemini uygulamaktan sakınılmış olur. CRC işleminin sonucu ile gelen verideki CRC baytlarını karşılaştırarak hata olup olmadığını tespit eder. Eğer hata var ise frame_error çıkışının üçüncü bitini '1' yapar. Gelen verinin uzunluğu ile LENGTH baytını karşılaştırarak uzunlukta hata var ise frame_error çıkışının ikinci bitini '1' yapar. Bu kontroller bitince fr_cont_done çıkışını '1' yapar.

4.1.1.3.2.CRC Modülü

CRC modülü blok diyagramı Şekil 4.10’ da gösterlmiştir.

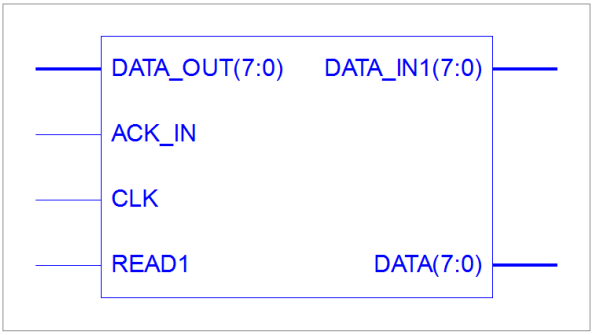


Şekil 4.10.CRC modülü.

Format Kontrol Modülü tarafından kontrol edilir. do_crc girişinden girilen komuta göre çalışır. Bu komutun durumuna göre çalışmadan bekleyebilir, alınan veriye CRC uygulayabilir veya gönderilecek veriye CRC uygulayabilir. Veri alma ve gönderme hızına göre uygulanacak CRC' nin başlangıç değeri değiştiği için send_rate ve rec_rate girişlerini bu ilk değeri belirlemek için kullanır. Bayt bayt giriş alarak her bayta CRC uygulayınca crc_done çıkışını '1' yapar. crc_rst girişi '1' olana kadar bir önceki CRC sonucunun üzerinden işlem yapmaya devam eder. crc_rst '1' olunca ilk durumuna geri döner. Sonucu CRC adlı iki baytlık çıkışına verir.

4.1.1.4.Çift Yönlü Tampon

Çift yönlü tampon blok diyagramı Şekil 4.11' de verilmiştir.



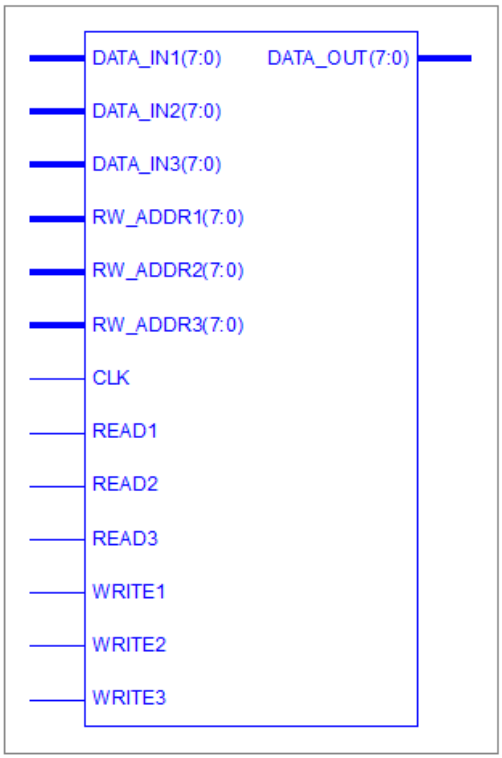
Şekil 4.11.Çift yönlü tampon.

Host ile çift yönlü veri alışverişinin tek bir port üzerinden yapılmasını sağlar. data bir baytlık çift yönlü giriş çıkıştır. data_in1 hostun data' ya yazdığı giriş ve data_out NFCIP-1' in data' ya yazdığı çıkıştır. read1 girişi Kontrol Modülünden gelir. read1 '1' olunca data' ya data_out' u yazar. Bu durumda host okuma yapar ve data_in1 yüksek empedans özelliği gösterir. Bu durum ack_in girişi host tarafından '1' yapılıncaya

kadar sürer. ack_in '1' olunca data data_in1' e yazılır ve hosttan gelecek veri okunabilir duruma geçilir.

4.1.1.5.Rastgele Erişimli Bellek

Rastgele erişimli bellek blok diyagramı Şekil 4.12' da gösterlmiştir.



Şekil 4.12.Rastgele erişimli bellek.

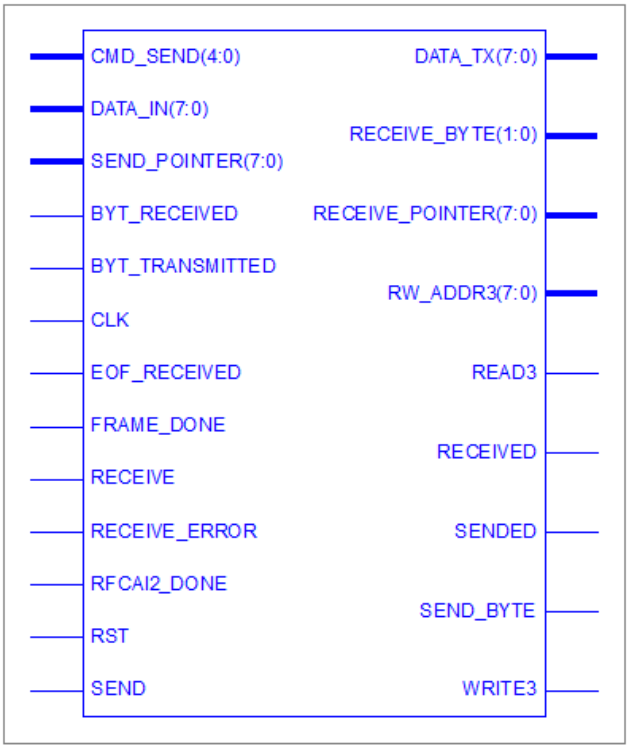
NFCIP-1 modülü için gönderilecek verinin ve alınan verinin yazıldığı bellek görevini görür. Yazma işlemi saatle senkron olmak zorunda iken okuma işlemi senkron değildir. Üç farklı yazma girişi (read1, read2, read3), üç farklı yazılacak veri girişi (data_in1, data_in2, data_in3), üç farklı adres seçme girişi (rw_addr1, rw_addr2, rw_addr3) ve bir veri çıkışı (data_out) vardır. Girişler sırasıyla Kontrol Modülü, Veri Paketi Kontrol Modülü ve Haberleşme Modülünden gelir. Bu girişlerin bellek girişlerine uygulanmasında belirtilen sırada bir öncelik vardır. Bu yöntemle aynı belleği her üç modülde ihtiyacı olduğunda kullanmaktadır.

4.1.1.6.Haberleşme Modülü

Veri paketi gönderiminin ve veri alışının yapıldığı modüldür. Alıcı Verici Kontrol Modülü, Alıcı Verici Modülü ile Bit Kodlayıcı ve Alış Hızı Belirleyici alt modüllerinden oluşur.

4.1.1.6.1.Alıcı Verici Kontrol Modülü

Alıcı verici kontrol modülü blok diyagramı Şekil 4.13' de verilmiştir.



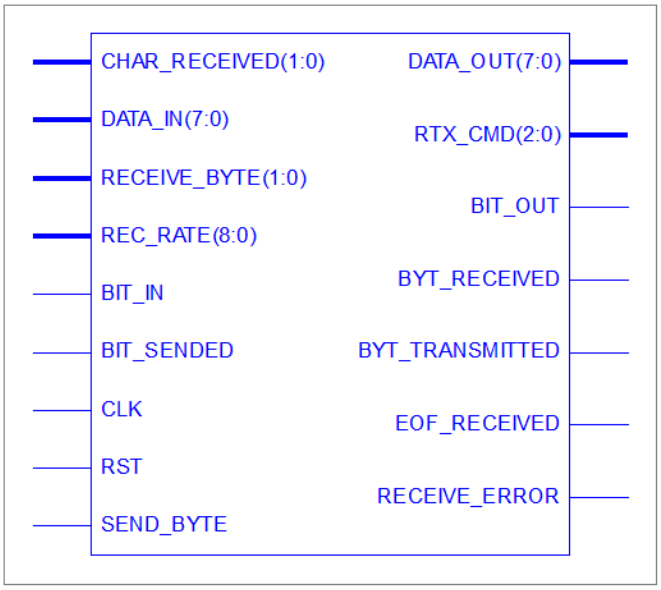
Şekil 4.13.Alıcı verici kontrol modülü.

Veri gönderme işlemi Kontrol Biriminden gelen send girişinin '1' olması ile başlar. Bu giriş '1' olunca Veri Paketi Kontrol Modülünden gelen frame_done girişinin '1' olması beklenir. Bu girişte '1' olduğunda bellekten veri okunmaya başlanır. Okunan her bayt Alıcı Verici Modülüne giriş olarak verilir ve send_byte çıkışı '1' yapılır. byte_transmitted girişi '1' olunca okuduğu yeni baytı Alıcı Verici Modülüne verir. Bu işlem send_pointer adresine kadar okuma yapılana kadar devam eder. Gönderme işlemi bitince sended çıkışı '1' yapılır.

Veri alma işlemi biraz daha farklıdır. Hem cmd_send hem de receive girişine bakılır. Öncelik cmd_send girişindedir. Bu giriş wait_unmodulated komutu ise Alıcı Verici Modülüne veri almadan önce bir süre boyunca modüle edilmemiş taşıyıcı beklendiğini bildirir. rec_with_detect_rate komutu ise alınan verinin bit hızının bilinmediği ve bu hızın belirlenmesi gerektiğini bildirir. Bu iki komuttan biri olmadığı durumlarda ise receive girişine bakılır. Eğer '1' ise bit alış hızının bilindiği ve normal alış yapılacağını anlar. Bu durumlara göre Alıcı Verici Modülüne receive_byte çıkışıyla komut verir.

4.1.1.6.2.Alıcı Verici Modülü

Alıcı verici modülü blok diyagramı Şekil 4.14' de verilmiştir.



Şekil 4.14.Alıcı verici modülü.

Alıcı Verici Kontrol Modülünden gelen send_byte ve receive_byte girişlerine bakarak girişindeki bir baytı Bit Kodlayıcı ve Alış Hızı Belirleyici Modülü aracılığıyla gönderir veya aynı modülden gelen bitleri alır.

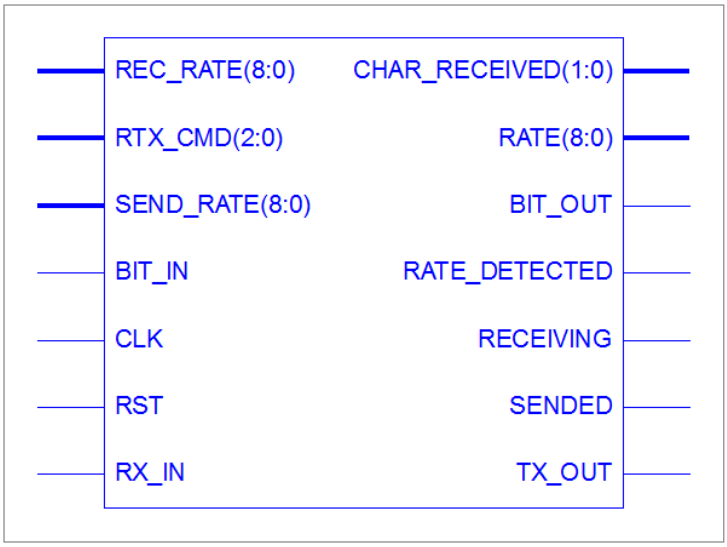
send_frame girişi '1' ise girişindeki baytı birer bit birer bit Bit Kodlayıcı ve Alış Hızı Belirleyici Modülü aracılığıyla gönderir. Bit Kodlayıcı ve Alış Hızı Belirleyici Modülünden gelen bit_sended girişi bir bitin kodlanarak gönderildiğini gösterir ve gönderilecek yeni bir bit bit_out çıkışına verir. Her baytı gönderdikten sonra byt_transmitted çıkışını '1' yaparak Alıcı Verici Kontrol Modülünü uyarır ve eğer send_frame girişi halen '1' ise data_in girişindeki baytı alır ve gönderme işlemine devam eder.

Veri alma işleminde ise receive_byte girişine Alıcı Verici Kontrol Modülü tarafından yazılan komuta bakar. Bu komutun Alıcı Verici Kontrol Modülünde anlatılan durumlarına göre Bit Kodlayıcı ve Alış Hızı Belirleyici Modülüne ya normal alış, ya modüle edilmemiş taşıyıcı bekleyerek alış ya da bit alış hızı belirleyerek alış yapması gerektiğini rtx_cmd çıkışı ile söyler. Bundan sonra Bit Kodlayıcı ve Alış Hızı Belirleyici Modülünden gelen char_received girişine bakar. char_received girişi busy durumunda ise bit alma işlemi devam ediyor demektir ve Alıcı Verici Modülü bekler. Eğer char_received girişi alınan işaretin veri olduğunu (char_received = data

durumu) söylüyorsa gelen biti bit_in girişinden alır ve alınan bit sayısı sekize ulaşınca byt_received çıkışını '1' yaparak Alıcı Verici Kontrol Modülünü uyarır. Bit alışında hatalı kodlanmış bir bit alındıysa char_received = hata olur. Bu durumda receive_error çıkışı '1' yapılarak hatalı bir veri paketi alındığı Alıcı Verici Kontrol Modülüne bildirilir. char_received girişinin eof olması durumunda Veri Sonu Karakterinin alındığı anlaşılır. Ancak bu durumda eof' un bir baytın sonunda gelip gelmediğine bakılır. Eğer baytın sonunda geldiyse eof_received çıkışı '1' yapılarak Alıcı Verici Kontrol Modülü bilgilendirilir. Eğer bayt ortasında geldiyse receive_error çıkışı '1' yapılarak hatalı bir veri paketi alındığı Alıcı Verici Kontrol Modülüne bildirilir.

4.1.1.6.3.Bit Kodlayıcı ve Alış Hızı Belirleyici Modülü

Bit kodlayıcı ve alış hızı belirleyici modülü blok diyagramı Şekil 4.15' de verilmiştir.



Şekil 4.15.Bit kodlayıcı ve alış hızı belirleyici modülü.

Bit bit veri alma ve veri gönderme ile target için initiatorun ilk veri alma ve gönderme hızını belirleyen modüldür. Alıcı Verici Modülünden gelen rtx_cmd komut girişine göre ya bit gönderir, ya bit alır ya da ilk hız belirleyerek bit alır.

Rtx_cmd komutu send_frame ise önce sof gönderir ve ardından bit_in girişindeki biti uygun şekilde kodlayarak (send_rate göre ya modified_miller ya manchester kodlama ile) gönderir. Biti gönderince bit_sended çıkışını '1' yaparak Alıcı Verici Modülünü uyarır. Eğer rtx_cmd halen send_frame ise bit_in girişindeki bitleri göndermeye devam eder. Herhangi bir bit gönderildikten sonra rtx_cmd girişi rtx_idle1 ise eof gönderilir ve gönderme işlemi sona erer.

Rtx_cmd ile verilen receive_frame ise önce rec_rate ile belirtilen hıza uygun kodlanmış şekilde sof ' un gelmesini bekler ve sof gelince bit bit veri alımı başlar. Her bit alındığında char_received çıkışına data yazılarak Alıcı Verici Modülüne bir bit alındığını bildirir ve alınan biti bit_out çıkışına yazar. Doğru şekilde kodlanmış bitler geldiği sürece bu işlem eof alınana kadar devam eder. Eof alınınca char_received çıkışına eof yazılarak Alıcı Verici Modülü bilgilendirilir ve bit alma işlemi sona erer. Eğer bit alma işlemi sırasında yanlış kodlanmış bir bit alınırsa char_received çıkışına hata yazılarak Alıcı Verici Modülü bilgilendirilir ve bit alımı sona erer.

Rtx_cmd komutu wait_unmod ise modüle edilmemiş taşıyıcı beklenir. Modüle edilmemiş taşıyıcı gelince sof beklenir ve sof gelince bit alma işlemine geçilir.

Rtx_cmd komutu detect_rate ise modüle edilmemiş taşıyıcı beklenir. Aynı zamanda initiatorun bit alışveriş hızı da belirlenir. Bu durumda iken rx_in girişinden düşen kenar geldikten sonra 424 Kbps hızında örnekleme yapılarak gelen ilk dört bite bakılır. Eğer "1100" alındıysa hız 106 Kbps' tir, "0000" ise 212 Kbps' tir ve eğer "0100" ise 424 Kbps' tir. Hız belirlenince belirlenen hız rate çıkışına yazılır ve rate_detected çıkışı '1' yapılarak Kontrol Birimi uyarılır. Kontrol Birimi bu hızı send_rate ve rec_rate çıkışlarına yazar. Hız belirlendikten sonra sof alınır ve bit alma işlemiyle devam edilir.

4.2.AES Algoritmasını Gerçekleyen Modül

AES algoritmasını gerçekleyen modül [6] numaralı tezden hazır olarak alınmıştır. AES modülü şifreleyici ve şifre çözücü modüllerinden oluşmaktadır.

4.2.1.Şifreleyici Modül

Şifreleyici modülünün blok diyagramı Şekil 4.16'da gösterilmiştir.

Şekil 4.16.Şifreleyici.

Şifreleyici start girişi ‘1’ olunca key girişinden anahtarı ve plaintext girişinden şifrelenecek veriyi alır. Bölüm 3.2’de anlatılan işlemleri uygular ve 10 saat periyodu sonra şifrelenmiş veri ciphertext çıkışında oluşur. Ready çıkışını ‘1’ yaparak şifreleme işleminin bittiğini haber verir.

4.2.2.Şifre Çözücü Modül

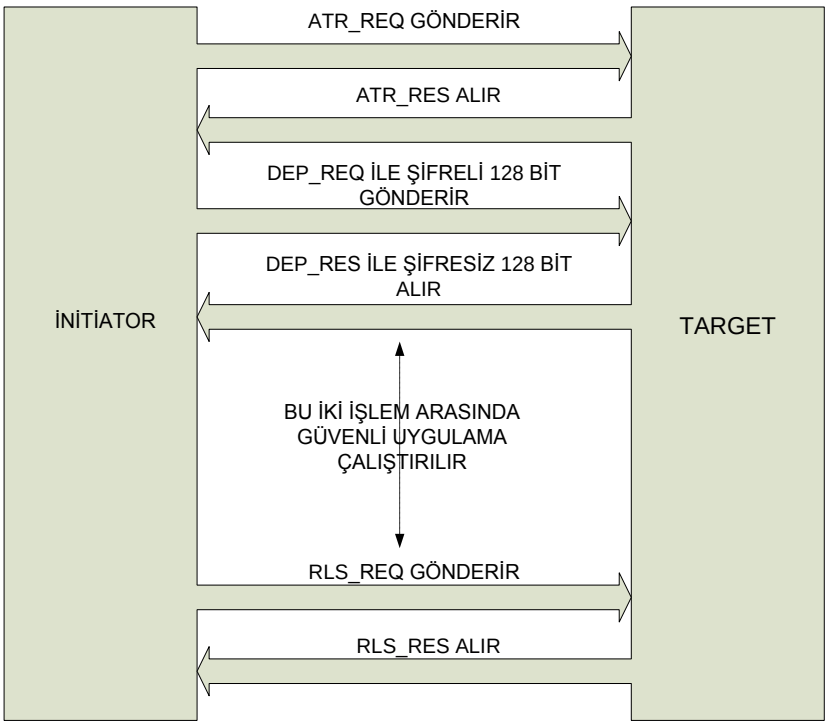
Şifre çözücü modülünün blok diyagramı Şekil 4.17’da gösterilmiştir.

Şekil 4.17.Şifre Çözücü.

Şifre çözücü start girişi ‘1’ olunca key girişinden anahtarı ve ciphertext girişinden şifresi çözülecek veriyi alır. Bölüm 3.2’de anlatılan işlemleri uygular ve 10 saat periyodu sonra şifresiz veri plaintext çıkışında oluşur. Ready çıkışını ‘1’ yaparak şifre çözme işleminin bittiğini haber verir.

4.3.MicroBlaze'in Sisteme Eklenmesi

Tasarlanan NFC modülü ve AES; MicroBlaze işlemcisi de eklenerek ISE Project Navigator ortamında bir üst modül altında birleştirilmiştir. MicroBlaze eklenerek elde edilen sistemin Şekil 4.18'deki prosedüre göre çalışacağı düşünülmüştür.



Şekil 4.18.İnitiator ve Target Arası Güvenli Haberleşme.

Kullanılan donanım sentezleme yazılımı ortamında hem initiator hem target aynı MicroBlaze' e çevresel donanım olarak eklenmiştir. İnitiator Şekil 4.18'e göre önce atr_req gönderip atr_res alarak haberleşmeyi başlatır. Haberleşme başlatıldıktan sonra MicroBlaze 128 bitlik bir sayıyı şifreleyici modülü kullanarak şifreler ve initiator' a bu şifreli veriyi dep_req komutu ile göndermesini söyler. Target bu şifreli veriyi alır ve MicroBlaze'i kesme kullanarak uyarır. MicroBlaze şifreli veriyi alır ve şifre çözücüyü kullanarak açık veriyi elde eder. Target'a şifresi çözülmüş veriyi dep_res ile göndermesini söyler. İnitiator şifresi çözülmüş 128 bitlik veriyi alır ve MicroBlaze'i kesme ile uyarır. MicroBlaze bu veriyi alır ve şifreleyip göndermiş olduğu şifresiz 128 bitlik sayı ile karşılaştırır ve aynı olduğunu görür. Bu durumda initiator target ile ortak anahtara sahip olduğunu anlar. Güvelik gerektiren uygulamalar bu prosedür sonrası çalışmaya başlar. İstenirse uygulama içinde de şifreli veri alışverişi yapılabilir. Uygulama bitirildikten sonra rls_req yollayıp rls_res alarak haberleşme sona erdirilir.

4.4. Sistemin Test Edilmesi

Şekil 4.18'deki prosedüre göre sistem için simülasyon yapılmıştır ve Şekil 4.19'daki dalga formu elde edilmiştir. Dalga formlarına bakıldığında initiatorun öncelikle RFCA uygulayarak haberleşmeyi 106 Kbps (460 saat periyodunda bir bit) ile başlatıp psl_req kullanarak hızı 212 Kbps (230 saat periyodunda bir bit)'e çıkardığı görülür. Daha sonra 128 bitlik veri şifrelenip dep_req ile gönderilmiştir. Target aldığı şifreli veriyi çözüp göndermiştir. Dalga formlarında initiator tarafından gönderilen şifreli verinin (sifrelenmis_veri olarak gösterilen sinyal) target tarafından hatasız alındığı (ciphertext_inv sinyali) ve şifresi çözülen verinin (plaintext_inv sinyali) initiator tarafından hatasız alındığı(gelen sinyali) görülmektedir. Sonuç olarak kimlik doğrulama gelen isimli 128 bitlik sayı ile plaintext_cipher'ın aynı olmasıyla tamamlanmıştır. Son olarak rls_req gönderip rls_res alarak haberleşme bitirilmiştir.

Şekil 4.19. Simülasyon sonucu.

5.SONUÇLAR

Çalışma boyunca hedeflenen güvenli bir yakın alan haberleşmesi sisteminin tasarımı olmuştur. Bu hedef doğrultusunda öncelikle NFC protokolü VHDL donanım tasarım diliyle tasarlanmıştır. [6] numaralı tezden alınan AES şifreleyici ve şifre çözücü ile birlikte sisteme MicroBlaze işlemcisinin de eklenmesiyle güvenli bir yakın alan haberleşme sistemi elde edilmiştir. Elde edilen sisteme ilişkin simülasyonlar yapılmış ve simülasyon sonuçları başarılı olmuştur. Bu sistem ile işlemci üzerindeki herhangi bir güvenli olması gereken uygulama gerekli kimlik doğrulama işlemini yaptıktan sonra çalıştırılabilir. Böylece kısa etkileşim mesafesi dolayısı ile bir miktar güvenli olan NFC daha da güvenli hale getirilmiş olur.

KAYNAKLAR

[1] **Spartan 3E Starter Kit Board User Guide** : http://www.xilinx.com.

[2] **MicroBlaze Processor Reference Guide** : http://www.xilinx.com.

[3] **ECMA-340,** 2008. Near Field Communication - Interface and Protocol (NFCIP-1) , *ECMA International*.

[4] **Paus, A.,** 2007. Near Field Communication in Cell Phones, pp. 6,7.

[5] **FIPS 197**, 2001. Advanced Encryption Standard, *National Institute of Standarts and Technology*.

[6] **Özkan, A.,** 2011. Gsm Ağı Üzerinden Güvenli Ses İletim, Lisans Tezi, İTÜ Elektronik Mühendisliği, İstanbul.

ÖZGEÇMİŞ

1989 yılında Malatya'da doğan İsmail DEMİR orta öğrenimini 2007 yılında Bolu Fen Lisesi'nde tamamladı. Aynı yıl İstanbul Teknik Üniversitesi Elektronik Mühendisliği Bölümü'ne girdi. Gömülü sistemler ve yazılım alanına ilgi duymaktadır.