İSTANBUL TEKNİK ÜNİVERSİTESİ

ELEKTRİK-ELEKTRONİK FAKÜLTESİ

BİR RFID TAKİP ETME SİSTEMİ İÇİN OKUYUCULARI SÜRECEK BİR ALT SİSTEMİN FPGA ÜZERİNDE GERÇEKLENMESİ

Bitirme Ödevi

Oğuzhan ÇİK

040100413

Bölümü: Elektronik Haberleşme Mühendisliği

Programı: Elektronik Haberleşme Mühendisliği

Danışmanı: Doç. Dr. Berna ÖRS YALÇIN

Mayıs 2015

ÖNSÖZ

Bana sayısız değer katan okuluma, yardımını esirgemeyen ve her konuda anlayışlı davranan danışman hocam Doç. Dr. Berna ÖRS YALÇIN'a, projenin diğer kısımlarında yer alan arkadaşlarım Alp ORAN'a, Bahadır GÜN'e, Onur AZBAR'a, Ramazan AKTAŞ'a, okuldaki diğer arkadaşlarıma ve son olarak benim bu günlere gelmemde en büyük paya sahip değerli aileme sonsuz teşekkürlerimi sunarım.

Mayıs 2015 Oğuzhan ÇİK

İÇİNDEKİLER

Sayfa

ÖNSÖZ ... **ii**

İÇİNDEKİLER ... **iii**

KISALTMALAR ... **v**

ŞEKİL LİSTESİ ... **vi**

TABLO LİSTESİ ... **viii**

ÖZET ... **ix**

SUMMARY ... **x**

1 GİRİŞ ... **1**

2 ÖNBİLGİLER ... **4**

2.1 Seri Çevresel Arayüz Haberleşme Protokolü ... 4

2.2 Kesme ... 7

2.3 Sahada Programlanabilir Kapı Dizisi ... 8

2.4 Microblaze İşlemci ... 10

2.5 RFM23B Modülü ... 13

2.6 Kullanılan Ölçüm Düzenekleri ... 16

2.6.1 Nexys 3 FPGA Geliştirme Kartı ... 16

2.6.2 Agilent 16802A Lojik Analizör ... 17

3 MODÜLÜN FPGA İLE SÜRÜLMESİ ... **19**

3.1 Microblaze İşlemci Konfigürasyonu ve Yan Donanımlar ... 19

3.1.1 Microblaze Konfigürasyonu ... 19

3.1.2 Seri Çevresel Arayüzü Yan Donanımı ... 22

3.1.3 Kesme Kontrolü Yan Donanımı ... 24

3.2 RF22 Kütüphanesinin Mikroblaze İşlemcisi için Hazırlanması ... 26

3.2.1 Donanımsal SPI Sınıfı ... 27

3.2.2 RF22 Sınıfı ... 31

4 SONUÇLAR ... **37**

KAYNAKLAR ... **38**

EKLER.....40
EK-A.....40
EK-B.....41

ÖZGEÇMİŞ.....42

KISALTMALAR

GPS: Küresel Konumlandırma Sistemi (Global Positioning System)

RFID: Radyo Frekansı Kimliklendirme (Radio Frequency Identification)

SPI: Seri Çevresel Arayüz (Serial Peripheral Interface)

CPOL: Saat Polaritesi (Clock Polarity)

CPHA: Saat Fazı (Clock Phase)

MOSI:Birincil Çıkışı Slave Girişi (Master In Slave Out)

MISO: İkincil Girişi Slave Çıkış (Master In Slave Out)

ISR: Kesme Servis Rutini (Interrupt Service Routine)

FPGA: Sahada Programlanabilren Kapı Dizisi (Field Programmable Gate Array)

LUT: Doğruluk Tablosu (Lookup Table)

UCF: Kullanıcı Kısıtı Dosyası (User Constraints File)

IP: Akıllı Özellik (Intellectual Property)

USB: Evrensel Seri Yol (Universal Serial Bus)

UART: Evrensel Asenkron Alıcı Verici (Universal Asynchronus Receiver Transimitter)

EDK: Gömülü Geliştirme Seti (Embedded Development Kit)

SDK: Yazılım Geliştirme Seti (Software Development Kit)

MB: Megabayt

KB: Kilobayt

ŞEKİL LİSTESİ

Sayfa

Şekil 1.1: RFID Etiket ve Okuyucuların İç Mekanda Dizilimi 2
Şekil 1.2: Projenin Parçaları 2
Şekil 1.3: Tüm Sistemin Çalışma Diyagramı 3
Şekil 2.1: SPI Birincil ve İkincil Cihazları ve Bağlantı Pinleri [7] 4
Şekil 2.2: SPI Biricil Cihaza Birden FazlaİkincilBağlanması [7] 6
Şekil 2.3: İkincil Cihazların Kaskat Bağlanarak Birincil Cihaza Bağlantı Yapılması [7] 6
Şekil 2.4: Kesmenin İşleyişi 7
Şekil 2.5: FPGA İç Yapısı 8
Şekil 2.6: Bir CLB Yapısı 9
Şekil 2.7: EDK Proje Başlangıcı 11
Şekil 2.8: Microblaze İşlemcinin Temel Ayarları 11
Şekil 2.9: Microblaze Ana Proje Ekranı ve IP Kataloğu 12
Şekil 2.10: Yazılım Geliştirilecek SDK Ortamı 13
Şekil 2.11: RFM22 RF Modülü Bağlanması [12] 14
Şekil 2.12: RFM22 İstenilen Kayıt (Register) Adresine Yazma İşlemi [12] 14
Şekil 2.13: RFM22 İstenilen Kayıt (Register) Adresinden Okuma İşlemi [12] 15
Şekil 2.14: RFM23 İstenilen Adrese Yığın Yazma (Burst Write) Modu [12] 15
Şekil 2.15: RFM23 İstenilen Adrese Yığın Okuma (Burst Read) Modu [12] 15
Şekil 2.16: Nexys3 FPGA Geliştirme Kartı [14] 16
Şekil 2.17: Agilent 16802A Lojik Analizör [15] 17
Şekil 2.18: SPI Haberleşmesini İncelemek Amacıyla Kurulan Ölçüm Düzeneği 18
Şekil 3.1: Microblaze İşlemcinin RF22 Modül için Kayan Nokta Ünitesinin Açılması 20
Şekil 3.2: Kart Üzerindeki Kullanılacak Donanımların Eklenmesi 21
Şekil 3.3: Saat Üreteci Üzerinden Mikroblaze İşlemcinin Saat Frkeansı Ayarları 21
Şekil 3.4: SPI IP'si Ayarları 22
Şekil 3.5: SPI IP'sinin İşlemci Dışındaki Kısımlar ile Bağlantılarının Yapılması 23
Şekil 3.6: Dışarıya Açılan Pin Olarak Yapılandırılan SPI IP Pinlerinin PMOD ile Bağlanması 23
Şekil 3.7: Kesme Kontrolü IP'si Ayarları 24
Şekil 3.8: İşlemci ile kesme IP'sinin MHS Dosyası Üzerinden Bağlanışı 25
Şekil 3.9: "Ports" Menüsünden Butonun Kesme Olarak Ayarlanması 25
Şekil 3.10: Kesme Sisteminin Bağlantılar Tamamlandıktan Sonraki Durumu [18] 26
Şekil 3.11: UCF Dosyasında Buton Yerine PMOD Pinlerine Bağlantı Yapılması 26
Şekil 3.12: Donanımsal SPI Sınıf Diyagramı 27

Şekil 3.13: Saat Fazının 1 Olması Durumu 27
Şekil 3.14: Donanımsal SPI Sınıfı "begin" Fonksiyonu 28
Şekil 3.15: "setSS" Fonksiyonu 29
Şekil 3.16: Transfer Fonksiyonun Birinci Versiyonunun SPI Haberleşmesi 29
Şekil 3.17: Transfer Fonksiyonun İkinci Versiyonunun SPI Haberleşmesi 30
Şekil 3.18: Lojik Analizör Yardımıyla Modülden Okunan Değerin Kontrol Edilmesi 30
Şekil 3.19: Arduino'ya (sağda) ve Mikroblaze işlemciye (solda) ait kod parçaları 31
Şekil 3.20: “spiWrite” Fonksiyonu 32
Şekil 3.21: “spiBurstRead” Fonksiyonu 32
Şekil 3.22: “spiBurstWrite” Fonksiyonu 32
Şekil 3.23: “reset” Fonksiyonu 33
Şekil 3.24: Kesme IP'si ile RF22 Sınıfının Kesme Rutini Fonksiyonu Bağlantısı 34
Şekil 3.25: “isr0” Fonksiyonu 34
Şekil 3.26: Hafıza Ayarlanının Yapılandırılması için Linker Script Menüsünün Açılması 35
Şekil 3.27: Linker Script Menüsü ve Hafızanın Seçilmesi 35
Şekil 3.28: Önerilen Yeni Data Toplama Ünitesi 36

TABLO LİSTESİ

Sayfa

Tablo 2-1:SPI Modları 5

ÖZET

Bu proje herhangi bir acil durum anında bina içinde hangi noktalarda insanların kaldığını ve yine binanın tamamen boşaltılıp boşaltılmadığını belirlemek için kullanılacaktır. Bu şekilde büyük bir bina içinde acil bir durum anında kurtarma ekiplerinin işi kolaylaştırılarak hızlıca binanın tahliyesi sağlanabilecektir. Bu işlem için bina içinde konum belirlenmesi gerekmektedir. Ancak günümüzde yaygın olarak kullanılan konum belirleme sistemlerinden birisi olan GPS iç mekanda konum belirleme için yetersiz kalmaktadır. Bir bina içinde konum belirleme için en uygun çözümlerden birisi de RFID okuyucu ve etiketler kullanmaktır. Bu RFID okuyuculardan gelen veriler ile bilgisayar veya harici başka bir sistemde uygun bir algoritma kullanarak iç mekânda konum belirlemek mümkündür. Ancak bu sistem büyük bir bina veya tesis için düşünüldüğünden artan okuyucu sayısı nedeniyle bilgisayar işlemcisi hız açısından yetersiz kalabilmektedir. Bu sebeple sistemin işlem birimi FPGA üzerinde, bilgisayar işlemcisinden ayrı bir donanım olarak tasarlanmıştır. Bu donanım Microblaze ile tasarlanan sanal bir işlemci tarafından kontrol edilmektedir ve konum bilgilerini hesaplayarak bilgisayar ortamındaki bir kullanıcı arayüzüne aktarmaktadır. Bu kullanıcı arayüzünde gelen konum bilgileri izlenmektedir. Ayrıca okuyuculardan gelen konum bilgilerini işlem birimine aktaran ve yine Mikroblaze işlemci kontrolünde bir data toplama birimi de mevcuttur. Bu ünitede bu işlemci okuyucudan gelen verileri işlem birimine iletecek donanımsal haberleşme protokollerini kontrol etmektedir. Bu haberleşme protokolleri tasarlanmamış, Microblaze sisteminin içinde bulunanan hazır IP'ler kullanılmıştır. Ayrıca okuyucudan kesme yardımıyla hatalar veya tamamlanan işler mikroişlemci tarafından denetlenmektedir. Mikroişlemci üzerinden çalışacak yazılım ise projede kullanılan RFID okuyucunun Arduino için geliştirilmiş kütüphanesi baz alınarak yazılmıştır.

SUMMARY

This project aims to detect where the people are in a specific building and if whole building are evacuated. Thus, this system makes not only the rescue crew's work easy but also the evacuation swift. For this evacuation, the system requires indoor localization subsystem. However, GPS ,which are one of the most common used positioning systems, are not sufficent for indoor localization. Therefore, RFID readers and tags, that are one of the alternative solutions instead of GPS, can be used. The location of the reader in a building can be found by using a proper algorithm working on a processing unit. Number of the readers increses in case of using this system in massive buildings and facilities. As a result of this, the processor of the computer may be inadequate. To overcome this problem, the processing unit of the indoor localization system must be a seperate hardware designed on FPGA. The processing unit transfers the location information calculated on it ot the user interface on a computer to just monitor. Besides, there is a data collection unit that reads the data from RFID readers and sends it to the processing unit. The data collection unit is also controlled by a Microblaze processor that includes communication protocols between reader and itself. These protocols are ready to use instead of implementing. In the date collection unit Microblaze uses interrupts to find completed processes and overcome the exceptions during the communication between reader and FPGA in the data collection unit.

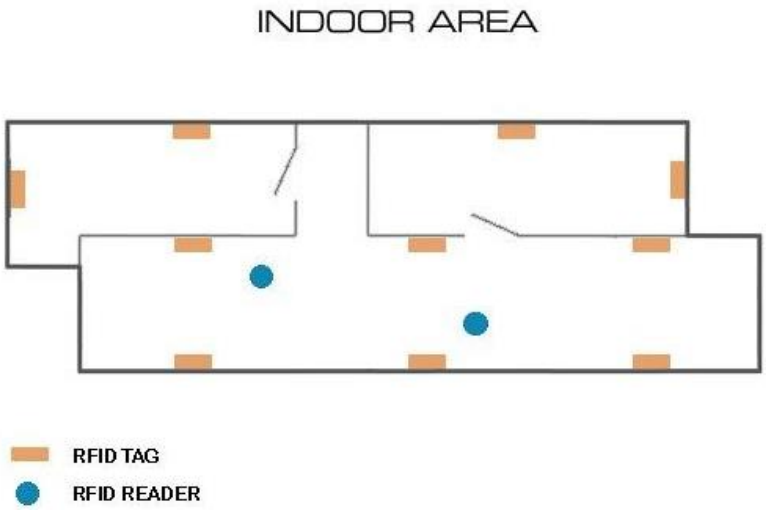
1 GİRİŞ

Günümüzde herhangi bir nesnenin veya kişinin konumunun çeşitli yöntemlerle belirlenmesi oldukça yaygınlaşmıştır [1]. Konum belirleme genel olarak iç ve dış mekanda olmak üzere iki ana başlık altında incelenebilir. Dış mekanda konum belirlemek için yaygın olarak küresel konumlama sistemi (Global Positioning System-GPS) [1-2] ve küresel uydu seyir sistemi (Global Navigation Satellite System-GLONASS) [3] kullanılmaktadır. Bu sistemlerde üç boyutta konum belirlemek için en az dört uydu gerekmektedir. Ancak dış mekanda konum belirlemede başarılı sonuçlar vermesine rağmen GPS veya GLONASS, iç mekanda konum belirlemek için yetersiz kalmaktadır çünkü bu sistemler uydu ile haberleşme gerektirmektedir [2]. İç mekanda uydu haberleşmesi girişim ve gölgeleme etkileri yüzünden dış mekadakine kıyasla daha zor sağlanabilmektedir. Radyo frekansı kimlikleme (Radio Frequency Identification - RFID) tabanlı konumlama sistemleri de yetersiz kalan bu sistemlere alternatif olarak geliştirilmiştir [2]. Konum belirlemek için çeşitli RFID temelli metodlar kullanılır. Bunlardan en yaygın kullanılanı ise GPS'teki metoda benzer bir şekilde çalışan üçgenleme metodudur. Üçgenleme metodunda da aynı GPS'de olduğu gibi en az dört RFID etiketi gereklidir. Bu etiketler GPS sistemindeki uydular olarak düşünülebilir.

Avrupa Birliği tarafından Servis Olarak Bina (Building as a Service-BaaS) [4] projesi kapsamında yürütülen proje, teknoloji ile donatılmış binalar oluşturmayı amaçlamaktadır. Bu projeye göre binalar ayrı sistemler ile izlenip yönetilecektir. Bu proje, yangın dedektörü, ışıklandırma ve güvenlik geçiş kontrolü gibi çeşitli parçalara sahiptir. Ayrıca bu sistem, acil bir durumda binanın hızlı bir şekilde tahliyesini sağlayacak, bina içindeki kişilerin sayısını ve konumunu belirleyecek iç mekanda konum belirleyen bir sistemi de içermektedir. Bu sistem acil bir durum anında bina içindeki kişilerin konumlarının belirlenerek binanın hızlı ve güvenilir bir şekilde tahliye edilmesini amaçlamaktadır.

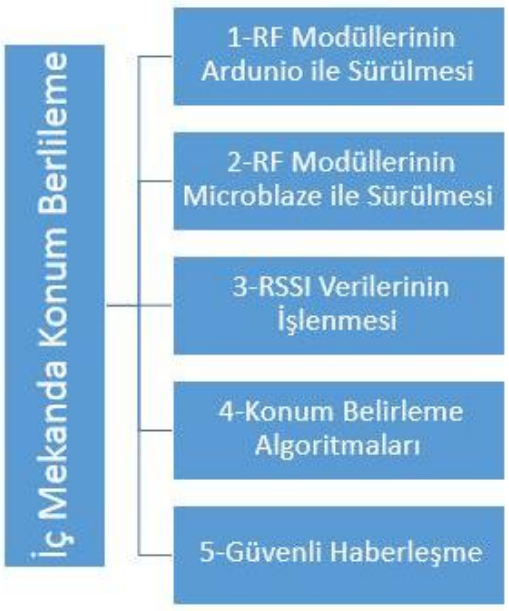
Bu sistemde duvarda sabit RFID etiketler, konumu belirlenecek RFID okuyucular olacaktır (Şekil 1.1). Ayrıca okuyucudan gelecek verileri işleyecek bir işlem birimi

olacaktır. İşlem birimi okuyuculardan gelen verileri işleyerek konum belirleme işlemini gerçekleştirecektir. Bu işlem birimi bir bilgisayar üzerinde gerçeklenebilir. Ancak sistem büyük bir bina veya tesis için düşünüldüğünde artan etiket ve okuyucu



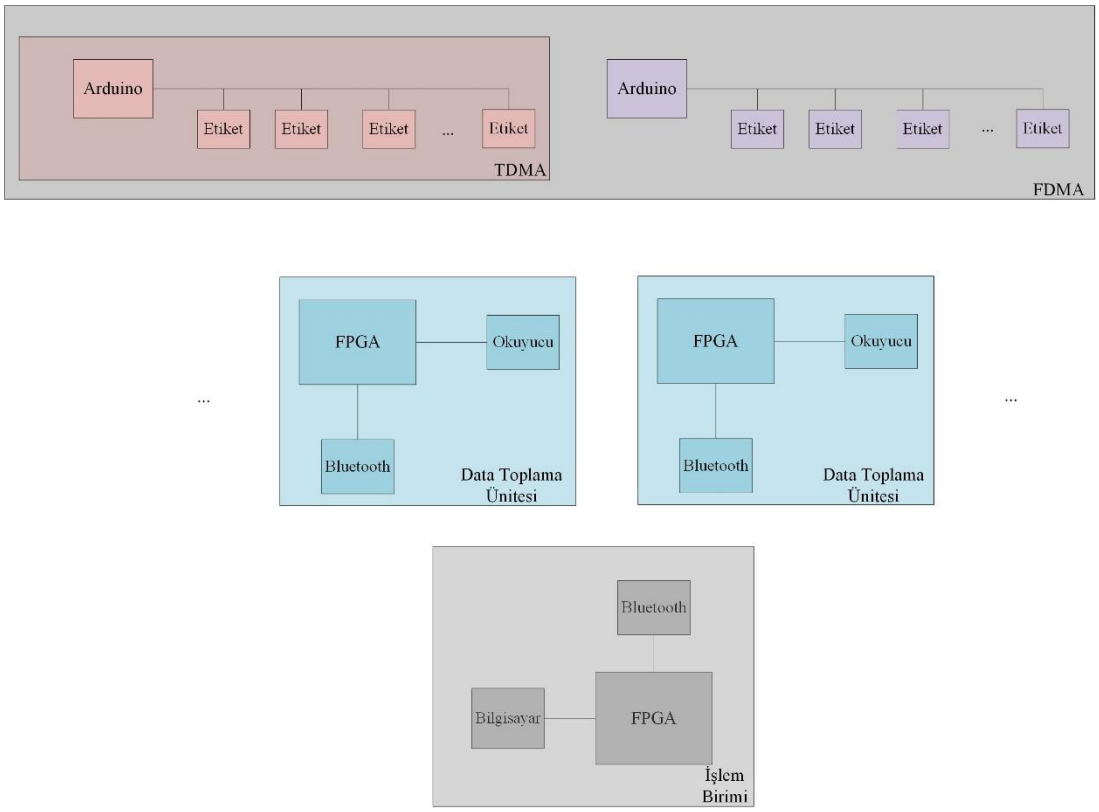
Şekil 1.1: RFID Etiket ve Okuyucuların İç Mekanda Dizilimi

sayısı sebebiyle işlem biriminin bilgisayar olması çeşitli yetersizliklere sebep olacaktır. Bu sebeple işlem biriminin ayrı bir donanım olarak tasarlanıp bilgisayarın sadece izleme amaçlı kullanılması daha hızlı ve güvenilir bir sistem sunacaktır. İşlem birimi için FPGA kullanılacak bu konum belirleme sisteminin tasarımı Şekil 1.2'deki parçalara ayrılmıştır.



Şekil 1.2:Projenin Parçaları

Ayrıca konum belirleme sistemi, Arduinolar tarafından sürülen etiketler, data toplama ünitesi ve işlem biriminden oluşmaktadır. Burada etiketlerden okunan datalar data toplama ünitesindeki FPGA kontrolünde Bluetooth ile işlem birimine gönderilmektedir. İşlem birimi data toplama ünitesinden gelen bu dataları mesafeye dönüştürülüp konum belirleme algoritmaları [18,19] ile okuyucunun etiketlere göre bulunduğu konum belirlenmekte ve bilgisayar tarafında bir kullanıcı arayüzü ile izlenmektedir. Sistemin genel görünümü Şekil 1.3'deki gibidir.



Şekil 1.3: Tüm Sistemin Çalışma Diyagramı

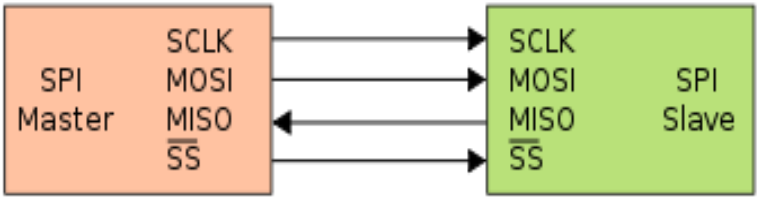
Bu bitirme projesi, data toplama ünitesindeki FPGA ile okuyucu arasındaki donanımsal haberleşmeden ve etiketten gelen datanın toplama ünitesinde okunması üzerine yoğunlaşacaktır.

2 ÖNBİLGİLER

2.1 Seri Çevresel Arayüz Haberleşme Protokolü

Seri Çevresel Arayüz (Serial Peripheral Interface-SPI); mikroişlemci veya mikrokontrolörler ile çeşitli yan birimler arasında iletişim kurmayı sağlayan bir haberleşme protokolü olup, yaygın olarak kullanılmaktadır [5]. Bu yan birimlere örnek olarak dijital sensörler, RFID okuyucular, flash hafıza, elektronik olarak silinebilir programlanabilir salt oku bellek (Electronically Eraseable Programmable Read Only Memory – EEPROM), dijital analog çevirici (Digital Analog Converter - DAC), analog dijital çevirici (Analog Digital Converter - ADC) ve daha fazlası gösterilebilir. SPI birincil (master) ile ikincil (slave) cihaz arasında iletişim kurulmasını sağlayan, senkron bir iletişim protokolüdür. Birincil ve ikincil cihaz arasında data alışverişine olanak sağlar. Şekil 2.1'den hareketle bu haberleşme protokolü dört pine sahiptir. Bunlar saat işareti (Serial Clock – SCK), birincil girişi ikincil çıkışı (Master In Slave Out - MISO), birincil çıkışı ikincil girişi (Master Out Slave In - MOSI) ve slave seçici (Slave Select - SS)'dir [6].

Şekil 2.2'de görüldüğü üzere, SPI haberleşme protokolünün pinlerinden ikisi MISO ve MOSI'dir. Bu pinler aracılığı ile birincil ve ikincil cihazlar arasında data bağlantısı kurulmuş olur. Saat işaretinin kontrolünde data alışverişi aynı anda ve çift yönlü olarak gerçekleşir. Bir başka deyişle birincil cihaz karşı tarafa data gönderirken aynı anda karşı taraftan gelen dataları okur. Bu şekilde diğer haberleşme protokollerinden farklı olarak çift yönlü data alışverişi gerçekleşmiş olur.



Şekil 2.1: SPI Birincil ve İkincil Cihazları ve Bağlantı Pinleri [7]

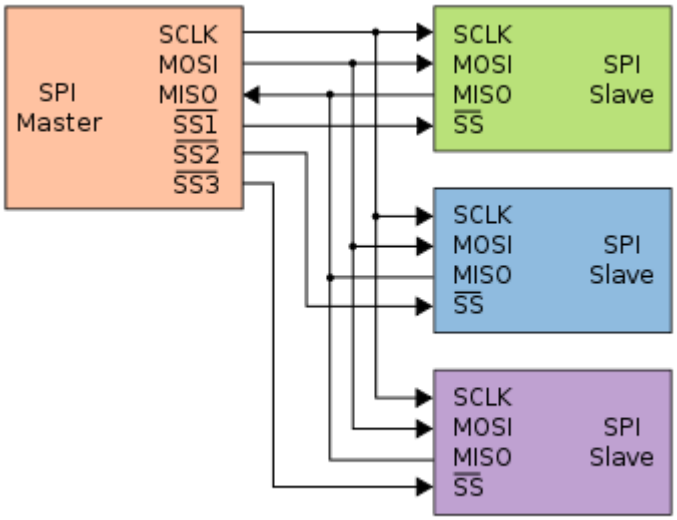
SPI haberleşme protokolünün diğer bir pini ise SCK pinidir. SPI haberleşme protokolü diğer asenkron seri haberleşme protokollerinden (RS-232, RS-485, CAN vb.) farklı olarak bir saat işareti ile senkronize edilir ve kontrol edilir. Bu saat işareti birincil cihaz tarafından sağlanır, dolayısıyla SPI haberleşme protokolünün ayarları sadece birincil cihaz tarafında yapılabilir. SPI saat işaretinin frekansı da birincil cihaz tarafında belirlenir. Bu SPI saat işareti, genellikle işlem biriminin mevcut mikroişlemci veya mikrokontrolörün belirli oranlarda bölünmesi ile elde edilir. SPI saat işareti okuma ve yazma işlemlerinde datanın ne zaman yazılıp okunacağını kontrol eder. Bu yazılma ve okunma işi saat işaretinin yükselen veya düşen kenarında yapılabilir. Bu noktada SPI haberleşmesinin modlarından bahsetmek gereklidir. SPI haberleşmesinin saatinin hangi kenarında transfer yapılacağını bu SPI modları belirler. Bu modlar saat polaritesi (Clock Polarity - CPOL) ve saat fazı (Clock Phase - CPHA) tarafından belirlenir. CPOL saatin sıfır veya bir aktif olarak çalışmasını belirlerken CPHA bu işaretin birinci kenarı veya ikinci kenarında mı data alışverişi yapılacağını belirler. Bu iki özelliği birlikte düşünürsek Tablo 2-1'deki durumlar ortaya çıkar ki bu da SPI modları olarak adlandırılır. Bu modlarda birincil cihaz ayarlanarak saat işaretinin istenilen kenarlarında ve aktifliğinde data alışverişi yapılabilir.

Tablo 2-1:SPI Modları

SPI Modu	Saat Polaritesi (CPOL)	Saat Fazı (CPHA)
0	0	0
1	0	1
2	1	0
3	1	1

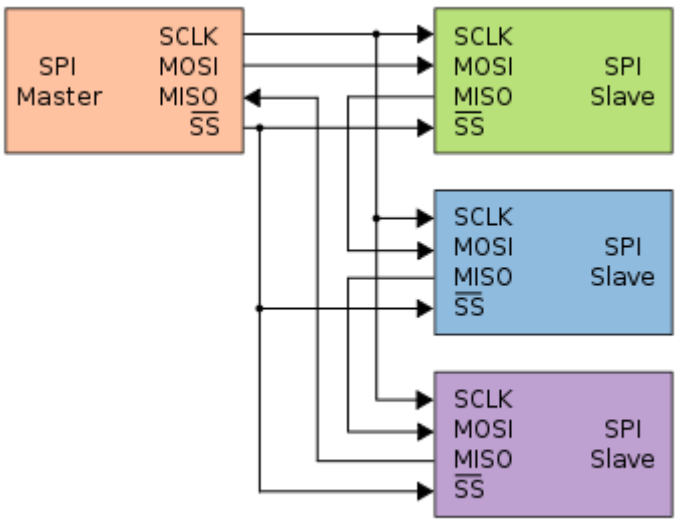
SPI protokolünün son pini ise SS pinidir. Bu pin Şekil 2.2'deki gibi master cihaza birden fazla slave bağlanması durumunda birincil cihazın hangi ikincil cihaz ile haberleşileceğine karar verir, diğer bir deyişle ikincil cihazlar arasından bir tanesini seçer. SS pini sıfır aktif olarak çalışır. Bu haberleşme protokolünde aynı anda birden

fazla ikincil cihaz ile haberleşmek mümkün olmadığından hangi ikicil cihaz seçilmek isteniyorsa onun SS pini sıfır, diğerlerinin ise bir yapılması gereklidir. Seçili olmayan cihazlar serbest olarak kalırlar ve diğer bir birincil cihazla haberleşebilirler.



Şekil 2.2: SPI Biricil Cihaza Birden FazlaİkincilBağlanması [7]

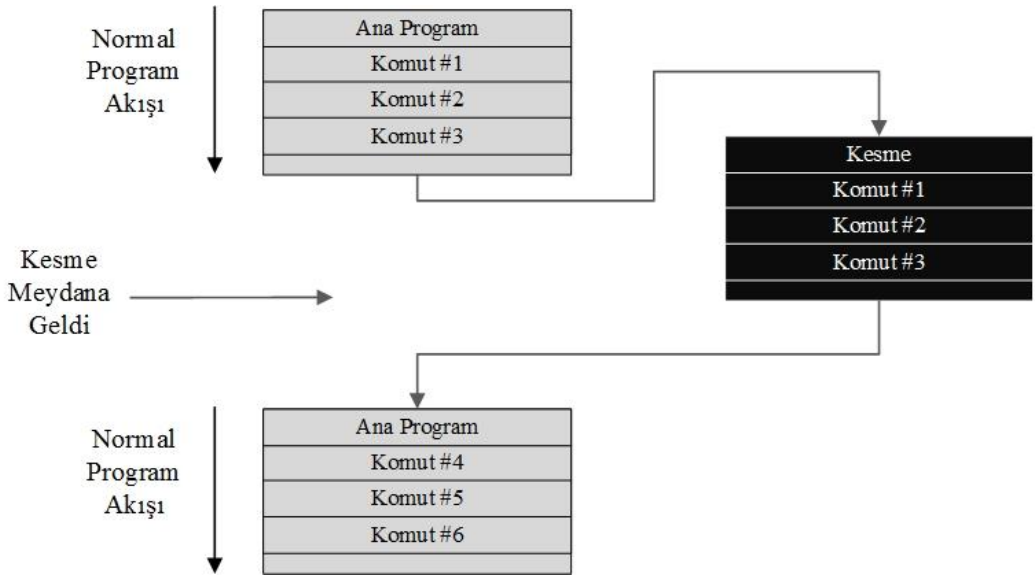
Ayrıca tek bir SS pini ile slave cihazların kaskat bağlanması da mümkündür. İkincil cihazın MISO pini bir sonraki slave cihazın MOSI pinine bağlanarak kaskat bağlantı sağlanmış olur. Bu şekilde datalar birincil cihazdan ilk ikincil cihaza, ilk ikincil cihazdan son ikicil cihaza doğru geçerek birincil cihaza ulaşır (Şekil 2.3). Bu esnada bütün ikincil cihazlar aynı anda seçilir. Bu sebeple bütün ikincil cihazlar aynı anda seçildiğinden diğer bir birincil cihaz tarafından seçilemezler.



Şekil 2.3: İkincil Cihazların Kaskat Bağlanarak Birincil Cihaza Bağlantı Yapılması [7]

2.2 Kesme

Kesme, mikrokontrolör veya mikroişlemci kullanılan sistemlerde meydana gelen istenmeyen durumların üstesinden gelmek amacıyla, sistem zamanlamasında veya sistemin açılıp kapanmasında sıkça kullanılmaktadır [8]. Ayrıca bir data baytının eternet veya UART üzerinden iletilmesinde de kesme önemli rol oynar. Bir sistemde kesme meydana geldiğinde program kendi çalışmasını askıya alarak hafızada bir yere kaydeder, kesme fonksiyonu tetiklenir ve kesme fonksiyonu tamamlandıktan sonra kesme servis rutini (Interrupt Service Routine - ISR) tarafından program kaldığı yerden tekrar başlatılır [9]. Şekil 2.4'te kesme rutininin işleyişi gösterilmiştir. Yazılımsal ve donanımsal olmak üzere iki çeşit kesme vardır.



Şekil 2.4: Kesmenin İşleyişi

Yazılımsal kesme, mikroişlemci veya mikrokontrolör üzerinde çalışan programda herhangi bir istisna meydana gelmesi durumunda mikroişlemci veya mikrokontrolörün kendisi tarafından tetiklenir. Örnek olarak bir artırma yapılırken tanımlanan değişkenin en yüksek sınırları aşıldığında veya sıfıra bölme işlemi gerçekleştirilmeye çalışıldığında bilgisayar işlemcisi tarafından verilen hatalar gösterilebilir. Ayrıca bilgisayarlar disk kontrolörlerinden data okurken veya yazarken yine yazılımsal kesme kullanılmaktadır.

Donanımsal kesme ise mikrokontrolörün veya mikroişlemcilerin bağlı oldukları donanımlardan gelen işaret doğrusunda tetiklenir. Burada dış donanımlardan gelen

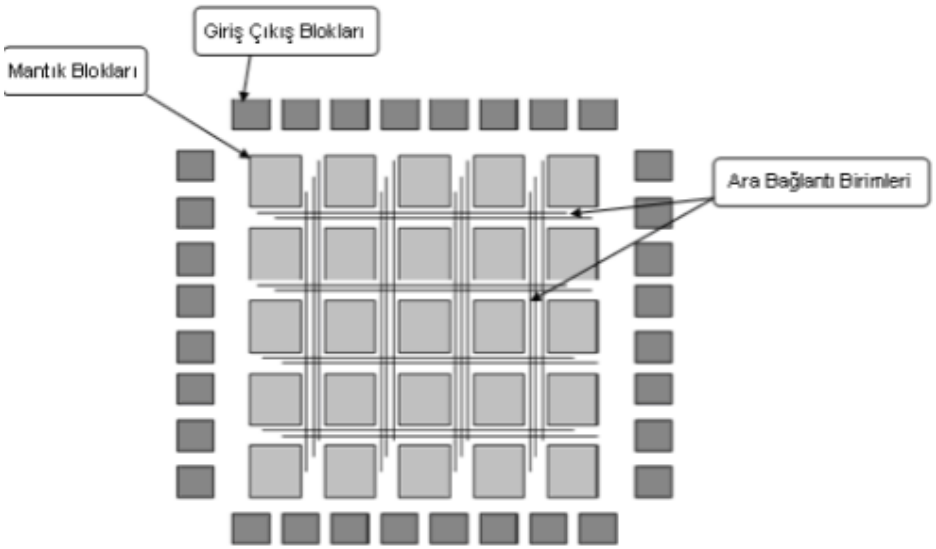
kesme, bir elektriksel sinyaldir. Gelen sinyal mikrokontrolörün veya mikroişlemcinin kesme pininden alınarak kesme tetiklenir ve uygun kesme rutini tekrarlanır. Örnek olarak dijital bir sensörün hata vermesi, bir kablosuz iletişim sağlayan sistemin veri alındığında veya gönderildiğinde işlemin tamamlandığını bildiren bir kesme göndermesi verilebilir. Bu kesme sinyalleri asenkron sinyallerdir ve herhangi bir saat işareti ile kontrol edilmez.

Kesme dijital işaretin kenarlarında veya seviyelerinde tetiklenecek şekilde ayarlanabilir. Bu durumda sırasıyla kenar veya seviye tetiklemeli kesmeler elde edilir. Bazı durumlarda hem kenar hem de seviye tetiklemeli kesmeler kullanılır bu da hibrit kesme olarak adlandırılır.

2.3 Sahada Programlanabilir Kapı Dizisi

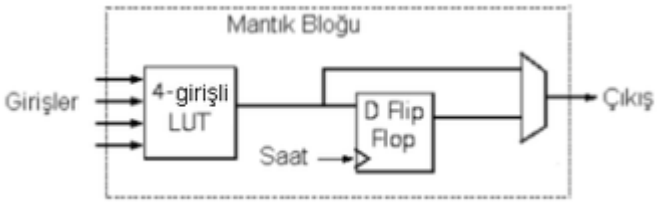
Sahada programlanabilir kapı dizileri (Field Programmable Gate Array-FPGA), programlanabilir yarı iletken devrelerdir [10]. ASIC tasarımı kolaylaştırmak ve maliyeti düşürmek amacıyla geliştirilmiş ve günümüze kadar da geliştirilmesi devam etmiştir. Günümüzde FPGA'ler birçok alanda yaygın olarak kullanılmaktadırlar.

FPGA entegre devreleri Şekil 2.5'teki gibi programlanabilir mantık blokları (Configurable Logic Block - CLB) ve bu blokları birbirine bağlayan, değiştirilebilir ara bağlantılardan oluşur.



Şekil 2.5: FPGA İç Yapısı

Burada CLB yapıları Şekil 2.6'da görüldüğü üzere eski nesil FPGA'ler için dört girişli bir doğruluk tablosu (Lookup Table - LUT), bir D flip flop ve çoklayıcıdan oluşmaktadır. Yeni nesil FPGA'ler ise altı girişli LUT'lar içermektedir. FPGA devrelerinin programlanabilir olmaları bu LUT'lar ve değişebilen bağlantılar sayesinde olmaktadır. Herhangi bir lojik fonksiyonun doğruluk tablosu LUT'lar içine yüklenir ve bu LUT istenilen lojik fonksiyon gibi davranır [11] . Dört girişli bir LUT 16 (2^4) 'lık bir doğruluk tablosunu hafızada tutabilirken altı girişli LUT'lar 64 (2^6) 'lük bir doğruluk tablosunu hafızasında tutabilir.



Şekil 2.6: Bir CLB Yapısı

Donanım tanımlama dili (Hardware Description Language - HDL) olarak adlandırılan programlama dilleri ile FPGA üzerinde sayısal tasarım yapılır. Verilog ve VHDL bu programlama dillerine örnektir. Bu diller ile yazılan devreler bir sentez aracıyla sentezlenerek devrenin şematiği çıkarılır. Ayrıca tasarım yapan kullanıcı tasarımla ilgili kısıtları ve pin bağlantılarını sentezleyiciye kullanıcı kısıtı dosyasına (User Constraints File - UCF) yazacağı kısıtlarla anlatır. Sentezleyici ve serim yapacak yazılım bu kısıtları göz önünde bulundurarak tasarlanan sistemi sentezler ve FPGA üzerinde girilen kısıtlara uygun olarak serim işlemi gerçekleştirilmiş olur. Xilinx firmasının FPGA'leri için serim aşaması çevirme (Translate), planlama (Map) ve yerleştirme ve bağlama (Place and Route - PAR) olmak üzere üç adımda yapılır. Çevirme işlemi esnasında çizilen şematiği firmanın kendi kütüphanesindeki elemanlar ile gerçekler. Planlama işlemi bu elemamnların girilen kısıtlara göre yerleştirileceği yerleri belirler. Son olarak PAR işlemi ile tasarım, kısıtlara uygun bir şekilde FPGA üzerine serilir ve serilen bu tasarımın bağlantıları yine UCF dosyasında belirtilen kısıtlara göre yapılır. Bu üç adımın ardında oluşturulan bit uzantılı dosya yardımıyla FPGA programlanır ve tasarlanan sistem FPGA üzerinde çalıştırılır.

Ayrıca bu işlemlerin her birinin ardından FPGA'ya yükleme yapmaksızın gerçekleme yapmak mümkündür. Bu gerçeklemeler Davranışsal ve PAR sonrası gerçekleme olarak ikiye ayrılır. Davranışsal gerçekleme yazılan verilog kodundan çıkan devrenin

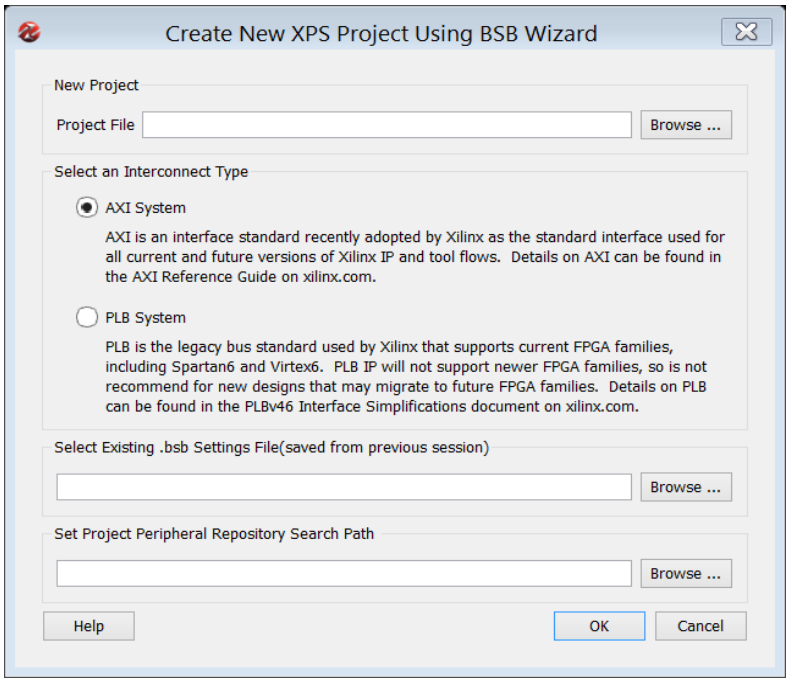
doğru bir şekilde çalışıp çalışmadığını belirlemek için kullanılır. Burada grafiklerde herhangi bir gecikme gözlemlenmez. Bu aşamada karşılaşılan hatalar giderilmeden serim işlemine geçilemez. PAR sonrası gerçekleme ise serim işlemi gerçekleştikten sonra FPGA üzerine yükleme yapmadan tasarımı test etme imkanı sunar. Bu gerçeklemede grafiklerde gecikmeler gözlemlenmektedir.

2.4 Microblaze İşlemci

Microblaze Xilinx firması tarafından üretilen FPGA'lere gömülebilen ve indirgenmiş komut takımıyla hesaplama (Reduced Instruction Set -RISC) mimarisine sahip, 32 bitlik bir ham işlemcidir [12]. Bu işlemci normal işlemcilere göre daha yavaş çalışmaktadır. Ancak FPGA üzerinde sayısal donanımla birlikte bir işlemci gömülmesine, yazılım donanım ortak tasarım yapılmasına olanak sağlamaktadır. Microblaze işlemci eski nesil FPGA'ler için PLB, yeni nesil için AXI bus yapısına sahiptir. Bu işlemcinin yanında akıllı özellik (Intellectual Property – IP) olarak adlandırılan ve daha önceden tasarlanmış yardımcı yan donanımlar bulunur. Bu donanımlar çeşitli haberleşme protokolleri, giriş-çıkış blokları, çeşitli hafıza elemanları, kayan nokta aritmetiği, kesme ve benzeri ile ilgili olacak şekilde ayrı ayrı tasarlanmıştır. Microblaze işlemci gömülü geliştirme Seti (Embedded Development Kit – EDK) arayüzünde ayarlanır ve yazılım geliştirme seti (Software Development Kit - SDK) arayüzünde üzerinde çalıştırılacak yazılım tasarlanır.

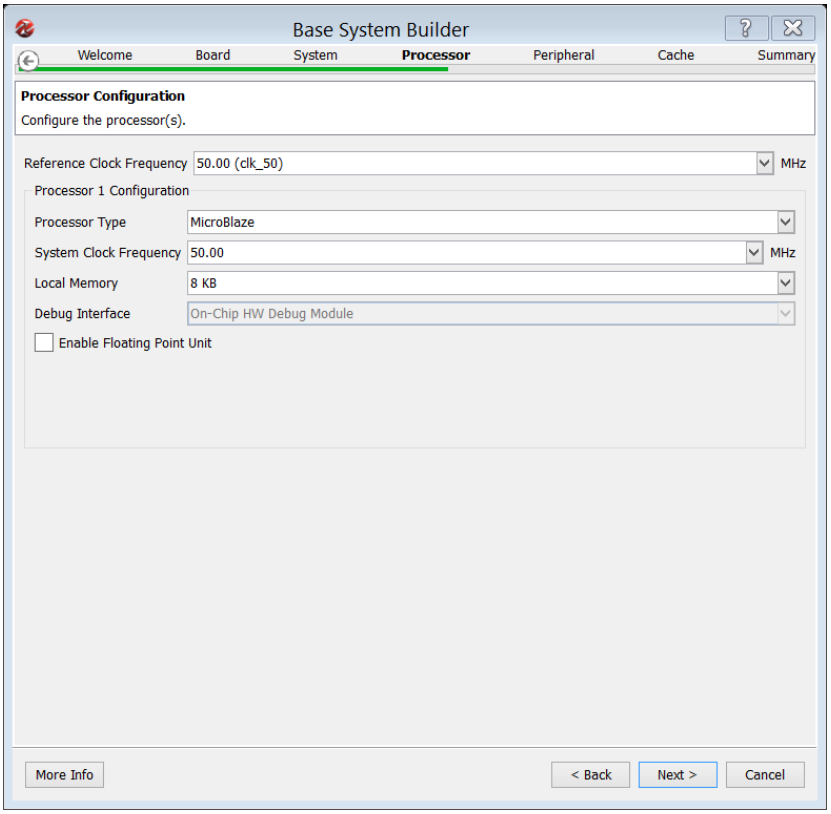
EDK arayüzü Microblaze işlemcinin ve yanına gömülecek hazır veya tasarımcı tarafından tasarlanmış IP'lerin[1] FPGA içine gömülecek hale getirildiği ortamdır. Bu hazır IP'ler SPI, RS-232,RS-485,CAN gibi haberleşme protokolleri; rastgele erişimli bellek (Random Acces Memory - RAM), salt oku bellek (Read Only Memory - ROM) gibi çeşitli hafıza elemanları; ışık yayan diyot (Light Emitted Diode – LED), buton, anahtar, PMOD girişler gibi giriş çıkış elemanları ve daha bir çoğu ile ilgili olabilir.

[1] Kullanıcı kendi tasarımını yapıp çalışmasını test ettiği donanımları Mikroblaze işlemci ile kullabilir. EDK arayüzünde ve dökümanlarında "Custom IP" tasarımı olarak bilinmektedir.



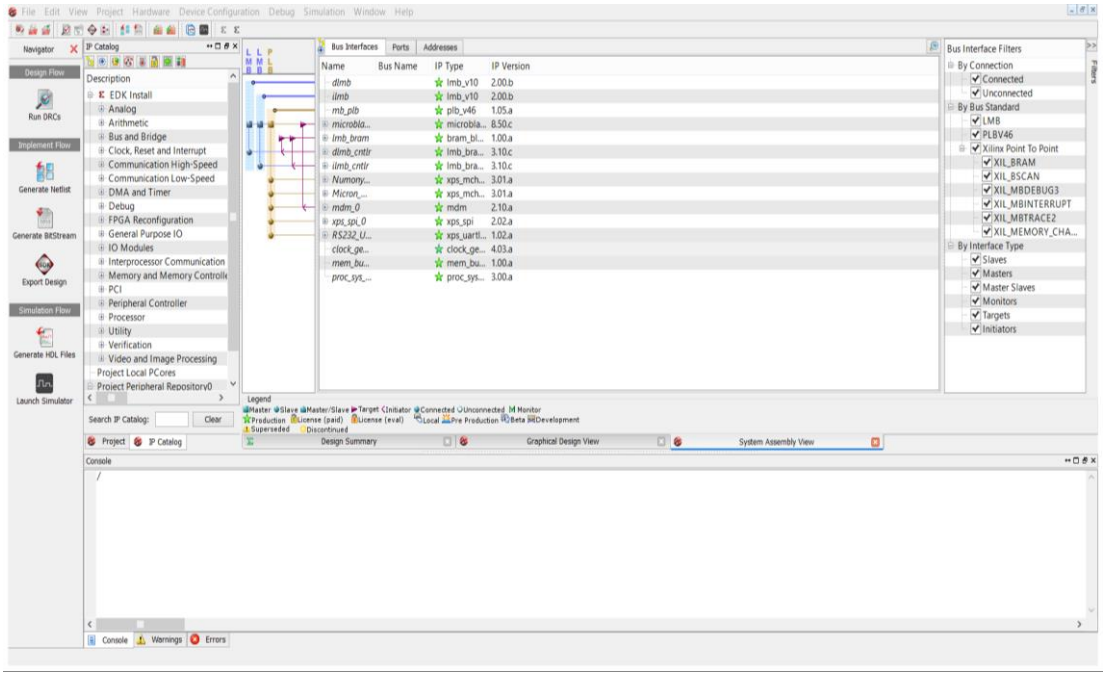
Şekil 2.7: EDK Proje Başlangıcı

EDK arayüzü Şekil 2.7'deki ekranla başlar. Burada yeni nesil FPGA'ler (Virtex-6,Virtex-7,Artix-7 vb.) AXI ve eski nesil FPGA'ler için (Spartan6, Spartan3E, vb.) PLB bus yapısı seçilerek Microblaze işlemcinin ayarlanmasına başlanır.



Şekil 2.8: Microblaze İşlemcinin Temel Ayarları

Daha sonra ilerleyen adımlarda Şekil 2.8'deki ekrandan işlemcinin saat frekansı, yerel hafızası (ara bellek) ve kayan nokta aritmetiğine dair seçenekler seçilir. Bir sonraki ekranda istenilen IP'ler işlemciye ek olarak tasarıma eklenir. Bu noktada bütün IP'ler listelenmez, sadece FPGA geliştirme kartı üzerinde yer alan bazı temel donanımlar seçilir. Bu noktada eklenen IP'ler için UCF dosyasına çıkış pinleri ile donanım bağlantısını yapan herhangi bir kısıt girmeye gerek yoktur, bu kısıtlar hazır olarak eklenir. Ayrıca bu sihirbazın devam eden adımında komut ve data arabellekleri eklenebilir. Daha sonra Şekil 2.9'daki proje ana ekranına geçilir. Bu ekranda işlemci ve işlemciye ait eklenmiş bütün IP'ler yönetilebilir. Bu ekranda IP, sihirbaz aşamasında eklenmeyen veya tasarımcının kendi tasarladığı IP'ler eklenebilir. "Ports" menüsü altında IP'ler için kullanılacak pinler FPGA içinden dış dünyaya açılır. Dış dünyaya açılmayan pinlere hiçbir şekilde bağlantı yapılamaz. Eklenen bu IP'ler için

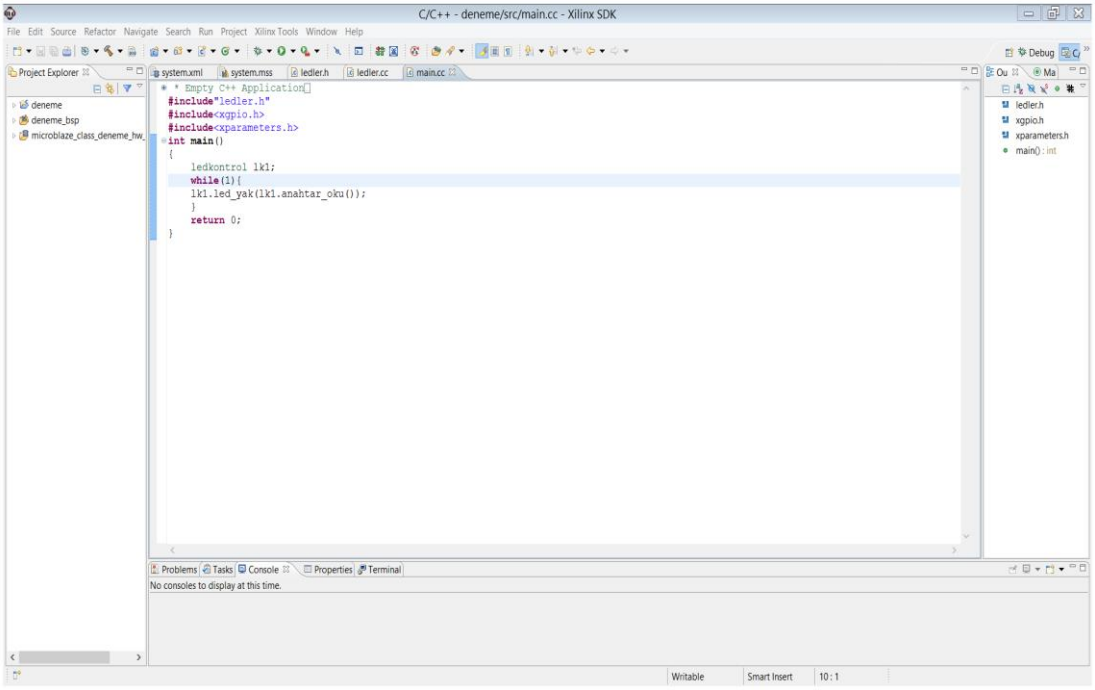


Şekil 2.9: Microblaze Ana Proje Ekranı ve IP Kataloğu

UCF dosyasına IP'nin dış dünyada nereye, hangi donanıma bağlanacağını gösteren kullanıcı kısıtı yazılmalıdır. "Adresses" menüsü içinde ise Mikroblaze işlemci için adresleme işlemi gerçekleştirilir. Adresleme işlemi yapılmadan tasarımın sentezlenmesi mümkün değildir.

Daha sonra Şekil 2.10'daki ekranda "Export Design" butonuna basılarak SDK ortamına geçilir. Burada mikroişlemciye yüklenecek olan yazılım yazılır. Bu yazılım için C veya C++ dilinde yazılabilir. Ayrıca C++ dili ile nesneye yönelik programlama mantığında yazılımlar Microblaze işlemci ile çalıştırılabilir ancak hafıza kullanımına

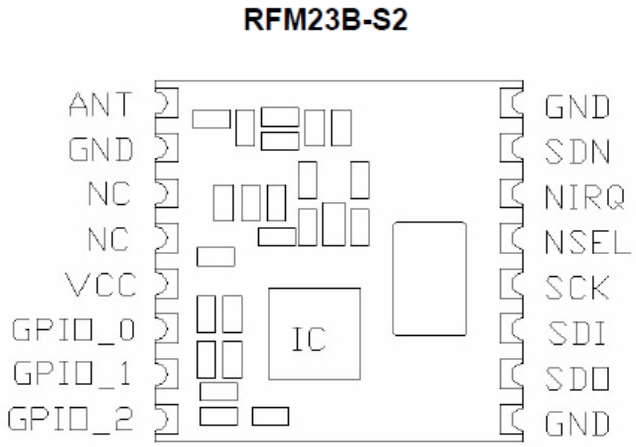
dikkat edilmelidir. SDK ortamında kullanıcak dile uygun bir şekilde proje eklenerek yazılım yazılabilecekŞekil 2.10'daki ortam açılır.



Şekil 2.10: Yazılım Geliştirilecek SDK Ortamı

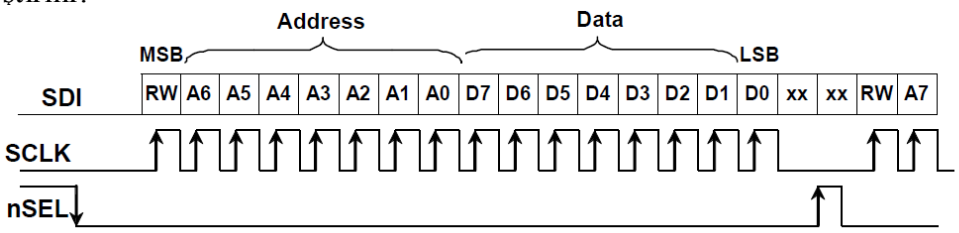
2.5 RFM23B Modülü

Şekil 2.11'deki RFM22 modülü hem okuyucu hem de etiket olarak kullanılabilen bir kablosuz iletişim cihazıdır [13]. Bu modül uzaktan kontrol, sensör ağları, endüstriyel kontrol ve ev otomasyonu gibi geniş alanda kullanılmaktadır. 433 MHz'lik çalışma frekansına ve -121 dBm hassaslığa sahiptir. 64 baytlık iletim ve alım (TX/RX) FIFOsu, otomatik paket kontrolü, uyandırma zamanlaması, düşük pil dedektörü gibi ek özelliklere sahiptir [13]. Bu modül genellikle bir mikrokontrolör veya mikroişlemci ile birlikte kullanılması için tasarlanmıştır. Ayrıca modül, kendi üzerinde bulunan ADC sayesinde herhangi bir ara eleman kullanılmadan mikroişlemci veya mikrokontrolöre bağlanabilir. Bu modül tamamen mikroişlemci üzerindeki yazılım ile kontrol edilebilecek bir yapıya sahiptir. Burada modülün mikroişlemci ile haberleşmesi ve mikroişlemci kontorolünde diğer bir modül ile haberleşmesi üzerinde durulacaktır. Modül ile ilgili diğer özellikler projenin modülün Arduino ile sürülmesi parçasında [14] detaylı bir şekilde anlatılmıştır.



Şekil 2.11: RFM22 RF Modülü Bağlanması [13]

RFM22 modülü mikroişlemci ile SPI haberleşme protokolü ile haberleşir. Şekil 2.11'deki bağlantı şemasında SDI ile belirtilen pin mikroişlemcideki MOSI'ye, SDO MISO'ya, NSEL SS'ye ve SCK ise SCK pinine bağlanarak SPI haberleşme bağlantısı yapılmış olur. Haberleşme protokolü modül üstünde yer alan entegredeki kayıtlara (register) yazmak veya bu kayıtlardan okumak için kullanılacaktır. Bu modülün SPI data alışverişi; 1 bitlik okuma-yazma seçme biti (R'/W), takip eden 7 bit adres alanı ve son 8 bit data olmak üzere toplam 16 bitlik dizi şeklinde gerçekleşir. Burada okuma-yazma seçme biti sıfır ise 7 bitlik adresten okuma, bir ise 7 bitlik adrese yazma işlemi gerçekleştirilir. İstenilen adrese yazma işlemi Şekil 2.12'de gösterildiği gibi SS pini sıfıra düştükten sonra[2], RW biti bir yapılarak takip eden 7 bitte istenilen adres gönderilir.Adresi takip eden 8 bitlik data istenilen adrese bu şekilde yazılmış olur. Yazma işlemi saat işaretinin yükselen kenarında bir bit transfer edilmek suretiyle gerçekleştirilir.

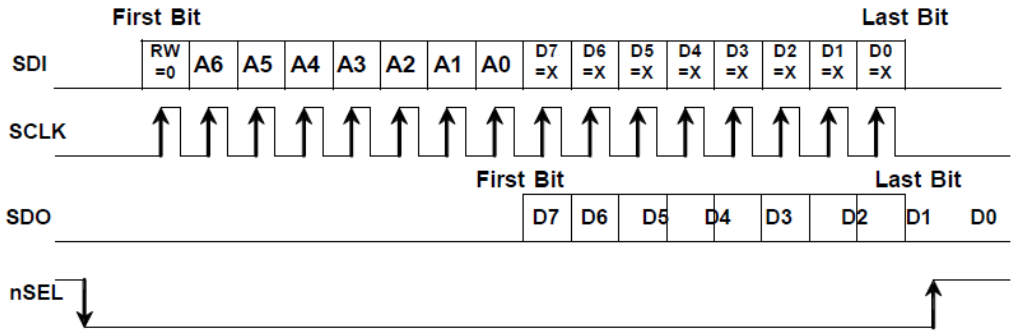


Şekil 2.12: RFM22 İstenilen Kayıt (Register) Adresine Yazma İşlemi [13]

Okuma işlemi ise R/W biti sıfır yapılarak takip eden 7 bitte adres gönderilir.Şekil 2.13'de görüldüğü üzere adresin devamında gelen 8 bitlik data sıfır yapılarak, o

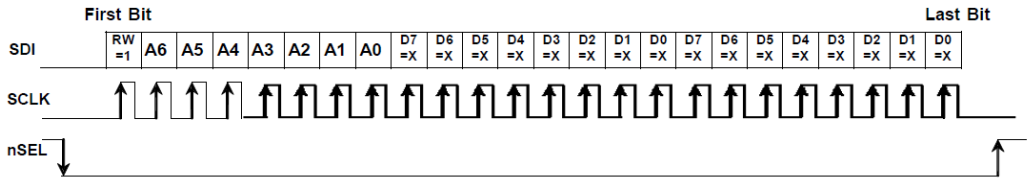
[2] SPI protokolünde, data alışverişinin başlaması için öncelikle SS pini sıfıra düşmeli, slave bu şekilde seçilmelidir. Slave seçilmeden herhangi bir data alışverişi mümkün değildir.

adresten okunan data ile doldurulur. Bu işlem yazma işleminde olduğu gibi saat işaretinin yükselen kenarında gerçekleştirilir.

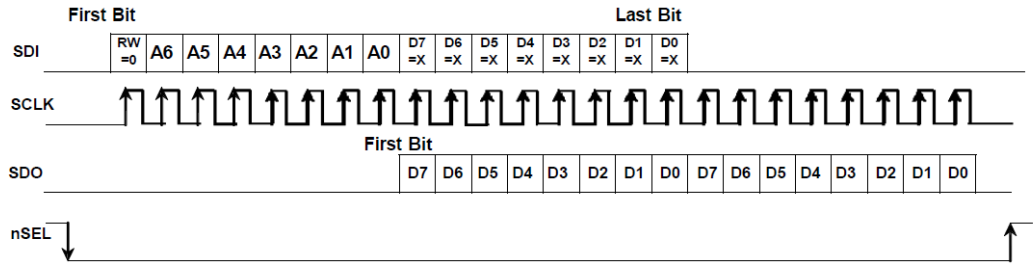


Şekil 2.13: RFM22 İstenilen Kayıt (Register) Adresinden Okuma İşlemi [13]

Bu işlemlerden sıralı kayıtlardan okuma yapılırken adres gönderme işleminin gereksiz yere tekrarlanmasını önlemek için yığın okuma (Burst Read) ve yığın yazma (Burst Write) modları bulunmaktadır. RW bitinden sonra adres bir kere gönderilir, SS pini sıfırda kaldığı sürece datanın sıralı kayda yazma veya sıralı kayıttan okuma işlemi Şekil 2.14 ve Şekil 2.15'teki gibi gerçekleştirilir.



Şekil 2.14: RFM23 İstenilen Adrese Yığın Yazma (Burst Write) Modu [13]



Şekil 2.15: RFM23 İstenilen Adrese Yığın Okuma (Burst Read) Modu [13]

Bu dört okuma-yazma modu için modüle bağlanacak olan SPI saat işareti SCK'nın frekansı 10 MHz değerini geçmeyecek şekilde istenilen değerde ayarlanabilir.

Ayrıca bu modül başka bir modül ile haberleşirken veya normal çalışma modunda iken kesmeler ile işlem birimini uyarmaktadır. Bu kesmeler veri paketleri karşı tarafa iletildiğinde, karşı taraftan veri geldiğinde veya herhangi bir hata oluşması durumunda işlem birimini NIRQ pinini sıfıra çekerek sadece uyarmaktadır. Bu sebeple Şekil 2.11'deki bağlantı şemasındaki NIRQ pini işlem biriminin kesme ile ilgili pinine

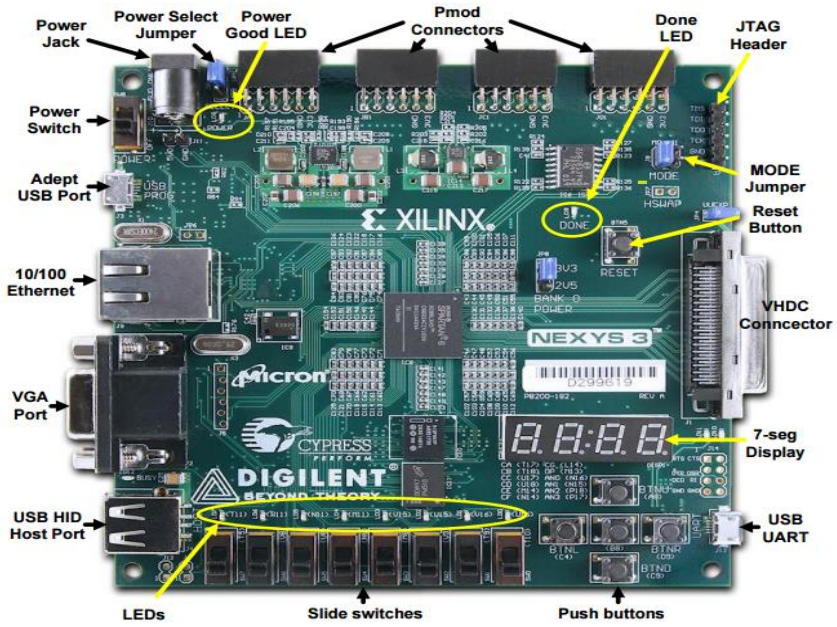
bağlanmış olmalıdır. Kesmenin neden meydana geldiğini belirleyabilmek için ise bu modülün kesme ile ilgili kayıtları kesme meydana geldiğinde okunması gereklidir. Bu durumda kesme rutini sonunda kesme ile ilgili kayıtlar sıfırlanarak ana program akışına dönülebilir. Bu sırada NIRQ pini modül tarafından tekrar bire çekilir, kesme durumu ortadan kalkar ve ana program kaldığı yerden devam eder.

2.6 Kullanılan Ölçüm Düzenekleri

Bu proje esnasında SPI Haberleşmesini gerçekleştirmek ve izlemek için Nexys3 FPGA geliştirme kartı ile Agilent 16802A Lojik Analizör kullanılmıştır.

2.6.1 Nexys 3 FPGA Geliştirme Kartı

Bu projenin işlem biriminin tasarlanması esnasında Digilent firması tarafından üretilmiş olan Şekil 2.16'daki Nexys3 FPGA geliştirme kartı kullanılmıştır. Bu kart üzerinde Xilinx Firmasının ürettiği "Spartan6 XC6SLX16 CSG324C" FPGA'i bulunmaktadır ve 100 MHz'lik bir kristal ile saat işareti mevcuttur.



Şekil 2.16: Nexys3 FPGA Geliştirme Kartı [15]

Yan hafıza donanımı olarak 16 MB'lık Numonyx PCM ve Mikron RAM kullanılmıştır. 8 adet LED ve anahtar, 5 adet buton, eternet, VGA, USB, USB-UART, VHDC bağlatnısı ve 4 grup PMOD pinleri (A,B,C ve D) mevcuttur. Proje ile ilgili olan PMOD konnektörlerinden, USB-UART bağlantısında bahsetmekte fayda vardır.

PMOD konnektörler kart dışından herhangi bir donanımın karta bağlanması için kullanılmaktadır. Bu konnektölerin her biri ikişer adet besleme (VDD) ve toprak (GND) pinine sahiptir. Bu pinlerden VDD 3.3V'luk besleme verirken 1A'e kadar akım çekilmesine olanak sağlamaktadır. Ayrıca herhangi bir kısa devreye karşı koruma önlemleri de alınmıştır. PMOD konnektörlerin bağlantı şeması Şekil B.1'de verilmiştir.

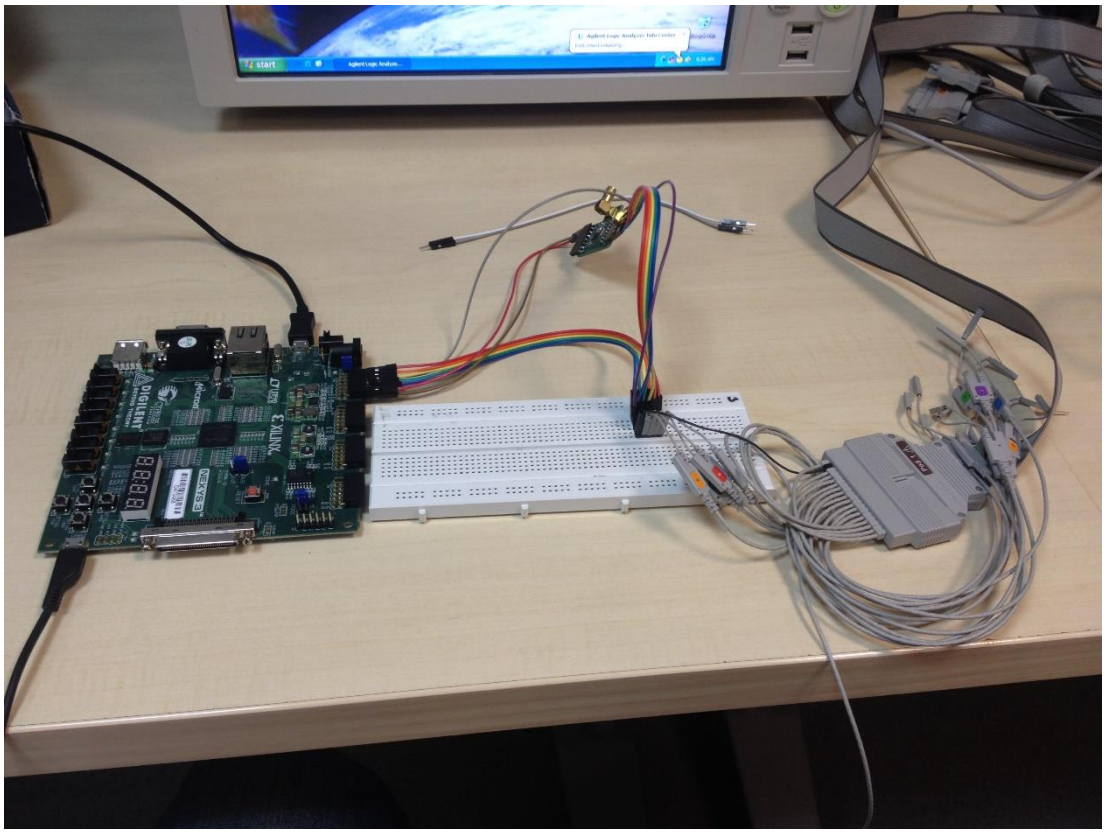
Proje ile ilgisi olan diğer konnektör ise USB-UART'tır. Bu bağlantı yardımıyla kart herhangi bir ara donanım olmaksızın bilgisayarın USB girişine bağlanabilir. Bu bağlantı yapıldığında bilgisayar kartı COM portu olarak görecektir. USB-UART yazılım üzerinde debug işlemi yapılırken çıktıları görmek için çok kullanışlı bir araç olarak kullanılmaktadır.

2.6.2 Agilent 16802A Lojik Analizör

Bu bitirme projesi süresince SPI haberleşmesini incelemek ve meydana gelen hataları giderip modül ile FPGA arasında sağlıklı bir haberleşme sağlamak için Agilent 16802A lojik analizörü (Şekil 2.17) kullanılmıştır. Bu lojik analizör 68 kanala sahiptir, 500 MB/s hıza kadar data akışını destekler. Ayrıca kendi içinde Windows XP işletim sistemi ve ölçüm programı bulunmaktadır. Bu program uygun bir şekilde konfigüre edilerek ölçüm işlemi gerçekleştirilir.

Şekil 2.17: Agilent 16802A Lojik Analizör [16]

Bu lojik analizör ile SPI haberleşmesini incelemek amacıyla Şekil 2.18'deki ölçüm düzeneği kurulmuştur. Modülü FPGA ile sürmek için yazılacak yazılım bu ölçüm düzeneğinin vereceği sonuçlar doğrultusunda hazırlanmıştır.



Şekil 2.18: SPI Haberleşmesini İncelemek Amacıyla Kurulan Ölçüm Düzeneği

3 MODÜLÜN FPGA İLE SÜRÜLMESİ

Projenin önemli kısımlarından biri de RFID okucunun FPGA üzerinde tasarlanacak işlem birimi ile haberleşebilmesidir. Çünkü hesaplama algoritması [17, 18] işlem birimindeki FPGA üzerinde donanımsal olarak gerçeklenmiş olup bu modüllerden gelen verileri işleyerek konum belirleme işini gerçekleştirecektir. Bu noktada konumu belirlenecek RFID okuyucu ile data toplama ünitesindeki FPGA'nın SPI protokolü ile haberleşmesi gerekmektedir. Bir önceki bölüm içinde okuyucu olarak kullanılacak RFM23 modülünün yazılım ile kullanılmasının daha kolay ve pratik olacağı belirtilmişti. Bu sebeple hazırlanan donanım ile bu okuyucuların haberleşme işlemini Microblaze işlemci yapacaktır. Microblaze işlemci üzerine yazılacak olan RF22 kütüphanesi kullanılarak okuyucunun kolay ve anlaşılır bir şekilde haberleşmesi ve kontrolü sağlanabilecektir. Modülün FPGA ile sürülmesi işi mikroblaze işlemci konfigürasyonu ve yan donanımları ile RF22 kütüphanesinin Mikroblaze işlemci için hazırlanması olmak üzere iki ayrı başlık altında incelenecektir.

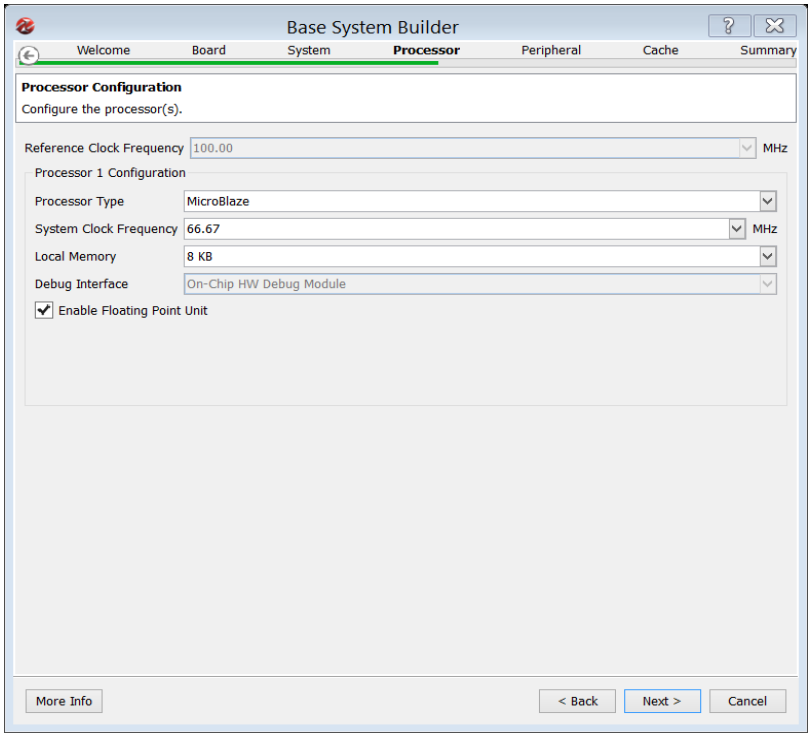
3.1 Microblaze İşlemci Konfigürasyonu ve Yan Donanımlar

Microblaze işlemcinin RFID okuyucu modülü ile haberleşebilmesi için mikroblaze işlemcinin uygun yan donanımlar ile birlikte ayarlanmış olması gereklidir. Bu noktada modül için kullanılacak iki adet yan donanım vardır: SPI ve Kesme Kontrolü IP'leri.

3.1.1 Microblaze Konfigürasyonu

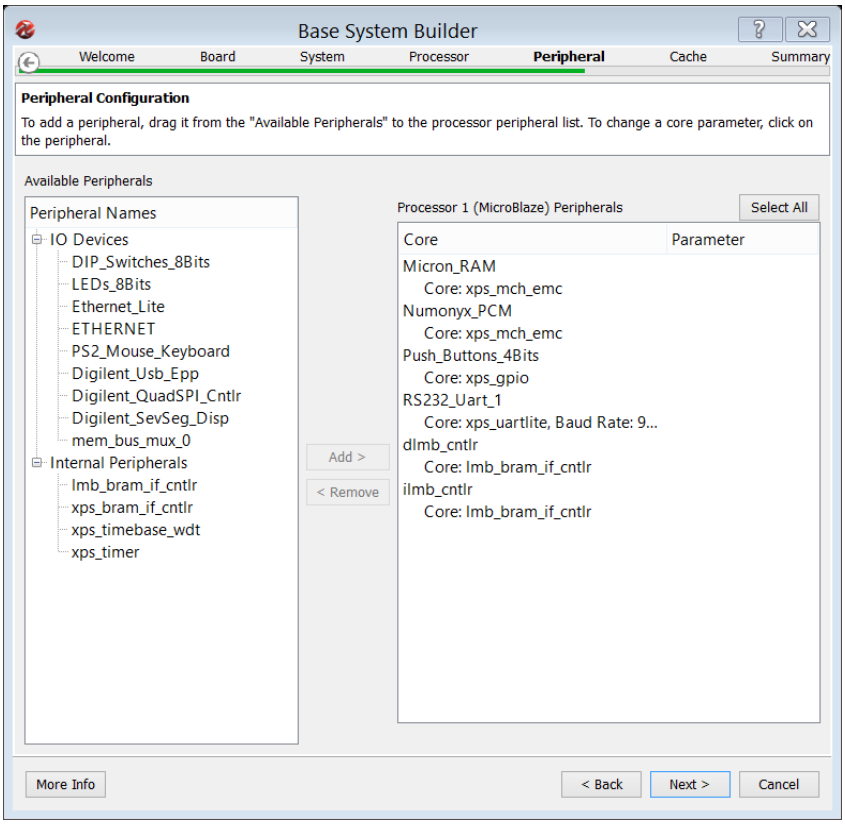
Modül ile tasarlanan sistemin haberleşmesinde önemli rol oynayacak Microblaze işlemci EDK ortamında daha önce belirtilen şekilde bir proje halinde oluşturuldu. Microblaze işlemci için Şekil 3.1'deki gibi kayan nokta ünitesi de aktif edilerek bir sonraki donanım ekleme ekranına geçildi. Kayan nokta aritmetiği modülün frekans değerleri ve merkez frekansı ayarlarında ondalıklı sayılar kullanılması sebebiyle açılmaktadır. Aksi takdirde modülün frekans ayarlarında ondalıklı sayılar ondalık kısmı atılarak işlem görür ki bu durum hatalı konfigürasyonlara ve haberleşme hatalarına neden olmaktadır. Arabellek değeri, FPGA içindeki kaynakları

kullanılabilecek sayısıal donanıma bırakmak amacıyla en düşük seçenek olan 8 kb olarak seçildi.

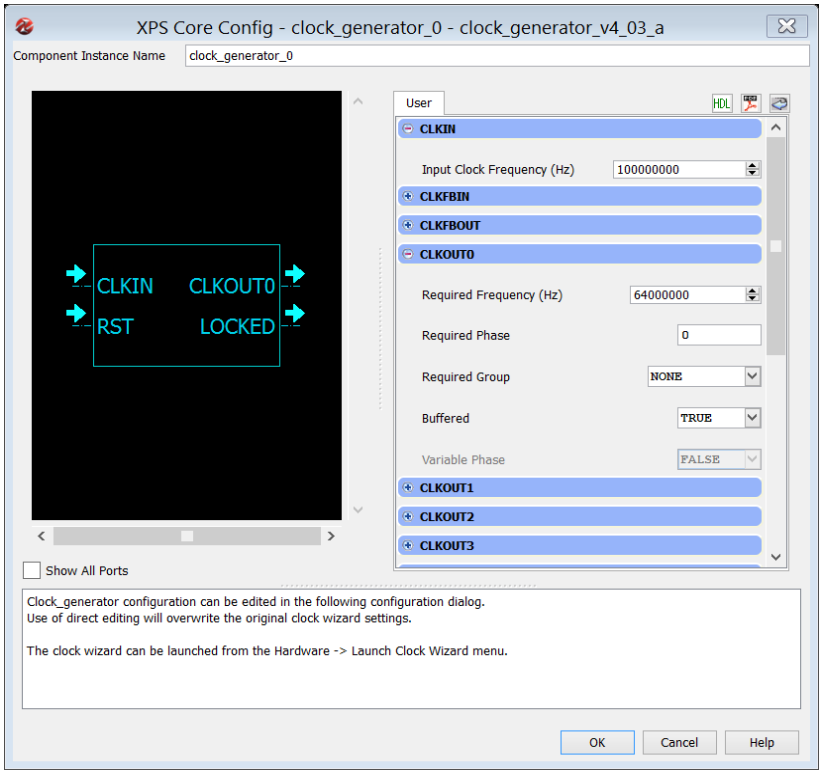


Şekil 3.1: Microblaze İşlemcinin RF22 Modül için Kayan Nokta Ünitesinin Açılması

Daha sonra kart ile ilişkili yan donanımlar sihirbazın bir sonraki adımında Şekil 3.2'deki gibi projeye eklenirler. Bu donanımlar Micron RAM, Numonyx PCM, 4 bitlik butonlar ve evrensel asenkron alıcı verici (Universal Asynchronus Receiver Transmitter - UART) donanımlarıdır. RAM veya PCM, RF22 kütüphanesinin nesneye dayalı programlama mantığı ile yazılmış olmasından dolayı hafızada meydana gelebilecek taşmaları önlemek içindir kullanılmıştır. Yazılacak program hafıza olarak blok RAM değil, kart üzerindeki bu RAM veya PCM'den birini kullanacaktır. UART bu kütüphanenin çalışmasının izlenmesinde kullanılacaktır. UART yardımıyla, oluşan bir hata bir programda ekrana bastırıldığı gibi terminale bastırılarak kısa sürede giderilmesi amaçlanmaktadır. Kart üstündeki buton IP'si ise kesme kontrolörü başlığı altında neden kullanıldığı detaylı bir şekilde açıklanmıştır. Bu aşamada 66.67 MHz olan işlemci saat frekansı, SPI IP'sinin bu değeri ikinin kuvvetleri oranında böldüğü düşünülerek Şekil 3.3'teki saat üretecinden 64 MHz olarak değiştirildi.



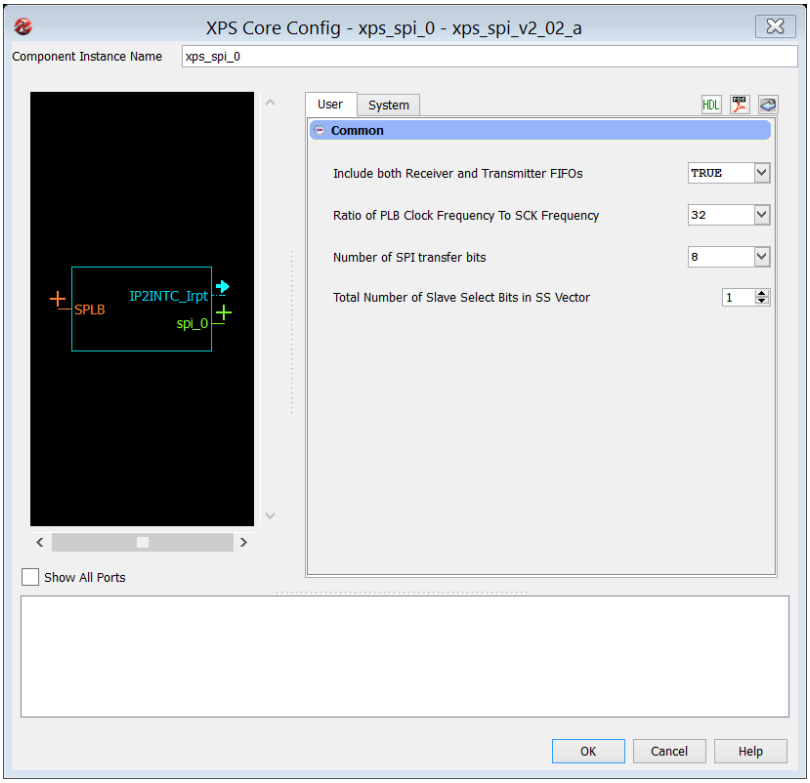
Şekil 3.2: Kart Üzerindeki Kullanılacak Donanımların Eklenmesi



Şekil 3.3: Saat Üreteci Üzerinden Mikroblaze İşlemcinin Saat Frkeansı Ayarları

3.1.2 Seri Çevresel Arayüzü Yan Donanımı

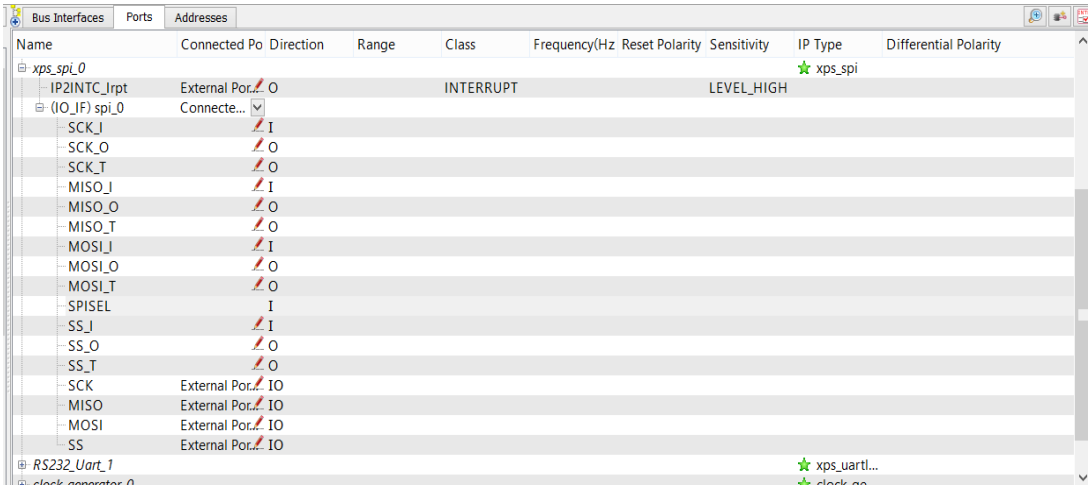
Mikroblaze işlemcinin yanına koyulabilecek IP'lerin çeşitli haberleşme protokolleri ile ilgili olduğundan bahsedilmişti. Modül ile SPI haberleşmesi bağlantısını kuracak SPI IP'si sisteme IP kataloğundan eklendi, Şekil 3.4'teki ekranda ayarları yapıldı. Bu



Şekil 3.4: SPI IP'si Ayarları

menüde daha önce 64 MHz olarak ayarlanan işlemci saat frekansını bölerek SPI haberleşmesinin saat frekansı olarak kullanacak olan bölücü değeri 32 olarak seçilmiştir. SPI saat frekansı ise 2 MHz olarak RFM23 modülünün belirlediği sınırlar içinde kalacak şekilde ayarlanmıştır. SPI transferleri modüle uygun olarak 8-bit olarak seçilmiştir. Son olarak projenin bu kısmında bir tane ikincil cihaz olacağından SS vektöründeki bit sayısı bir olarak bırakılmıştır. Bu vektör değeri içinde sadece bir tane bir olacak şekilde diğer bir değişle ikinin kuvvetleri olacak şekilde değer almaktadır. Birden fazla slave cihazla haberleşme gerçekletirileceği zaman SS vektörünün kaç bitlik olacağının bu kurala uygun bir şekilde yeniden belirlenerek değiştirilmesi gerekecektir.

IP eklendikten sonra bağlantılar "Ports" menüsü altında Şekil 3.5'teki gibi dışarıya açılan pin olarak ayarlandı. Burada SPI kesmeleri ve birincil olarak tasarlanan FPGA'da SPISEL[3] pini kullanılmayacağı için bağlantıları "No connection" seçeneği ile kesilmiştir. Daha sonra kart üzerindeki PMOD[4] çıkışlar ile bağlantı UCF dosyasına Şekil 3.6'daki satırlar eklenerek yapılmışıtır.



Şekil 3.5: SPI IP'sinin İşlemci Dışındaki Kısımlar ile Bağlantılarının Yapılması

UCF dosyasındaki bağlantılar, kullanılacak geliştirme kartının kılavuzuna [15] göre uygun şekilde yapılmalıdır. Aksi taktirde sentez ve yerleştirme hataları ile karşılaşılabilir, herhangi bir hata vermeden sentezi ve yerleştirmeyi geçen sistem yanlış bir donanıma bağlanmış olabilir.

```
Net xps_spi_0_MISO_pin LOC=T12  |  IOSTANDARD = LVCMOS33;#SDO JA1
Net xps_spi_0_MOSI_pin LOC=V12  |  IOSTANDARD = LVCMOS33;#SDI JA2
Net xps_spi_0_SCK_pin LOC=N10 |  IOSTANDARD = LVCMOS33;#SCK JA3
Net xps_spi_0_SS_pin LOC=P11  |  IOSTANDARD = LVCMOS33;#NSEL JA4
```

Şekil 3.6: Dışarıya Açılan Pin Olarak Yapılandırılan SPI IP Pinlerinin PMOD ile Bağlanması

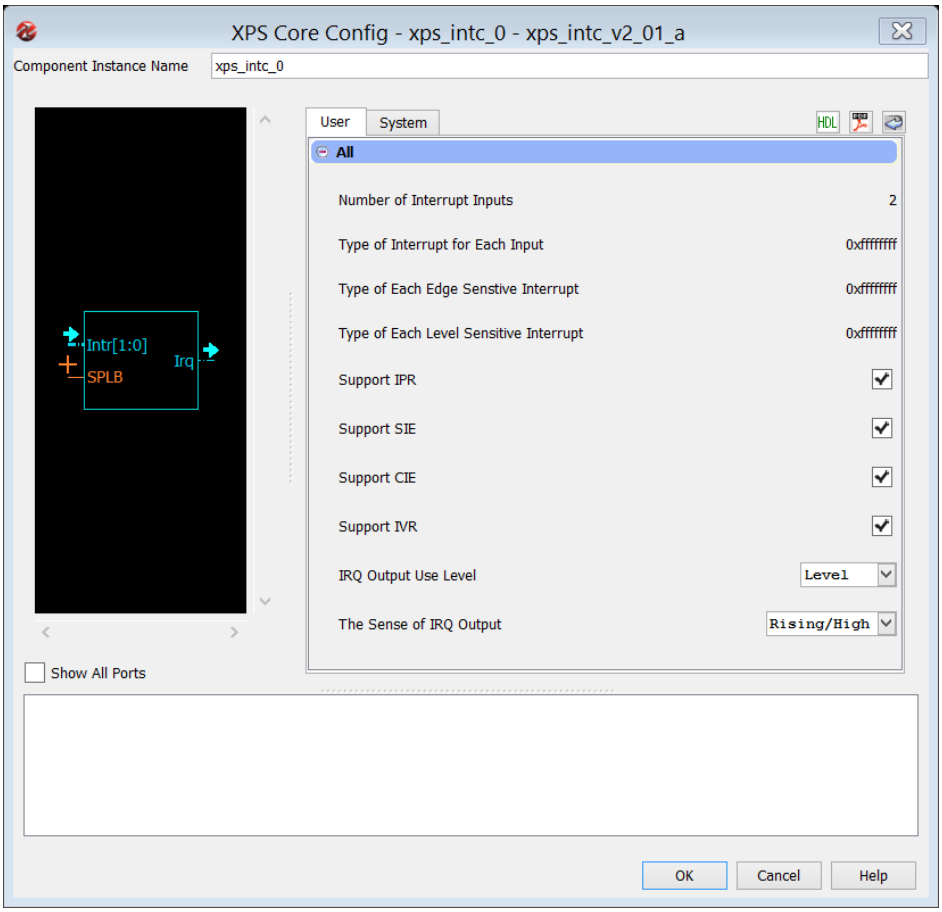
SPI IP'sinin eklenmesinin ardında kesme kontrolü IP'sinin eklenmesine geçilir.

[3] SPISEL pini sadece slave olarak tasarlanan Mikroblaze sisteminde kullanılabilir.

[4] Herhangi bir iç donanım kullanılmadan karta dışarıdan bağlantı yapılmasını sağlayan pinler PMOD olarak adlandırılmışlardır. Bununla ilgili detaylı bilgi ekler kısmında yer almaktadır.

3.1.3 Kesme Kontrolü Yan Donanımı

RFM23 modülü kesme ile çalışan bir modül olduğu için Microblaze ile RFID okuyucunun kesme pinlerinin bağlantısının yapılması gereklidir. Ancak Microblaze için eklenecek kesme kontrolü IP'si sadece kart üzerindeki ve FPGA içine konulabilen kesmeler için çalışmaktadır. Normal şartlar altında Mikroblaze grafik arayüzü üstünden işlemciye PMOD pinleri ile kart dışından herhangi bir kesme bağlamak mümkün değildir. Ancak 4-bitlik buton IP'si yardımıyla MHS[5] dosyasına el ile müdahale ederek PMOD üzerinden kesme bağlamak mümkündür [19, 20]. Buna göre sisteme ilk olarak bir kesme kontrolü IP'si IP kataloğundan seçilmiştir ve Şekil 3.7'deki gibi kesme kontrolörünün IRQ çıkışı bir seviyesine ayarlanmıştır. Farklı bir ayar seçilmesi istenilen çalışma sonuçlarını vermeyecektir. Bu şekilde kesme kontrolü IP'si projeye eklenir.



Şekil 3.7: Kesme Kontrolü IP'si Ayarları

[5] Microblaze işlemcide pinlerin, veri yollarının (bus) bağlantılarını ve eklenen IP'lerin ayarlarını içeren dosyadır. Grafik arayüzde yapılan her değişiklik bu dosyada da değişikliğe sebep olur. Sentezleyici bu dosyayı baz alarak sistemi sentezler.

Daha sonra MHS dosyası üzerinden kesme kontrol IP'sinin Irq çıkışı işlemcinin kesme girişine "mb_interrupt" ismiyle, Intr girişi ise butona "btn_interrupt" ismiyle Şekil 3.8'deki gibi bağlanmıştır. Bu bağlantı, MHS dosyası kaydedildikten sonra grafik arayüzü üstünde "Ports" menüsünde de gözlenebilir.

```
BEGIN microblaze
 PARAMETER INSTANCE = microblaze_0
 PARAMETER C_USE_BARREL = 1
 PARAMETER C_USE_FPU = 1
 PARAMETER C_DEBUG_ENABLED = 1
 PARAMETER HW_VER = 8.50.c
 BUS_INTERFACE DLMB = dlmb
 BUS_INTERFACE ILMB = ilmb
 BUS_INTERFACE DPLB = mb_plb
 BUS_INTERFACE IPLB = mb_plb
 BUS_INTERFACE DEBUG = microblaze_0_mdm_bus
 PORT MB_RESET = mb_reset
 PORT Interrupt = mb_interrupt
END

BEGIN xps_intc
 PARAMETER INSTANCE = xps_intc_0
 PARAMETER HW_VER = 2.01.a
 PARAMETER C_BASEADDR = 0x81800000
 PARAMETER C_HIGHADDR = 0x8180ffff
 BUS_INTERFACE SPLB = mb_plb
 PORT Irq = mb_interrupt
 PORT Intr = btn_interrupt
END
```

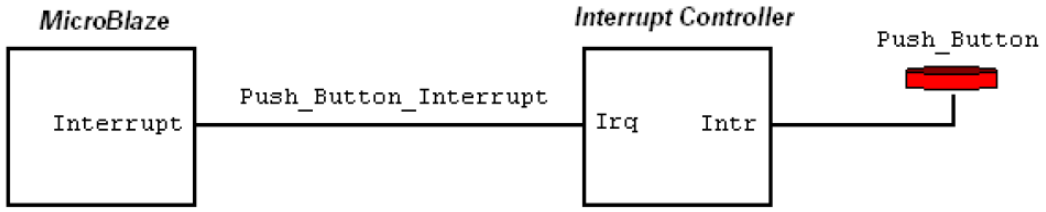
Şekil 3.8: İşlemci ile Kesme IP'sinin MHS Dosyası Üzerinden Bağlanışı

Kart üzerindeki buton donanımının IP'si Microblaze konfigürasyonu kısmında 1 bitlik buton olarak kesme destekleyecek şekilde projeye eklenmişti. Modülün kesme pini sıfır aktif olarak çalışmaktadır. Bu sebeple butonun Şekil 3.9'da olduğu gibi "Ports" menüsü içindeki "External Ports" başlığı altında "Class" özelliğini "INTERRUPT", sensivity özelliğini ise "EDGE_FALLING" seçerek RFID okuyucu modülüne uygun bir şekilde kesme işaretinin düşen kenarında tetiklenecek bir kesme gerçeklenmiş olur. Ayrıca buton IP'sinin kesme sinyali üzerinde herhangi bir işlevi olmadığından yerden kazanmak için buton IP'si bağlantıları korunarak silinebilir.

Name	Connected Po	Direction	Range	Class	Frequency(Hz	Reset Polarity	Sensitivity	IP Type	Diff
External Ports									
fpga_0_Push_Buttons_4Bits_GPIO_IO_I_pin	xps_intc_0::I...	I		INTERRU...			EDGE_FA...		
fpga_0_RS232_Uart_1_RX_pin	RS232_Uart...	I		NONE					
fpga_0_RS232_Uart_1_TX_pin	RS232_Uart...	O		NONE					
fpga_0_clk_1_sys_clk_pin	clock_gener...	I		CLK	100000000				
fpga_0_mem_bus_mux_0_DQ_pin	mem_bus_m...	IO	[0:15]	NONE					
fpga_0_mem_bus_mux_0_FLASH_ADDR_pin	mem_bus_m...	O	[5:7]	NONE					
fpga_0_mem_bus_mux_0_FLASH_CEN_O_pin	mem_bus_m...	O		NONE					
fpga_0_mem_bus_mux_0_FLASH_RPN_O_pin	mem_bus_m...	O		NONE					
fpga_0_mem_bus_mux_0_MEM_ADDR_pin	mem_bus_m...	O	[0:22]	NONE					
fpga_0_mem_bus_mux_0_MEM_OEN_pin	mem_bus_m...	O		NONE					
fpga_0_mem_bus_mux_0_MEM_WEN_pin	mem_bus_m...	O		NONE					
fpga_0_mem_bus_mux_0_MOSI_QUAD_SPI_pin	mem_bus_m...	IO		NONE					
fpga_0_mem_bus_mux_0_QUAD_SPI_C_O_pin	mem_bus_m...	O		NONE					
fpga_0_mem_bus_mux_0_QUAD_SPI_S_O_pin	mem_bus_m...	O		NONE					
fpga_0_mem_bus_mux_0_RAM_BEN_O_pin	mem_bus_m...	O	[0:1]	NONE					
fpga_0_mem_bus_mux_0_RAM_CEN_O_pin	mem_bus_m...	O		NONE					
fpga_0_rst_1_sys_rst_pin	clock_gener... proc_sys_res...	I		RST		1			
xps_spi_0_MISO_pin	xps_spi_0::[s...	IO		NONE					
xps_spi_0_MOSI_pin	xps_spi_0::[s...	IO		NONE					
xps_spi_0_SCK_pin	xps_spi_0::[s...	IO		NONE					

Şekil 3.9: "Ports" Menüsünden Butonun Kesme Olarak Ayarlanması

Bu bağlantı sağlandığında Şekil 3.10'daki şema oluşmuş durumdadır. Butona basılarak işlemciye kesme sinyali gönderilebilir. Yapılan testlerde kurulan bu sistemin başarılı bir şekilde çalıştığı gözlemlenmiştir. Bu noktada buton yerine UCF dosyasında PMOD pinlerinden birine bağlantı Şekil 3.11'deki satır ile yapılabilir.



Şekil 3.10: Kesme Sisteminin Bağlantılar Tamamlandıktan Sonraki Durumu [19]

Ancak bu işlem EDK ortamının yeni sürümlerinde ek olarak "CLOCK_DEDICATED_ROUTE=FALSE" kısıtını gerektirmektedir. Bu kısıt kullanılmazsa sentezleyici hata verecektir. Bu kısıt ile bu bağlantının saat işaretinden bağımsız bir şekilde FPGA üzerinde serilmesine olanak sağlanır.

```
Net fpga_0_Push_Buttons_4Bits_GPIO_IO_I_pin LOC=M10  |  IOSTANDARD = LVCMOS33
                                |CLOCK_DEDICATED_ROUTE=FALSE;#NIRQ JA7
```

Şekil 3.11: UCF Dosyasında Buton Yerine PMOD Pinlerine Bağlantı Yapılması

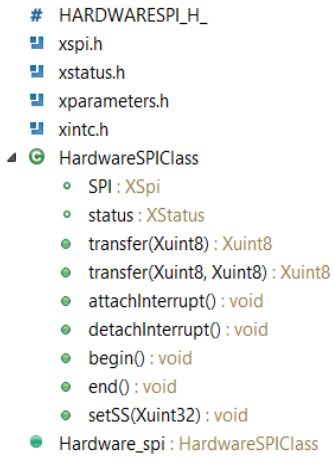
Son olarak tasarlanan bu sistem "Adresses" menüsü içinde adreslenerek sentez ve serim işlemi gerçekleştirilir ve tasarım SDK ortamına aktarılır.

3.2 RF22 Kütüphanesinin Mikroblaze İşlemcisi için Hazırlanması

Mikroblaze işlemci ve kullanılacak yan donanımlar yani IP'ler hazırlanıp, sentezlendikten sonra SDK ortamında Mikroblaze işlemcisinde çalışacak yazılım yazılmıştır. Bu noktada RFM22 modülü için Arduino ile çalışması için yazılmış kütüphane baz alınmıştır. Bu kütüphane C++ dili ile nesneye dayalı programlama tekniği ile yazılmıştır. Bu yazılım sadeleştirilerek sadece "Donanımsal SPI Sınıfı" ve "RF22 sınıfı" olmak üzere iki ana sınıf kullanılmıştır. Bu sınıflar SPI Protokolünü ve kesme sistemini uygun bir şekilde kullanarak diğer modüller (etiketler) ile başarılı şekilde haberleşmektedirler. Bu yazılım donanımsal SPI sınıfı ve RF22 sınıfı olarak iki ana başlıkta incelenecektir.

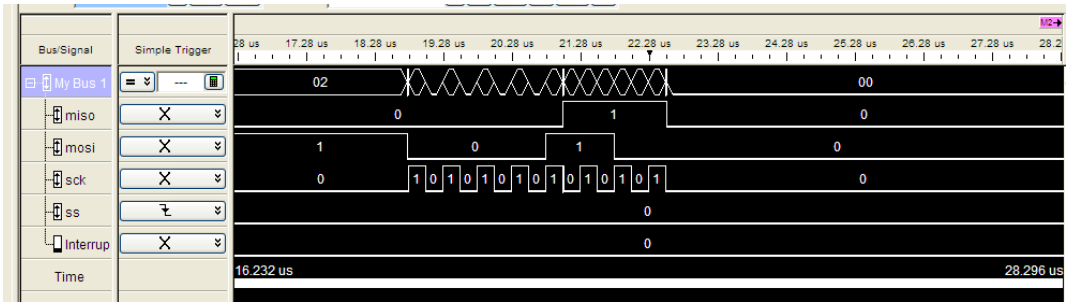
3.2.1 Donanımsal SPI Sınıfı

Donanımsal SPI sınıfı, SPI IP'si kullanılarak Microblaze işlemci ile RF modül arasında SPI ile haberleşmesini yazılım ortamında modelleyecektir. Bu sınıfın, sınıf diyagramı Şekil 3.12'de verilmiştir. Bu sınıf SPI IP donanımının bilgilerini tutan "XSpi" tipinde ve durum bilgilerini tutan "XStatus" tipinde bir değişkenden oluşmaktadır.



Şekil 3.12: Donanımsal SPI Sınıf Diyagramı

Bu sınıf içindeki "begin" ve "end" fonksiyonları eklenen SPI IP'sini başlatıp, sonladırmak için kullanılır. "begin" fonksiyonuna bakılırsa fonksiyon Microblaze işlemci ile ilgili kesmeleri açarak başlamaktadır. Burada kesme IP'sinin yazılımla ilişkilendirilmesi için öncelikle işlemcinin kesmeleri açılmalı daha sonra kesme IP'si ile yazılım ilişkilendirilmelidir. Daha sonra kesme IP'si başlatılabilir. Daha sonra sınıf değişkeni olarak tanımlanan "status" kullanılarak SPI IP'sinin düzgün bir şekilde yazılım ortamında kaydının yapılıp yapılmadığı UART ile terminal ekranına basılır. Daha sonra SPI ip'si birincil olarak ayarlanır. Bu noktada saat işaretinin ayarlarının da yapılması gerekir. RFM23 modülünün SPI protokolü ile Microblaze işlemciyle bağlantı kurabilmesi için ek olarak saat fazının bir olarak ayarlanması gereklidir. Bununla ilgili lojik analizör ile yapılan ölçüm sonuçlarına göre



Şekil 3.13: Saat Fazının 1 Olması Durumu

(Şekil 3.13) saat fazı bir yapılarak RFM23 modülünün haberleşme koşullarına uygun bir ortam hazırlanmıştır. Bu ayarlar hazır maskeler birbiriyle "veya" işlemine sokularak bir sayı oluşturulmuş ve bu sayı ayar olarak SPI IP'sine verilmiştir. Ayar işleminin başarılı bir şekilde gerçekleşip gerçekleşmediği "status" değişkenine dönen değere göre UART ile bilgisayar ekranındaki terminalden izlenmiştir. Son olarak SPI IP'si ikincil cihaz seçmeyecek şekilde başlatılır ve SPI kesmeleri kullanımayacağından kapatılır veya diğer bir deyişle "polled" moduna alınır [21]. Bu fonksiyon ile ilgili kod Şekil 3.14 ile verilmiştir. "end" fonksiyonu ise çalışan SPI IP'sini durdurur.

```
/// Initialize the SPI library
/// Initializes the SPI bus by setting SCK, MOSI, and SS to outputs, pulling SCK and MOSI low, and SS h
void begin()
{
    microblaze_enable_interrupts();
    XIntc_MasterEnable(XPAR_XPS_INTC_0_BASEADDR);
    XIntc_EnableIntr(XPAR_XPS_INTC_0_BASEADDR,0x000001);
    print("Interrupt init success.. \r\n");

    XStatus status=XST_SUCCESS;
    status=XSpi_Initialize(&SPI,XPAR_SPI_0_DEVICE_ID);
    switch(status){
        case XST_SUCCESS:
            print("SPI Init success... \r\n");
            break;
        case XST_DEVICE_IS_STARTED:

            print("SPI Device is started... \r\n");
            break;
        case XST_DEVICE_NOT_FOUND:
            print("SPI Device not found... \r\n");
            break;
    }
    //Be careful "clk phase 1" option to set the mosi changes to rising edge of spi clk
    //You can use loopback mode to see the data sent
    status=XSpi_SetOptions(&SPI,XSP_MASTER_OPTION|XSP_MANUAL_SSELECT_OPTION|XSP_CLK_PHASE_1_OPTION);
    switch(status){
        case XST_SUCCESS:
            print("SPI Options are successfully saved... \r\n");
            break;
        case XST_DEVICE_BUSY:
            print("SPI Device is busy... \r\n");
            break;
        case XST_SPI_SLAVE_ONLY:
            print("SPI Device is slave only... \r\n");
            break;
            }
    XSpi_SetSlaveSelect(&SPI,0);
    XSpi_Start(&SPI);
    XSpi_IntrGlobalDisable(&SPI);//In polled mode you should close SPI Interrupts
}
```

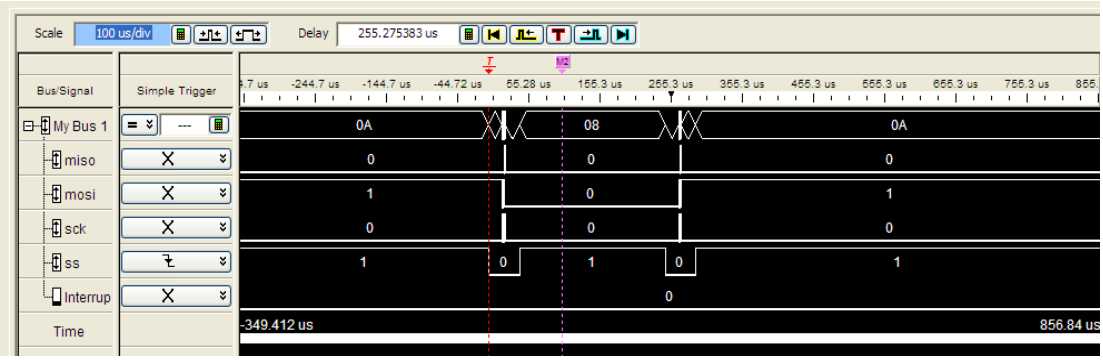
Şekil 3.14: Donanımsal SPI Sınıfı "begin" Fonksiyonu

"setSS" fonksiyonu (Şekil 3.15) ise hangi slave cihazın seçileceğini belirlemektedir. Bu fonksiyonun içine daha önce de belirtildiği gibi ikinin kuvvetleri olacak şekilde sayılar girilmelidir. Ancak sistem için tek bir ikincil cihaz olacağı için fonksiyon içine bir girilerek ikincil cihaz seçilir, sıfır girilerek serbest bırakılır.

```
void setSS(Xuint32 val){
    XSpi_SetSlaveSelect(&SPI,val);
  }
```

Şekil 3.15: "setSS" Fonksiyonu

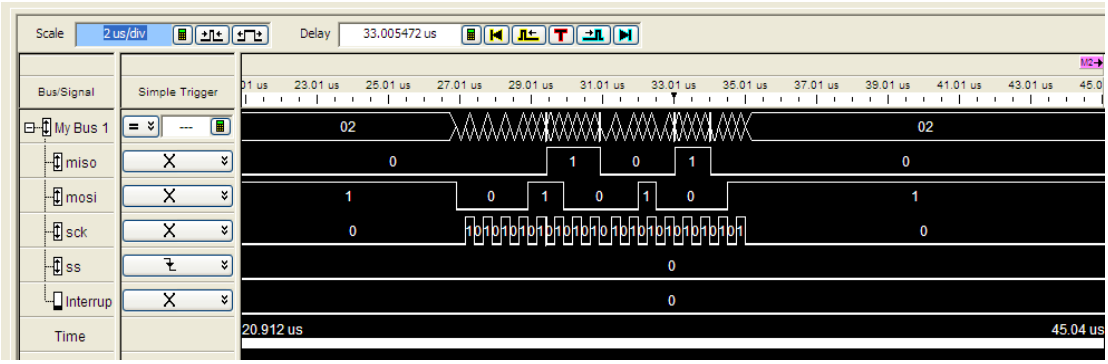
"Transfer" fonksiyonu ise çift yönlü data transferini sağlayacak olan foksiyondur ve ikincil cihazdan gelen datayı döndürür. Bu fonksiyon iki tane "overload[6]" versiyonuna sahiptir. Tek değişken alan ilk versiyon ("transfer(Xuint8)"), bu değişkenin tuttuğu değeri ikincil cihaza gönderirken, ikincil cihazdan gelen datayı döndürür. Bu versiyon adres ile datayı aynı anda gönderecek şekilde yazılmamıştır ve bu işlemi gerçekleştirmek için iki kere çağırılıp ilkinde adres, ikincisinde ise data gönderilir. Ancak SPI IP'si fonksiyonun her çağırılışında lojik analizör ile Şekil 3.16'daki gibi gözlemlendiği üzere SPI iletişimini keserek yeniden başlatmakta ve bu süre içinde yaklaşık 200 ns geçmektedir.



Şekil 3.16: Transfer Fonksiyonun Birinci Versiyonunun SPI Haberleşmesi

Ancak adres ve data bilgilerinin arada herhangi bir kesilme olmadan arka arkaya gönderilmesi gerekmektedir. Bu şekilde RF modülün kayıtlarına herhangi bir yazma

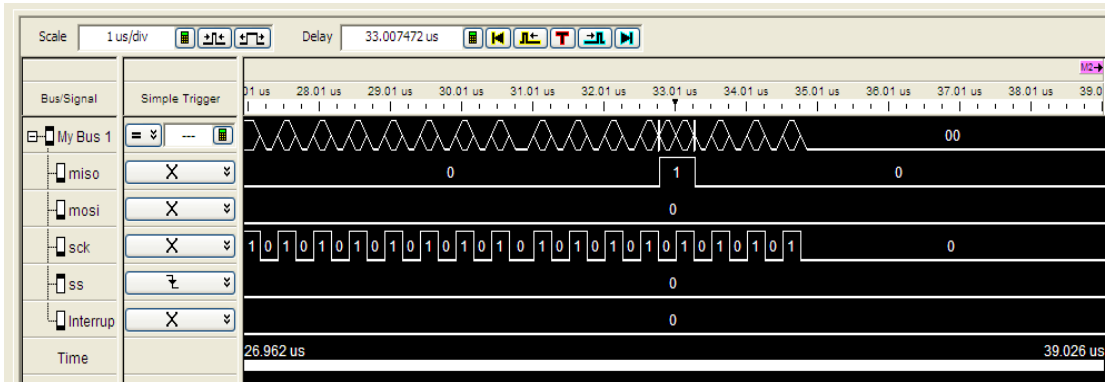
[6] Nesneye dayalı programlama mantığında bir fonksiyon içine farklı değişkenler almak ve aynı isimle kullanılmak şartıyla aynı isimle yazılabilir. Overload, basitçe bir fonksiyonun özelleşmiş yeni bir versiyonu olarak düşünülebilir.



Şekil 3.17: Transfer Fonksiyonun İkinci Versiyonunun SPI Haberleşmesi

ve kayıttan okuma işlemi yapılamamaktadır. Kayıttan okuma, data sıfır olarak girilip dönen değeri okuyarak yapılmaktadır. Bu soruna çözüm olarak transfer fonksiyonunun ikinci overload versiyonu yazılmıştır. Bu fonksiyon adres ve data bilgileri ile iletişimi, bunları arka arkaya ekleyip iki 8 bitlik toplamda 16 bitlik bir dizi olarak gerçekleştirmektedir. Bu şekilde SPI iletişimi Şekil 3.17'deki gibi kesilmeden veya arada herhangi bir boşluk olmadan gerçekleşir. Kayıttan okuma işlemi için ise yine data değeri sıfır olarak girilir ve fonksiyonun döndürdüğü değere bakılır. Data alışverişi için transfer fonksiyonunun bu ikinci overload versiyonu kullanılmıştır.

Bu ikinci fonksiyonun okuma işlemini test etmek için modülün kayıtlarının 0x00 adresindeki cihaz tipi değeri okunmaya çalışılmıştır. Bu adresten dönen değer hem yazılımda hem de Şekil 3.18'deki gibi lojik analizör üzerinde okunmuştur. Modülün kataloğuna göre okunan bu değer doğrudur ve okuma işlemi de başarılı bir şekilde gerçekleşmiştir.



Şekil 3.18: Lojik Analizör Yardımıyla Modülden Okunan Değerin Kontrol Edilmesi

3.2.2 RF22 Sınıfı

RFM22 modülü Arduino için hazırlanmış kütüphane ile sürülebilmektedir. Bu kütüphane bu projede de planlandığı üzere C++ dilinde nesneye dayalı programlama mantığında yazılmıştır. Bu kütüphane ile ilgili detaylı bilgi modülün Arduino kullanılarak sürülmesi işinde [14] mevcuttur. Bu kütüphane, yazılmış diğer kütüphanelere göre daha kullanışlı ve kolay anlaşılır bir şekilde yazılmıştır. Bu sebeple RF22 kütüphanesi, Mikroblaze işlemci üzerinde çalışacak yazılımda temel kütüphane olacaktır. Bu kütüphane RF22 sınıfında ve bu kütüphane içinde tanımlı kayıt adreslerinden oluşmaktadır. Ayrıca RF22 sınıfı SPI haberleşmesi için donanımsal SPI sınıfını kullanmaktadır. RF22 sınıfına ait sınıf diyagramı EK-A'da verilmiştir.

İlk olarak kütüphanenin Arduino'ya özgü kısımları Mikroblaze yazılımına özgü hale getirildi (Şekil 3.19). Bunun için uint8_t olan 8 bitlik değişken tipi Mikroblazedeki Xuint8 olarak değiştirildi. Aynı işlem 16 bitlik değişkenler için de Xuint16 olarak değiştirildi. Arduinoya özgü "progmem" ve kesme (atomik blok) ifadeleri kaldırıldı.

```
uint8_t RF22::spiRead(uint8_t reg)
{
    uint8_t val;

    ATOMIC_BLOCK_START;
    digitalWrite(_slaveSelectPin, LOW);
    _spi->transfer(reg & ~RF22_SPI_WRITE_MASK);
    val = _spi->transfer(0);
    digitalWrite(_slaveSelectPin, HIGH);
    ATOMIC_BLOCK_END;
    return val;
}
```

```
Xuint8 RF22::spiRead(Xuint8 reg)
{
    microblaze_disable_interrupts();
    Xuint8 val;
    _spi->setSS(_slaveSelectPin);
    val=_spi->transfer(reg & ~RF22_SPI_WRITE_MASK,0);
    _spi->setSS(0);
    microblaze_enable_interrupts();
    return val;
}
```

Şekil 3.19: Arduino'ya (sağda) ve Mikroblaze işlemciye (solda) ait kod parçaları

RF22 sınıfı, modülüne yazma ve modülünden okuma işlemleri için donanımsal SPI sınıfındaki transfer fonksiyonunu kullanır. Bu Transfer fonksiyonunun kullanıldığı fonksiyonlar "spiRead" (Şekil 3.19) ve "spiWrite" (Şekil 3.20) fonksiyonlarıdır. Bu iki fonksiyon ikincil cihazı seçerek, kesmeleri kapatır. Transfer tamamlandıktan sonra kesmeleri açar ve ikincil cihazı serbest bırakır. Okuma fonksiyonunda adres yazma maskesinin değili ile "lojik ve" işlemine, yazma fonksiyonunda ise yazma maskesiyle "lojik veya" işlemine sokularak fonksiyonun ilk giriş parametresi elde edilir. Diğer bir deyişle yazma maskesi adres ile yazma veya okuma komutlarını birleştirmek için kullanılır. Transfer fonksiyonun ikinci parametresi yazma işlemi için yazılacak data, okuma işlemi için ise sıfır olarak verilir.

```
void RF22::spiWrite(Xuint8 reg, Xuint8 value)
{
    microblaze_disable_interrupts();
    _spi->setSS(_slaveSelectPin);
    _spi->transfer(reg | RF22_SPI_WRITE_MASK,value);
    _spi->setSS(0);
    microblaze_enable_interrupts();
}
```

Şekil 3.20: "spiWrite" Fonksiyonu

Ancak sıralı kayıtları okumak için "spiBurstRead" (Şekil 3.21) ve buralara yazmak için ise "spiBurstWrite" (Şekil 3.22) fonksiyonları kullanılır. Bu iki fonksiyon normal yazma ve okuma fonksiyonlarından faklı olarak uzunluğu kullanıcı tarafından belirlenen bir datayı istenilen adrese yazar veya istenilen adresten okur.

```
void RF22::spiBurstRead(Xuint8 reg,Xuint8* dest,Xuint8 len){
    Xuint8 sendbuff[len+1],rcvbuff[len+1],received[len];
    sendbuff[0]=reg & ~RF22_SPI_WRITE_MASK;
    for(int i=1;i<len+1;i++)
        sendbuff[i]=0;
    microblaze_disable_interrupts();
    _spi->setSS(_slaveSelectPin);
    XSpi_Transfer(&Hardware_spi.SPI,sendbuff,rcvbuff,len+1);
    _spi->setSS(0);
    microblaze_enable_interrupts();
    for(int i=1;i<len+1;i++)
        received[i-1]=rcvbuff[i];
    dest=received;
}
```

Şekil 3.21: "spiBurstRead" Fonksiyonu

```
void RF22::spiBurstWrite(Xuint8 reg,const Xuint8* src,Xuint8 len){

    Xuint8 sendbuff[len+1];
    //Xuint8 rcvbuff[len+1];//to see values in loopback mode
    microblaze_disable_interrupts();
    sendbuff[0]=reg | RF22_SPI_WRITE_MASK;
    for(int i=1;i<len+1;i++)
        sendbuff[i]=src[i];
    _spi->setSS(_slaveSelectPin);
    XSpi_Transfer(&Hardware_spi.SPI,sendbuff,0,len+1);//Normal mode
    //XSpi_Transfer(&Hardware_spi.SPI,sendbuff,rcvbuff,len+1);//Loopback mode
    _spi->setSS(0);
    microblaze_enable_interrupts();
    /*for(int i=0;i<6;i++)
        xil_printf("%d",rcvbuff[i]);*/
}
```

Şekil 3.22: "spiBurstWrite" Fonksiyonu

Ancak bu fonksiyonlar sınıf içinde donanımsal SPI sınıfı kullanılmadan gereçeklenmiştir. Eğer bu şekilde gerçekleme olmasaydı fonksiyonun sonunda uzunluğu kullanıcıya bağlı bir dizi döndürmek için dinamik hafıza kullanılması gerekecekti. Bu da bu fonksiyonun her kullanımından sonra dinamik hafızanın temizlenmesini gerektirecekti. Unutulan temizleme komutu hafızanın dolarak programın çalışamaz hale gelmesine neden olarak sistemin yeniden başlatılmasına dolayısıyla acil durumda çalışacak bu sistemin geçici bir süre durmasına sebep olacaktı. Özet olarak bu iki fonksiyon donanımsal SPI sınıfı kullanılmadan gerçeklenerek yazılımda herhangi bir açık bırakılmadan gerçeklenmiştir. Bunun dışındaki diğer fonksiyonlardaki ifadeler Mikroblaze ile uygun hale getirilerek yeniden yazılmış, testleri yapılmış ve çalıştığı gözlemlenmiştir.

Bahsedilen bu “spiWrite”, ”spiRead”, ”spiBurstRead” ve ”spiBurstWrite” fonksiyonları diğer fonksiyonların içinde okuma ve yazma işlemleri için kullanılmaktadır. Bu fonksiyonların düzgün bir şekilde çalışması iki modülün birbiri ile mikroblaze işlemci kontrolünde haberleşmesinde çok önemli bir role sahiptir. Bu fonksiyonları kullananan en önemli fonksiyonlardan birisi “init” fonksiyonudur. Bu fonksiyon RF22 modülünü çalışmasını başlatır. Bu fonksiyon içinde herhangi bir hata maydana geldiğinde UART üzerinden terminale hata basılır ve RF22 objesi veya modülü başlatılmaz. Bu şekilde yanlış ayarlamaların önüne geçilmiş olunur. Bu fonksiyon modülü sıfırlayarak ilgili kayıt adreslerine ilgili değerleri yazarak konfigürasyonu tamamlar.

Önemli fonksiyonlardan diğeri ise “reset” fonksiyonudur (Şekil 3.23). Adından da anlaşılacağı üzere sistemi sıfırlamaya yarar. Bu işlem, önceden adres değerleri ve içine yazılacak değerler değişken olarak tanmlananmış “RF22_REG_07_OPERATING_MODE1” adresine “RF22_SWRES” değeri yazılarak yazılımsal sıfırlama gerçekleşir. Donanımın kendini sıfırlaması için 10 ms beklenmesi yeterli olacaktır.

```
void RF22::reset()
{
    spiWrite(RF22_REG_07_OPERATING_MODE1, RF22_SWRES);
    // Wait for it to settle
    delay(10); // SWReset time is nominally 100usec
}
```

Şekil 3.23: “reset” Fonksiyonu

Bu kütüphane içinde Arduino'ya özgü atomic kütüphanesine ait atomic blok ifadeleri yer almaktadır. Bu ifadeler çalışma anında kesme işleminin bayraklarını (flag) kontrol ederek transfer esnasında kesmelerin kapatılmasını sağlamaktadır. Mikroblaze de ise kesmeler, "microblaze_disable_interrupts()" komutu ile kapatılmaktadır. Kesmelerin açılması ile "microblaze_enable_interrupts()" komutu ile yapılmaktadır. Atomik blok ifadeleri bu ifadeler ile değiştirilmiştir. Ayrıca Şekil 3.24'deki kod parçasında kesme anında çalışacak fonksiyon RF22 sınıfının "isr0" fonksiyonu yani kesme rutinine bağlanmıştır. Kesme tetiklendiğinde "isr0" fonksiyonu tetiklenir. Bu foknsiyon içinde "handleInterrupt" fonksiyonunun tetikler (Şekil 3.25). "handleInterrupt" fonksiyonu kesme kayıtlarını okur. Kesmenin hangi sebepten kaynaklandığını belirleyerek o ilgili işlemleri geerçekleştirir ve kesme kayıtlarını temizler. Kayıtlar temizlenince RF23 modülü kesme işaretini sıfırdan bire çekerek kesmeyi temizler ve sistem ana program akışıyla kaldığı yerden çalışmaya devam eder.

```
if (_interrupt == 0) //burada microblaze interruptlar baglanacak
{
    _RF22ForInterrupt[0] = this;
    XIntc_RegisterHandler(XPAR_INTC_0_BASEADDR,0,(XInterruptHandler)isr0,(void*)XPAR_INTC_SINGLE_BASEADDR);

//attachInterrupt(0, RF22::isr0, FALLING);
}
```

Şekil 3.24: Kesme IP'si ile RF22 Sınıfının Kesme Rutini Fonksiyonu Bağlantısı

Bu modülde kesmelerin ne zaman tetiklendiği ile ilgili bilgi Kesme bölümünde mevcuttur.

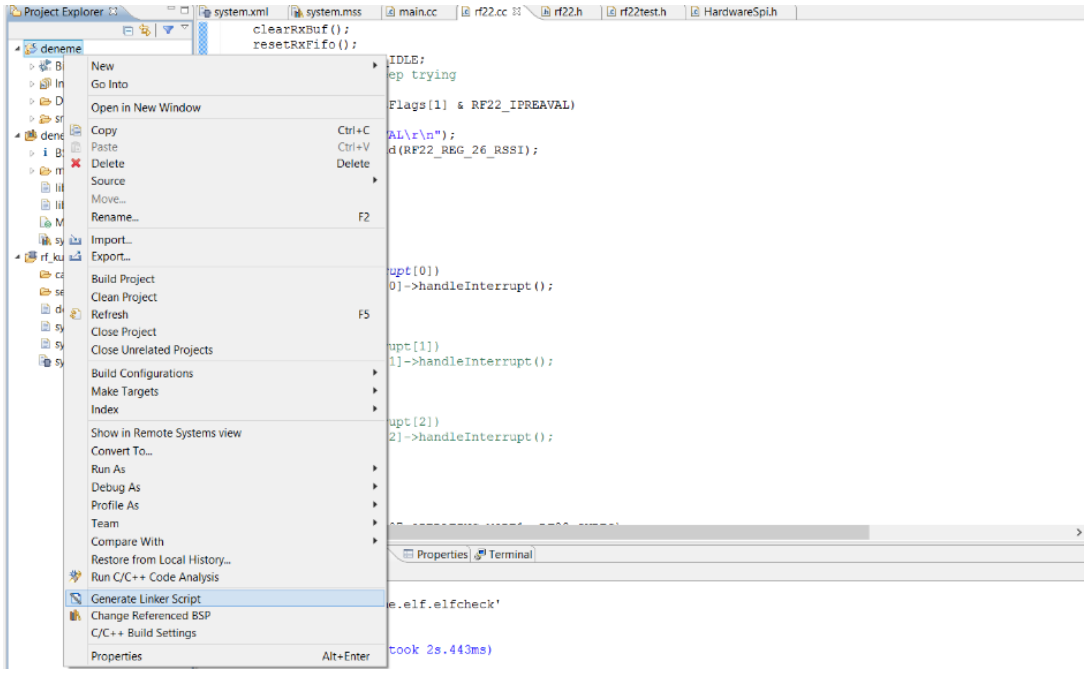
```
void RF22::isr0()
{
    if (_RF22ForInterrupt[0])
    _RF22ForInterrupt[0]->handleInterrupt();
}
```

Şekil 3.25: "isr0" Fonksiyonu

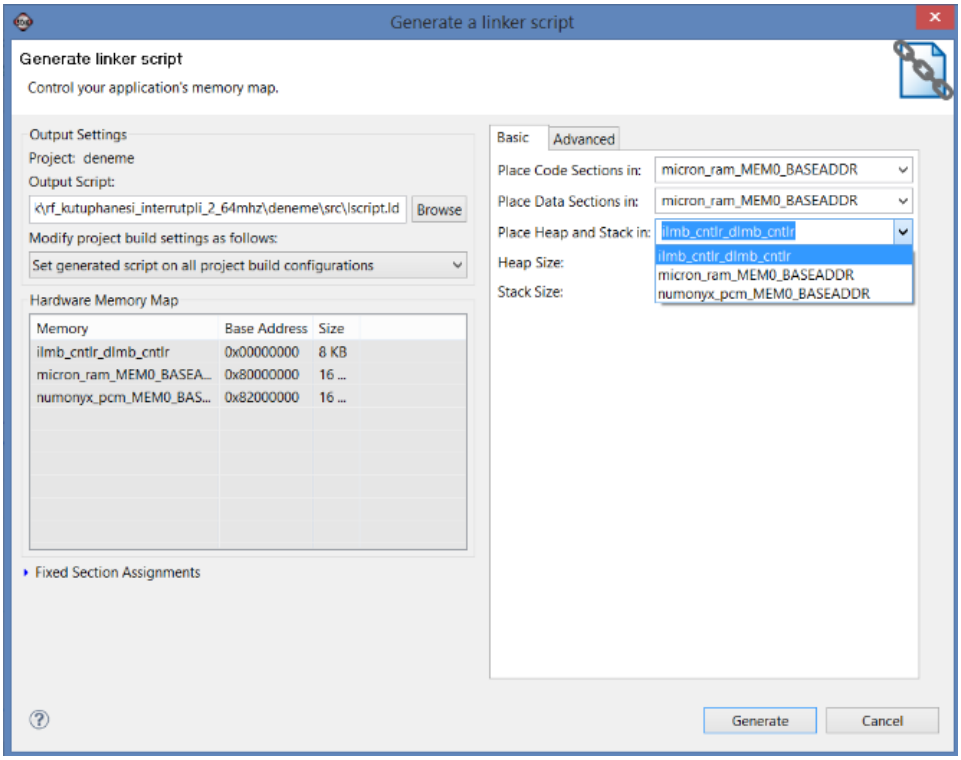
Bu fonksiyonlar dışındaki diğer fonksiyonlar projenin parçalrından modülün Arduino ile sürülmesi parçasında detaylı olarak anlatılmıştır [14].

Son olarak donanımsal SPI sınıfı ile RF22 sınıfı içindeki tüm fonksiyonların çalışabilmesi için programın hafıza birimi olarak hangi donanımı kullanacağının seçilmesi gereklidir. Bunun için öncelikle SDK ortamındaki projeye sağ tıklanarak açılan menüde (Şekil 3.26) "Generate Linker Script" komutu ile işlemcinin derleyici ayarlarının yapılacağı menü (Şekil 3.27) açılır. Bu menüde kod ve datanın saklanması

için Micron RAM seçilmiştir. Yığınlar (Heap ve Stack) için de aynı Mikron RAM seçilir. Bu şekilde mikroblaze işlemcide çalıştırılacak programın 16 MB'lık dış RAM hafızasını kullanması sağlanmıştır. Bu şekilde sınıflar için karşılaşılabilecek taşma sorunu çözülmüş olur.

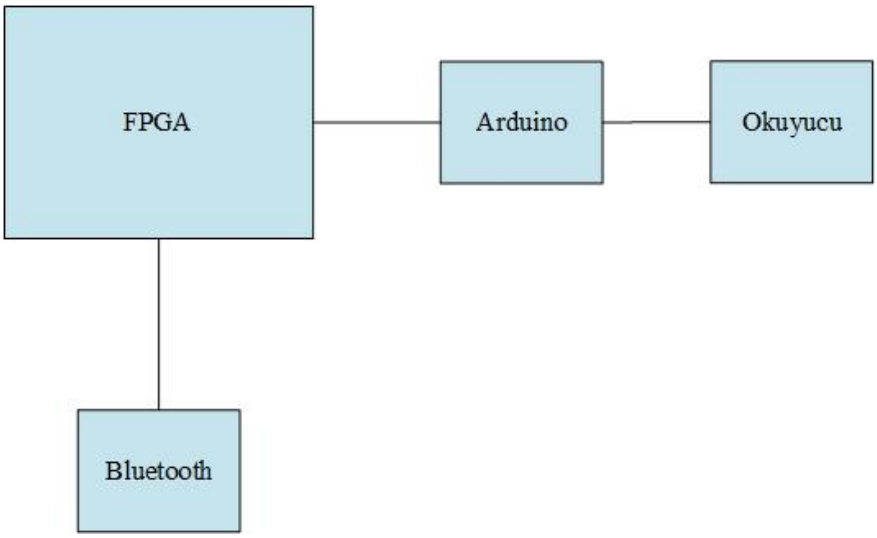


Şekil 3.26: Hafiza Ayarlanının Yapılandırılması için Linker Script Menüsünün Açılması



Şekil 3.27: Linker Script Menüsü ve Hafizanın Seçilmesi

Yapılan testler sonucunda Şekil 1.3'de belirtilen data toplama ünitesindeki FPGA kontrolünde RFID etiketler ve okuyucular sürekli bir şekilde haberleşememektedir. Bu durum yazma okuma FIFO'larının uygun şekilde kesme denetiminde temizlenemesinden kaynaklanmaktadır. Yapılan tüm denemelere rağmen bu hata giderilememiştir. Bunun yerine RF modül (okuyucu) ile FPGA arasına, baz alınan kütüphanenin sorunsuz çalıştığı Arduino konulmuş ve şekil 3.28'deki yeni data toplayıcı ünitesi elde edilmiştir.



Şekil 3.28: Önerilen Yeni Data Toplama Ünitesi

4 SONUÇLAR

Bu tezde, RFID iç mekan konum belirleme sisteminin data toplama ünitesinde okuyucu ile FPGA arasında donanımsal haberleşmeyi ve etiket ile okuyucunun haberleşmesini sağlayacak bir sistem tasarımı (Şekil 1.3) yapılmıştır. Bu sistemin gerçeklenmesi için Nexys3 geliştirme kartı üzerindeki Spartan 6 FPGA'sı üzerine Mikroblaze işlemci ve gerekli yan donanımlar kurularak donanımsal ve yazılımsal olarak yapılmıştır. Bu FPGA ile RFID okuyucu (modül) geliştirme kartı üzerindeki PMOD girişleri ile birbirine bağlanmıştır.

Yapılan çalışmalar süresince okuyucu olarak kullanılacak RF modülünün Arduino için hazırlanmış kütüphanesi ve Arduino üzerindeki habeleşmesi baz alınmıştır. Bu kütüphane mikroblaze ile derlenebilir şekilde nesneye dayalı programlama mantığında yeniden düzenlenmiştir. İletişimi sağlayacak yan donanımlar ve Mikroblaze işlemci FPGA içine gömülmüş hazırlanan bu yazılım işlemci üzerinde iki RF modülü haberleştirecek şekilde çalıştırılmıştır. Ancak okuyucu ve etiket RF modülleri birbirleriyle Mikroblaze işlemci kontrolünde kesintisiz bir haberleşme sağlayamamıştır. Bu durum gönderme ve alma esnasında FIFO'ların temizlenmeyerek iletişimi kesintiye uğratmasından kaynaklanmaktadır. Bu haberleşmenin sürekliliği yapılan bütün çalışmalara rağmen sağlanamamıştır. Bu hataların giderilmesi, Mikroblaze ile bu iki modülün kesintisiz haberleşmesini sağlayacaktır.

Bu sistem yerine, okuyucu RF modülü ile FPGA arasına bir Arduino koyularak (Şekil 3.28) data toplama ünitesi yeniden tasarlanmıştır. Burada Arduino üzerinde bahsedilen kütüphane çalışıtırılmıştır. Arduino kontrolünde haberleşen modüllerden gelen data yine Arduino kullanılarak FPGA üzerindeki Mikroblaze işlemcide okunabilmektedir. Bu data Bluetooth vasıtasıyla işlem birimine aktarılmaktadır.

KAYNAKLAR

[1] Zhou, Y., Liu, W. and Huang, P. (2007). Laser-activated RFID-based Indoor Localization System for Mobile Robots. In: *IEEE International Conference on Robotics and Automation*. IEEE, p.4600.

[2] Jin, G., Lu, X. and Park, M. (2006). An Indoor Localization Mechanism Using Active RFID Tag. In: *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*. IEEE Computer Society, p.1.

[3] Alçay, S. and İnal, C. (2010). Global Bir Ağda GPS/GLONASS, GPS ve GLONASS Sonuçlarının Karşılaştırılması. *Selçuk Üniversitesi Mühendislik ve Mimarlık Fakültesi Dergisi*, 25(1), p.29.

[4] Schuster, M. (2015). *Project*. [online] Baas-itea2.eu. http://baas-itea2.eu/cms/project-menu-item adresinden alındı. [10 Mayıs 2015 tarihinde erişildi].

[5] SPI: Overview and Use of the PICmicro Serial Peripheral Interface. (tarih yok). [ebook] pp.3. http://ww1.microchip.com/downloads/en/devicedoc/spi.pdf adresinden alındı. [7 Mayıs 2015 tarihinde erişildi].

[6] SPI: Overview and Use of the PICmicro Serial Peripheral Interface. (tarih yok). [ebook] Microchip Inc., p. 8, 11, 12. http://ww1.microchip.com/downloads/en/devicedoc/spi.pdf adresinden alındı. [7 Mayıs 2015 tarihinde erişildi].

[7] Wikipedia, (2015). *Serial Peripheral Interface Bus*. [online] http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus adresinden alındı. [11 Mayıs 2015 tarihinde erişildi].

[8] Users.ece.utexas.edu, (2015). *Chapter 12: Interrupts*. [online] http://users.ece.utexas.edu/~valvano/Volume1/E-Book/C12_Interrupts.htm adresinden alındı. [12 May 2015 tarihinde erişildi].

[9] Rosenthal, S. (1995). Interrupts might seem basic, but many programmers still avoid them. [online] Sltf.com.http://www.sltf.com/articles/pein/pein9505.htm adresinden alındı. [7 Mayıs 2015 tarihinde erişildi].

[10] Aktaş, M. (tarih yok). [online] İstanbul, pp.8. http://web.itu.edu.tr/orencik/BilgMimYenYakl2007/Mehmet_Aktas/FPGA_Mimarisi_Rapor.pdf adresinden alındı. [7 Mayıs 2015 tarihinde erişildi].

[11] Aktaş, M. (tarih yok). [online] İstanbul, pp.12,15. http://web.itu.edu.tr/orencik/BilgMimYenYakl2007/Mehmet_Aktas/FPGA_Mimarisi_Rapor.pdf adresinden alındı. [7 Mayıs 2015 tarihinde erişildi].

[12] Embedded System Tools Reference Guide - EDK 11.3.1, 2009, Xilinx Inc.

[13]RFM23 ISM Transceiver Module, Referans el kitabı, 2006, Hope Microelectronics CO., Ltd.

[14] Aktaş R., 2015. Distance Estimation Using Received Signal Strength for RFID Tracking System, *Lisans Tezi*, İ.T.Ü. Elektrik Elektronik Fakültesi, İstanbul.

[15] Nexys3 Kartı, Referans el kitabı, 2011, Digilent.

[16] Keysight.com, (2015). *16802A 68-Channel Portable Logic Analyzer | Keysight (Agilent)*. [online] http://www.keysight.com/en/pd-778704-pn-16802A/68-channel-portable-logic-analyzer?cc=TR&lc=eng adresinden alındı. [12 Mayıs 2015 tarihinde erişildi].

[17] Oran A., 2015. Hardware/Software Codesign of an RFID Based Indoor Localization System, *Lisans Tezi*, İ.T.Ü. Elektrik Elektronik Fakültesi, İstanbul.

[18] Azbar O., 2015. Implementation of an RFID Based Localization System on FPGA, *Lisans Tezi*, İ.T.Ü. Elektrik Elektronik Fakültesi, İstanbul.

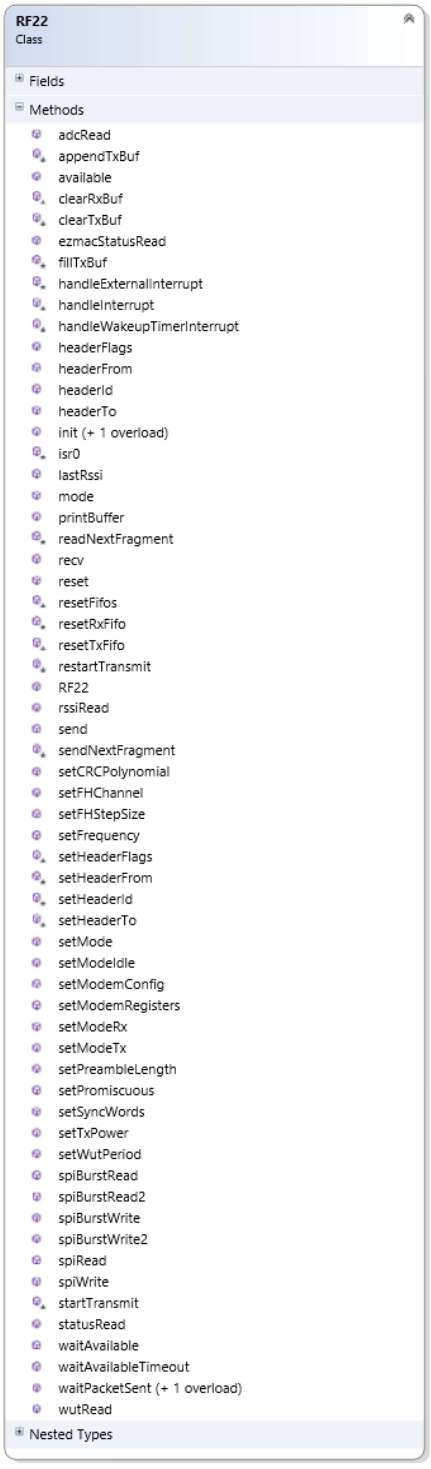
[19] McMahon, S. (tarih yok). Interrupt Creation and Debug on ML403. 1st ed. [ebook] pp.12-19. http://forums.xilinx.com/xlnx/attachments/xlnx/EMBEDDED/14753/1/Interrupt_debug.pdf adresinden alındı [7 Mayıs 2015 tarihinde erişildi].

[20] Agron, J. (tarih yok). How to Create and Program Interrupt-Based Systems: Microblaze-SOC + Interrupts.

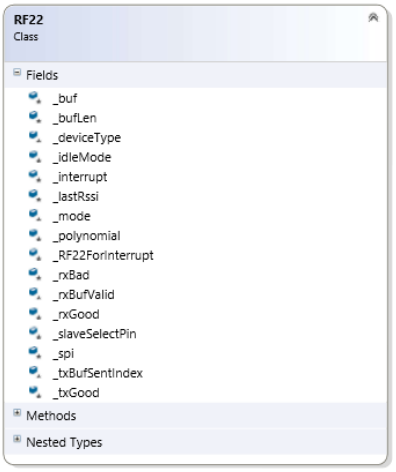
[21] LogiCORE IP XPS Serial Peripheral Interface (SPI) Reference Guide-(v2.02a), 2011, Xilinx Inc.

EKLER

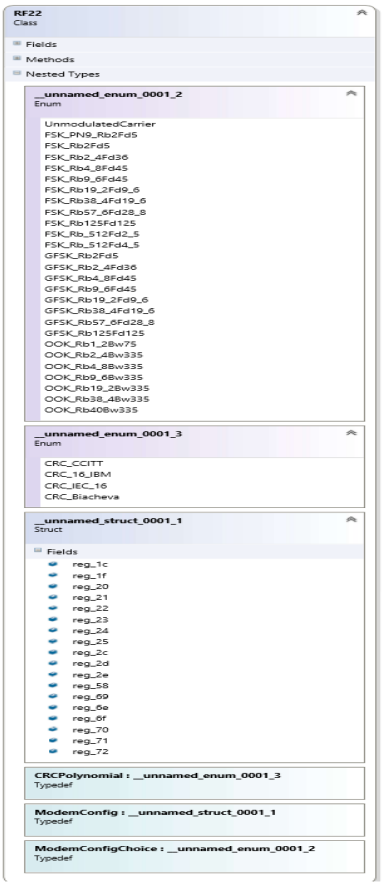
EK-A



Şekil A.1: RF22 Fonksiyonları

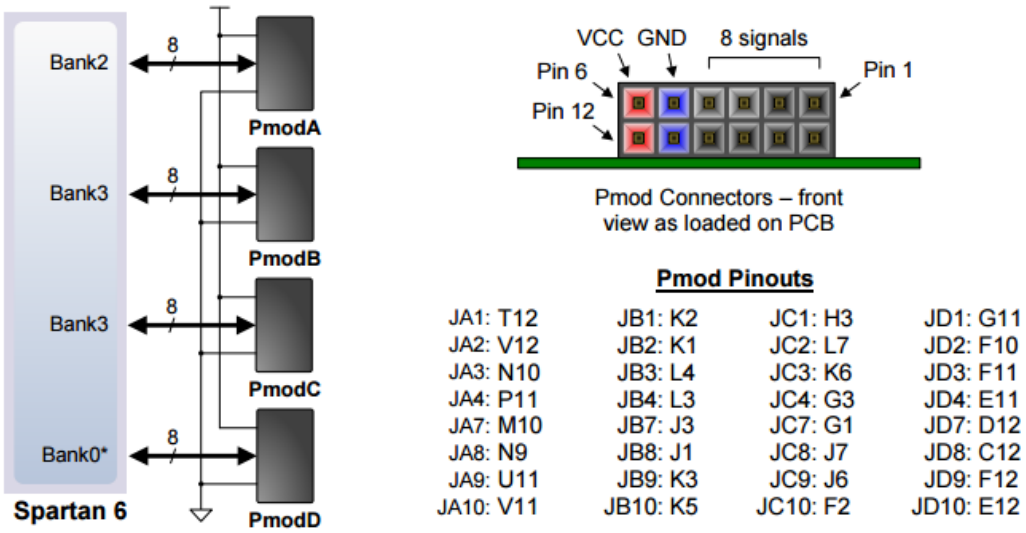


Şekil A.2: RF22 Sınıf Değişkenleri



Şekil A.3: RF22 İç Değişkenleri

EK-B



Şekil B.1 PMOD Bağlantı Şeması [14]

ÖZGEÇMİŞ

Adı Soyad: Oğuzhan Çik

Doğum Yeri ve Tarihi: Eskişehir, 1992

Lise: Tekirdağ Anadolu Öğretmen Lisesi,2006-2010

Lisans: İstanbul Teknik Üniversitesi, Elektronik-Haberleşme Mühendisliği; 2010 - 2015