T.C.

ULUDAĞ ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ

VHDL ve Verilog Donanım Programları ve Uygulama Alanları

Ahmed Melih Ulusoy

031511564

PROJE HAZIRLIK

BURSA 2021

T.C.

ULUDAĞ ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ

VHDL ve Verilog Donanım Programları ve Uygulama Alanları

Ahmed Melih Ulusoy

031511564

Projenin Danışmanı: DR. ÖĞR ÜYESİ ABDURRAHMAN GÜNDAY

BURSA 2021

Uludağ Üniversitesi Mühendislik Fakültesi tez yazım kurallarına uygun olarak hazırladığım bu Bitirme Projesi çalışmasında;

- Tez içindeki bütün bilgi ve belgeleri, akademik kurallar çerçevesinde elde ettiğimi
- Görsel, işitsel ve yazılı tüm bilgi ve sonuçları, bilimsel ahlak kurallarına uygun olarak sunduğumu
- Başkalarının eserlerinden yararlanılması durumunda, ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu
- Atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi
- Kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- Bu tezin herhangi bir bölümünü üniversitemde veya başka bir üniversitede başka bir üniversiteden başka bir tez çalışması olarak sunmadığımı

beyan ederim.

29.12.2020

Ahmed Melih Ulusoy
031511564

Danışmanlığımda hazırlanan Bitirme Projesi çalışması, tarafımdan kontrol edilmiştir.

04.01.2021

Dr. Öğr. Üyesi Abdurrahman Günday

ÖZET

Bu projede asenkron haberleşme protokolü olarak UART protokolü ve senkron haberleşme protokolü olarak ise SPI haberleşme protokolü baz alınarak FPGA ortamında VHDL dili kullanarak tasarım gerçekleştirilmiştir. Çalışmamızda UART alıcı ve verici, SPI ana ve bağımlı aygıtları Vivado Design Suite yazılımıyla simülasyon yoluyla sonuçlar açıklanmaya çalışılmıştır. SPI ana ve bağımlı cihazlar seçiminde her iki aygıtın FPGA olduğu varsayılmıştır. SPI 8 bit veri bir saat bölücüsü ile bir SPI saat oluşturulmaya çalışılmıştır. Bu SPI saatimiz ana FPGA aygıtımızda belirlenmiştir. UART protokolünün en önemli seçim nedeni günümüzde en çok kullanılan protokollerden biridir.

Projede temel hedef teorik olarak belirlediğimiz verileri en basit ve kolay yoldan anlaşılır bir şekilde simülasyon sonucunda gözlenmiştir. Simülasyonda verileri değiştirip tekrar tekrar FPGA kullanmadan gerçeğe yakın başarılı sonuçlar elde edilmiştir. Bundan sonraki kısımda projenin bir diğer aşaması olan çevre birimler ile hem donanımsal hem de yazılımsal olarak çevre birimleri ile bağlantı oluşturulacaktır. En yaygın kullanım SPI için SD kart ve sensörler UART için kontrol ve izleme ile protokoller birbirini destekleyecektir.

ABSTRACT

In this project, the UART protocol as the asynchronous communication protocol and the SPI communication protocol as the synchronous communication protocol were designed using VHDL language in the FPGA environment. In our study, we tried to explain the results by simulating UART receiver and transmitter, SPI master and slave devices with Vivado Design Suite software. In the selection of SPI master and slave devices, it is assumed that both devices are FPGA. An attempt was made to create an SPI clock with an SPI 8 bit data clock devider. The SPI clock is determined on our main FPGA device. The most important reason for choosing the UART protocol is one of the most used protocols today.

The main target of the project was observed as a result of the simulation in the simplist and most understandable way, we determined theoretically. In the simulation, successful results close to reality were obtained without changing the data and using FPGAs over and over again. In the next part, connection will be established with the peripheral units, which is another phase of the project, both in hardware and software. The most common use is SD card and sensors for SPI, control and monitoring for UART and protocols will support each other.

İÇİNDEKİLER

Sayfa No

ŞEKİLLER DİZİNİ

Sayfa No

ÇİZELGELER DİZİNİ

Sayfa No

1.GİRİŞ

Elektronik cihazlar arasındaki iletişim, insanlar arasındaki iletişim gibidir. Her iki tarafın da aynı dili konuşması gerekir. Elektronikte bu dillere iletişim protokolleri denir. Neyse ki, elektronik projelerin çoğunu oluştururken bilmemiz gereken sadece birkaç iletişim protokolü vardır. Bu tez konusunda en yaygın üç protokolden bazılarının temelleri hakkında tartışılacaktır: Seri Çevresel Arayüz (SPI) ve Universal Asynchronous Receiver/Transmitter (UART). İlk olarak, elektronik iletişim hakkında bazı temel kavramlara değinilmesi gerekmektedir. SPI ve UART, USB, Ethernet, Bluetooth ve WiFi gibi protokollerden biraz daha yavaştır, ancak çok daha basittirler ve daha az donanım ve sistem kaynağı kullanırlar. SPI ve UART, mikro denetleyiciler arasında ve mikro denetleyiciler ile sensörler arasında büyük miktarda yüksek hızlı verinin aktarılması gerek olmayan iletişim için idealdir.

Elektronik cihazlar, cihazlar arasında fiziksel olarak bağlanmış teller aracılığıyla veri bitleri göndererek birbirleriyle konuşurlar. Bit, bir kelimedeki harf gibidir ve bit ikilidir yalnızca 1 veya 0 olabilir. Bitler voltajlardaki hızlı değişikliklerle bir cihazdan diğerine aktarılır. 5 V ile çalışan bir sistemde, bit 0, 0 voltaj kısa bir darbe ile iletilir ve bit 1, 5 volt kısa bir darbe ile iletilir. Veri bitleri paralel ve seri biçimde iletilebilir. Paralel iletişimde veri bitlerinin tümü aynı anda, her biri ayrı bir kabloyla gönderilir. Seri iletişimde, bitler tek bir kabloyla tek tek gönderilir. SPI, birçok farklı cihaz tarafından kullanılan ortak bir iletişim protokolüdür. Günümüzde kullanım alanlarından birkaçı, SD kart modülleri, Radyo frekansı kimlikleme (Radio Frequency Identification – RFID) kart okuyucu modülleri ve 2.4 GHz kablosuz verici/alıcıların tümü, mikro denetleyicilerle iletişim kurmak için SPI kullanır. SPI haberleşmenin benzersiz bir avantajı, verilerin kesintisiz olarak aktarılabilmesidir. Sürekli bir akışta herhangi bir sayıda bit gönderilebilir veya alınabilir. UART ile veriler, belirli sayıda bit ile sınırlı olarak paketler halinde gönderilir. Başlatma ve durdurma koşulları, her paketin başlangıcını ve sonunu tanımlar böylece veri aktarım sırasında iletişim kesilir. SPI aracılığıyla iletişim kuran cihazlar, ana-bağımlı ilişkisi içindedir. Ana kontrol cihazıdır(genellikle bir mikro denetleyici), bağımlı(genellikle bir sensör, ekran veya bellek cihazıdır) ana birimden talimat alır.

SPI en basit konfigürasyonu bir ana cihaz ve bir bağımlı cihazdan oluşan sistemdir ancak bir ana birim birden çok bağımlı cihazı kontrol edebilir.

SPI'NIN AVANTAJLARI VE DEZAVANTAJLARI

SPI kullanmanın bazı avantajları ve dezavantajları vardır ve farklı iletişim protokolleri arasında seçim yapılırsa proje gereksinimlerine göre SPI ne zaman kullanılacağı bilinmelidir.

AVANTAJLAR

- Tam çift yönlü seri iletişim modunu destekler.
- I2C'den daha yüksek verim sağlar.
- Hız geleneksel asenkron seri iletişimden daha yüksektir.
- Arayüz donanımı çok basittir. Sadece basit bir kaydırma yazmacından oluşur.
- Çok basit donanım devresi nedeniyle güç tüketimi I2C ye oranla daha azdır,
- Bağımlılar, Ana cihazdan sağlanan bir saat frekansı kullandıklarından hassas osilatöre ihtiyaz duymazlar.
- Alıcı-Verici devresine gerek yoktur.

DEZAVANTAJLAR

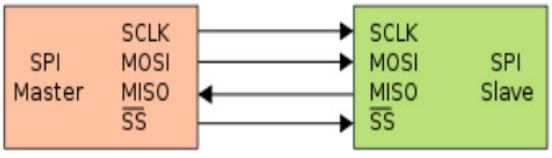
- I2C ve UART iletişimden daha fazla kablo gerektirir.
- Veri akış denetimi için donanım mevcut değildir.
- Ana ve bağımlı cihaz arasında bildirim yoktur.
- Bir seferde yalnızca bir ana cihaz olabilir.
- RS485, CAN ve LIN iletişimine kıyasla yalnızca kısa mesafeli iletişimi destekler.
- Bu iletişim, her bir bağımlı aygıt için bağımlı seçim (SS) veya entegre seçme hattı gerektirir.

Bu tezde, SPI ve UART çevresel birim haberleşmeleri ve uygulama alanı olarak ise (Field Programmable Gate Array – FPGA) üzerinde yoğunlaşılacaktır.

2. MATERYAL VE YÖNTEMLER

2.1 Seri Çevresel Arayüz Haberleşme Protokolü

Seri Çevresel Arayüz (Serial Peripheral Interface-SPI); mikroişlemciler veya mikrokontrolörler ile çeşitli yan birimler arasında iletişim kurmayı sağlayan bir haberleşme protokolü olup, yaygın olarak kullanılmaktadır. 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, ana (master) cihaz ile bağımlı (slave) cihaz arasında iletişim kurulmasını sağlayan, senkron bir iletişim protokolüdür. Ana ve bağımlı cihaz arasında veri alışverişine olanak sağlar. Senkron yani eşzamanlı veri iletimi denildiğinde iki aygıt başlangıçta kendilerini birbirleriyle senkronize eder ve ardından senkronize kalmak için sürekli olarak karakteri gönderir. Veriler gerçekten gönderilmediğinde bile sabit bir bit akışı, her bir cihazın diğer herhangi bir zamanda nerede olduğunu bilmesini sağlar. Yani, gönderilen her karakter ya gerçek veri ya da boş bir karakterdir. Senkron iletişimde, her bir veri baytının başlangıcını ve sonunu işaretlemek için ek bitler gerekli olmadığından, eşzamanlı olmayan yöntemlere göre daha hızlı veri aktarım sağlanır. Asenkron iletişimde, verilerin her bir baytının, başlangıcı ve bitişi, başlangıç ve durdurma bitleri tarafından tanımlanmalıdır. Bu ek iki bitin gönderilmesi gerekliliği, eş zamanlı olmayan iletişimin senkronize olmaktan biraz daha yavaş olmasına neden olmakla birlikte, işlemcinin ek boş karakterlerle uğraşmak zorunda olmaması avantajına sahiptir. Bu haberleşme protokolü dört pine sahiptir. Bunlar saat işareti (Serial Clock – SCK), ana giriş bağımlı çıkış (Master In Slave Out – MISO), ana çıkış bağımlı giriş (Master Out Slave In – MOSI) ve bağımlı seçici (Slave Select – SS)'dir. MOSI VE MISO pinleri aracılığı ile ana ve bağımlı cihazlar arasında veri bağlantısı kurulmuş olur. Saat işaretinin kontrolünde veri alışverişi aynı anda ve çift yönlü olarak gerçekleşir.



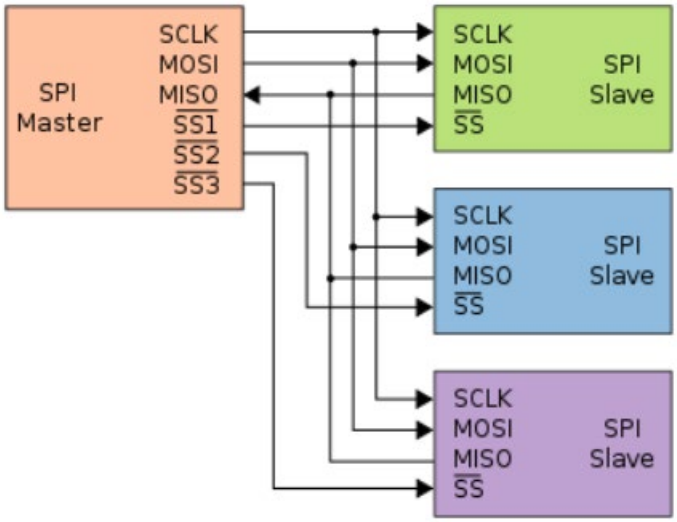
Şekil 2.1. SPI ana ve bağımlı cihazları ve bağlantı pinleri

Bir başka deyişle ana cihaz karşı tarafa veri gönderirken aynı anda karşı taraftan gelen verileri okur. SPI haberleşme protokolünün diğer bir pini ise SCK pinidir. SPI haberleşme protokolü diğer asenkron haberleşme protokollerinden farklı olarak (RS-232, RS-485, CAN vb.) farklı bir saat işareti ile senkronize edilir ve kontrol edilir. Bu saat işareti ana cihaz tarafından sağlanır, dolayısıyla SPI haberleşme protokolünün ayarları sadece ana cihaz tarafından yapılabilir. SPI saat işaretinin frekansı da ana cihaz tarafından belirlenir. Bu SPI saat işareti, genellikle işlem biriminin mevcut mikro işlemci veya mikro kontrolörün belirli oranlarda bölünmesi ile elde edilir. SPI saat işareti okuma ve yazma işlemlerinde verilerin ne zaman yazılıp okunacağını kontrol eder. Bu yazılma ve okunma işlemi saat işaretinin yükselen veya düşen kenarında yapılabilir. Bu noktada SPI haberleşmenin modlarından bahsetmek gerekir. SPI haberleşmenin 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 veri alışverişi yapılacağını belirler. Bu iki özelliği birlikte düşünerek Tablo 2-1'deki durumlar ortaya çıkar ki bu da SPI modları olarak adlandırılır. Bu modlarda ana cihaz ayarlanarak saat işaretinin istenilen kenarlarında ve aktifliğinde veri 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

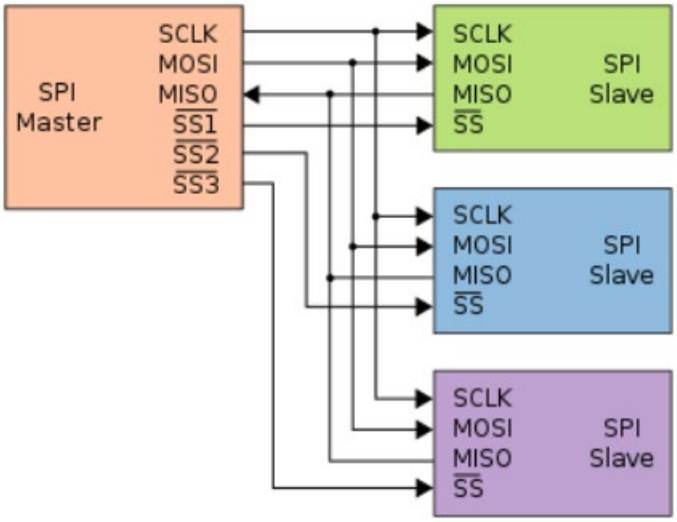
SPI protokolünün son pini ise SS pinidir. Bu pin master cihaza birden fazla slave bağlanması durumunda ana cihazın hangi bağımlı cihaz ile haberleşeceğine karar verir, diğer bir deyişle bapımlı cihazlar arasından bir tanesini seçer. SS pini sıfır aktif olarak çalışır. Bu haberleşme protokolünde aynı anda birden fazla bağımlı cihaz ile haberleşmek mümkün olmadığından hangi ikincil cihaz seçilmek isteniyorsa onun SS pini sıfır, diğerlerinin ise bir yapılması gerekmektedir. Seçili olmayan cihazlar serbest olarak kalırlar ve diğer bir birincil cihazla haberleşebilirler.



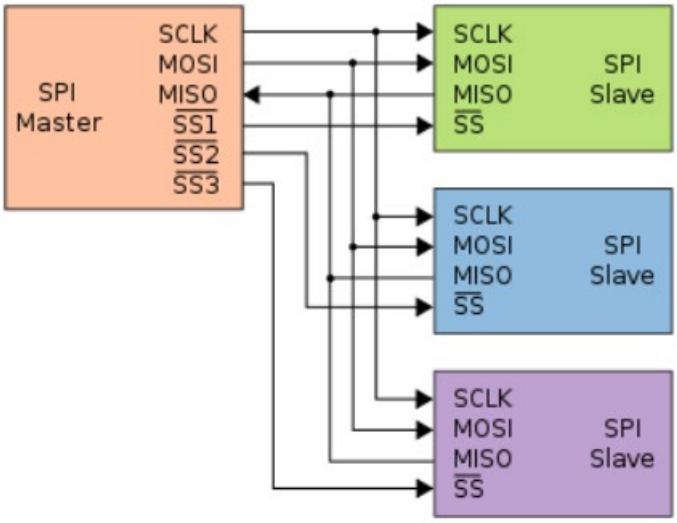
Şekil 2.2. SPI ana cihaza birden fazla bağımlı cihazın bağlanması

Ayrıca tek bir SS pini ile slave cihazların kaskat bağlanması mümkündür. Bu durumda bütün bağımlı cihazlar aynı anda seçildiğinden diğer bir ana cihaz tarafından seçilemezler.

Kaskat bağlanması işleminde bağımlı cihazın MISO pini bir sonraki bağımlı cihazın MOSI pinine bağlanarak kaskat bağlantı sağlanmış olur. Bu şekilde veriler ana cihazdan ilk bağımlı cihaza, ilk bağımlı cihazdan son bağımlı cihaza doğru geçerek ana cihaza ulaşır. Bu esnada bütün bağımlı cihazlar aynı anda seçilir. Bu sebeple bütün bağımlı cihazlar aynı anda seçildiğinden diğer bir ana cihaz tarafından seçilemezler.

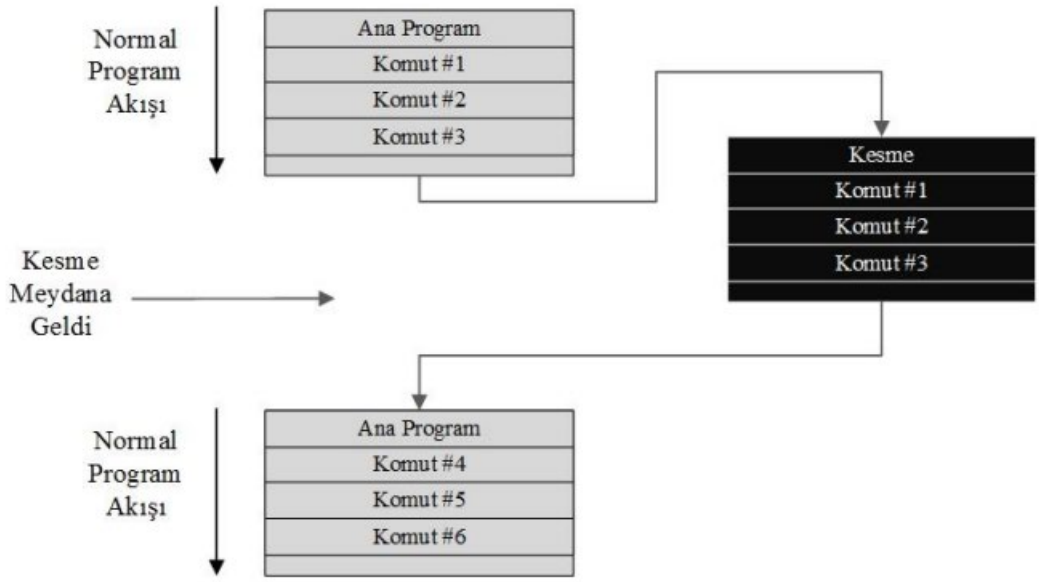


Şekil 2.3. Bağımlı cihazların kaskat bağlanarak ana cihaza bağlantı yapılması



Şekil 2.4. SPI ana cihaza birden fazla bağımlı cihaz bağlanması

Kesme, mikrodenetleyici veya mikroişlemci kullanılan sistemlerde meydana gelen istenmeyen durumların üstesinden gelmek amacıyla, sistem zamanlamasında veya sistemin açılıp kapanmasında sıkça kullanılmaktadır. Ayrıca bir veri batının Ethernet veya UART üzerinden iletilmesinde de kesme önemli bir 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 Servis Routine – ISR) tarafından program kaldığı yerden tekrar başlatılır. Aşağıda kesme işlem rutininin işleyişi gösterilmiştir. Yazılımsal ve donanımsal olmak üzere iki çeşit kesme vardır.

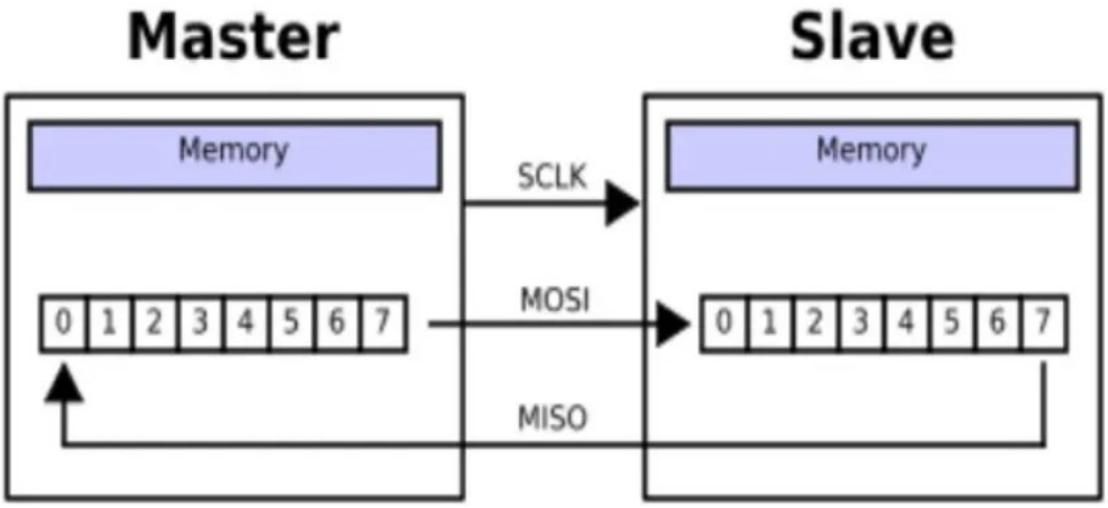


Şekil 2.5. Kesmenin işleyişi

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.

2.2 SPI Modül İç Yapısı

Bir ana cihaz ve bir bağımlı cihaz arasındaki veri aktarımı, aşağıdaki diyagramda gösterildiği gibi genellikle iki kaydırma yazmacından (Shift Register) oluşur. Bu kaydırma yazmaçları, hem ana hem de bağımlı cihaz için genellikle 8 bit boyutundadır.



Şekil 2.6. SPI ana ve bağımlı cihaz için iki kaydırma yazmacı

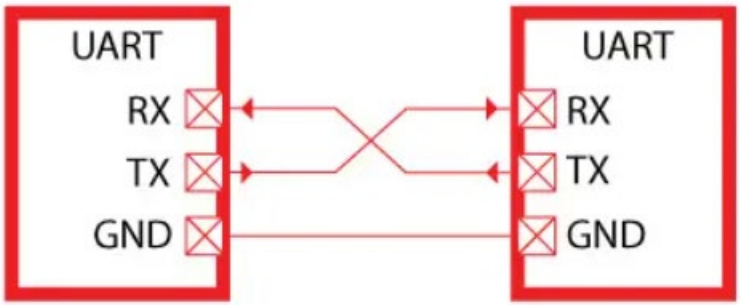
Genellikle iki kaydırma yazmacı dairesel bir tampon oluşturur. Veri aktarmak için bir ana cihaz aktif yüksek saat sinyalini bir bağımlı cihaz için çalıştırır ve saat sinyalinin frekansı, bir bağımlı cihazın çalışma frekansından düşük veya ona eşit olmalıdır. SPI cihazlar için maksimum frekans aralığı 1-70 MHz arasındadır. Bundan sonra ana cihaz, veri komutlarını aktarmak istediği bağımlı cihazı seçmek için SS pinini aktif düşük hale getirir. Her SPI saati, verileri tam çift yönlü modda aktarır. Bir bağımlı seçimi yapıldıktan sonra ana cihaz, bağımlı cihaza MOSI hattı üzerinden bir başlangıç biti gönderir ve bağımlı cihaz bu biti aynı zamanda okur. Ana cihaz, kaydırma yazmacını kullanarak bağımlı cihazla zamanı paylaşır. Verileri okuduktan sonra, bağımlı cihaz verileri kaydırma yazmacından belleğe depolar ve benzer işlem ana cihaz bağımlı cihazdan veri almak istediğinde gerçekleşir.

2.3 Evrensel Asenkron Alıcı-Verici

Evrensel Asenkron Alıcı-Verici (Universal Asynchronous Receiver-Transmitter – UART) asenkron seri iletişim gerçekleştiren özel bir donanım cihazıdır. Farklı baud hızlarında veri formatı ve iletişim hızlarının yapılandırılması için özellik sağlar. Bir sürücü devresi, iki devre arasındaki elektrik sinyal seviyelerini yönetir. UART iletişimi genellikle tek bir bileşen veya bir entegre devrenin parçasıdır. Bunu bir bilgisayar veya fare, monitör veya yazıcı gibi çevre aygıtları üzerinden iletişim kurmak için kullanabiliriz. Mikrodenetleyici çiplerinde, genellikle bir dizi özel UART donanım çevre birimi mevcuttur. UART iki cihaz arasındaki en basit iletişim protokollerinden biridir. Cihazlar arasına iki tel bağlayarak biri iletim hattı diğeri alıcı hat olmak üzere cihazlar arasında veri aktarır. Veriler, bir cihazdan diğerine bitler halinde teker teker aktarılır. Bu iletişim protokolünün temel avantajı, her iki cihazın da aynı çalışma frekansına sahip olması gerekmemektedir. Örneğin, farklı saat frekanslarında çalışan iki mikro denetleyici, seri iletişim yoluyla birbirleriyle kolayca iletişim kurabilir. Bununla birlikte, baud hızı olarak adlandırılan önceden tanımlanmış bir bit hızı, komutun her iki cihaz tarafından anlaşılması için genellikle her iki mikrodenetleyicinin flash belleğinde ayarlanır. İleten UART baytlarca veri alır ve bitleri sıralı bir biçimde iletir. Alıcı bitleri tam bir bayta yeniden birleştirir. Tek bir kablo üzerinden seri veri iletimi aslında birden fazla kabloyla paralel iletimden daha uygun maliyetlidir. İki UART cihazı arasındaki iletişim tek yönlü, tam çift yönlü veya yarı çift yönlü olabilir. Tek yönlü iletişim, sinyalin birinden diğerine hareket ettiği tek yönlü bir iletişim türüdür. Alıcı veriyi göndermek için yetkisi yoktur. Tam çift yönlü, her iki cihazın aynı anda iletişim gönderip alabildiği zamandır.

Bir UART, bir saat üreteci içerir. Bu, örneklemeye biraz zaman için izin verir. Aynı zamanda giriş ve çıkış kaydırma kayıtlarını da içerir. Bir okuma ve yazma kontrol mantığı vardır. Bir UART'ın isteğe bağlı diğer bileşenleri şunlardır; verici ve alıcı tamponları, ilk giren ilk çıkar (First In First Out – FIFO) tampon belleği, doğrudan bellek erişim (Direct Memory Access – DMA) denetleyicisidir. UART, SPI gibi diğer iletişim protokollerinden farklı bir aktarım protokolüne sahiptir. Bir mikrodenetleyicide fiziksel bir devre kaynağıdır. Aynı zamanda bağımsız bir entegre devre olarak da işlev görebilir. UART'ın önemli bir avantajı, veri iletmek için yalnızca iki kabloya gereksinim duymasıdır.

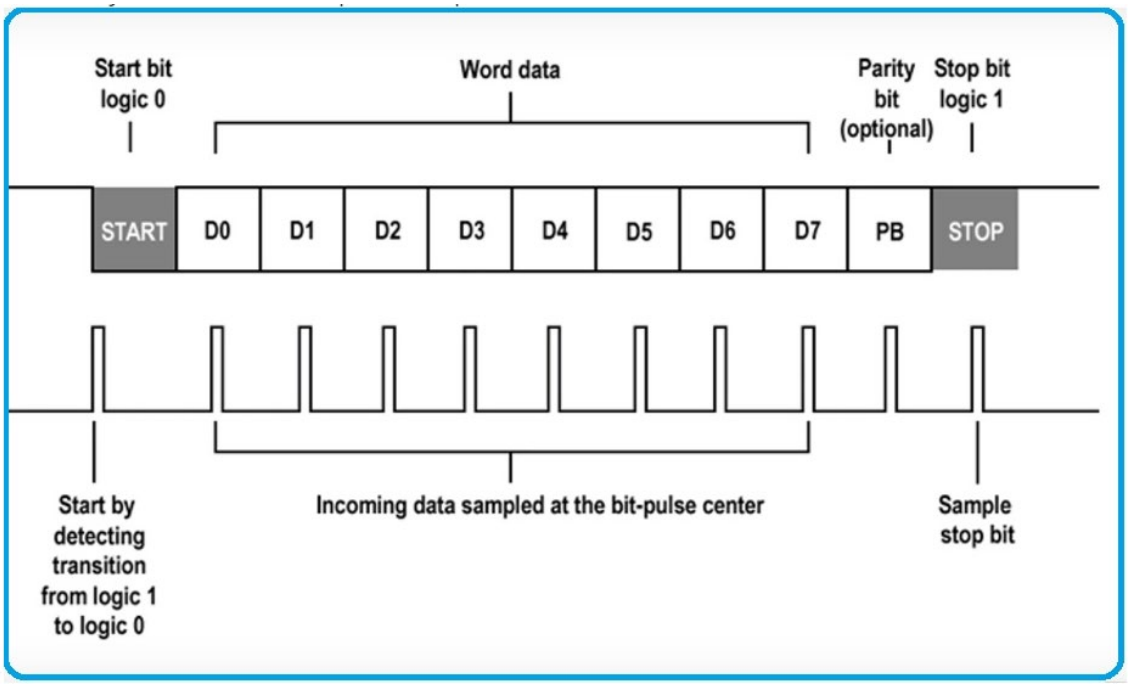
Doğrudan iletişim kurmak için iki UART gerekir. Bir uçta verici UART, paralel verileri bir merkezi işlem birimi (Central Process Unit – CPU) tarafından seri formata dönüştürülür, ardından verileri seri biçimde alıcı UART'a iletir bu da seri verileri alır ve tekrar paralel veriye dönüştürür. Bu verilere daha sonra alıcı cihazdan erişilebilir.



Şekil 2.7. UART alıcı-verici gösterimi

Saat sinyalleri yerine, gönderme ve alma biti, veri paketleri için başlatma ve durdurma bit sinyalleri kullanılır. Bu başlangıç ve bitiş bitleri, veri paketlerinin başlangıcını ve sonunu tanımlar. Bu nedenle alıcı UART, bitleri okumaya ne zaman başlayıp ne zaman duracağını bilir. Alıcı UART, başlangıç bitini algılayacak ve ardından bitleri okumaya başlayacaktır. Gelen bitleri okumak için kullanılan özel frekans, baud hızı olarak bilinir. Baud hızı, veri aktarım hızı için kullanılan bir ölçüdür. Baud hızı için kullanılan birim saniyedeki bit sayısıdır(bps) ve veri aktarımının başarılı olması için hem vericide hem de alıcıdaki UART hemen hemen aynı baud hızında çalışmalıdır. Ancak baud hızı, her iki UART arasında farklıysa, yalnızca %10 farklılık göstermelidir. Verici ve alıcı UART, aynı veri paketlerini alacak şekilde yapılandırılmalıdır.

Verici UART, bir veri yolundan veri alır. Bir veri yolu, mikrodenetleyici, bellek veya CPU gibi başka bir cihazdan veri göndermek için kullanılır. Gönderen UART verileri veri yolundan aldığında, bir başlangıç biti ve bir durdurma biti ekleyerek verileri işler. Bu da bir veri paketi oluşturur. Veri paketi bit bit Tx pininden seri olarak çıkarılır. UART daha sonra veri paketini Rx pini aracılığıyla parça parça okuyacaktır. Alıcı UART, başlangıç bitini, eşlik bitini ve durdurma bitlerini kaldırarak verileri orijinal biçimine geri döndürecektir. Alıcı, sonunda veri paketini veri yoluna paralel olarak aktaracaktır.



Şekil 2.8. UART zamanlama grafiğinin gösterimi

UART'ın ilettiği veriler şekilde görüldüğü gibi organize edilir. Bir başlangıç biti, 5 ile 9 veri biti içeren paketleri halinde düzenlenir. Bir eşlik biti isteğe bağlıdır.

▪**Başlangıç Bit:** Veri aktarılmadığında UART veri iletim hattı genellikle yüksek voltajdır. Transfer sürecini başlatmak için verici UART bir saat döngüsü boyunca yüksek voltajdan düşük voltaja geçer. Alıcı UART, yüksekten alçak gerilime geçişi algılayacak ve bitleri doğru baud hızında okumaya başlayacaktır.

▪**Eşlik Bit:** Eşlik biti, alıcı UART'a aktarım sırasında verilerin değişip değişmediğini söyleme işlevi görür. Farklı baud hızları veya uzun mesafeli veri aktarımı nedeniyle bitler değişebilir. UART, verileri aldıktan sonra veri çerçevesini okur daha sonra bit sayısını sayar ve çift mi tek mi olduklarını kontrol eder. Eşlik biti 0 ise o zaman çift eşliktir 1 ise tek eşliktir. UART'ın iletimin hatasız olduğunu bilmesi için eşlik bitinin verilerle eşleşmesi gerekir.

▪**Durdurma Bit:** En az iki bitlik süre için verici UART, iletim hattını düşük voltajdan yüksek voltaja yönlendirir.

UART protokolü aracılığıyla iki cihaz arasındaki iletişim, bitlerin aktarılmasıyla gerçekleşir. Bir bayt iletmek için arka arkaya toplam 8 bit gönderilir. Bir bit, mantıksal olarak düşük veya yüksektir. İki bit arasında zaman aralığına baud hızı veya bit hızı denir. Baud hızı, her iki cihazda da tanımlanmalıdır, böylece gönderici cihaz verileri bu belirli zaman aralığı ile bitlere kodlayabilir ve alıcı ardışık bitleri doğru zamanda bekleyebilir. En yaygın kullanılan baud hızları saniyede 9600 bittir. Diğer baud hızları da kullanılsa da, bit hızı ne kadar yüksekse veri bozulma olasılığı o kadar artar. İki cihaz arasında daha fazla fiziksel mesafe olduğunda daha düşük bit hızları kullanılır çünkü telin uzunluğu direnci arttırır ve dolayısıyla sinyal bozulur.

Veri baytını göndermeden önce alıcı cihaza verileri kaydetmeye başlaması için bilgi vermek üzere başlangıç biti gönderilir. Bu bit aktif düşüktür, yani alıcıyı bilgilendirmesi gerektiğinde bit düşüktür. Başlangıç bitinden sonra veri baytı gönderilir. Ardışık 8 bit iletilir ve bunların süresi baud hızına bağlıdır. Veri baytından sonra eşlik biti gönderilir. Bu bit isteğe bağlıdır ve aktarım sırasında veri bozulmasını kontrol etmek için kullanılır. Veri baytı iletilirken, veri baytının toplam 8 bitinden yüksek bit sayısı sayılır. Artık tek veya çift olmak üzere iki tür eşlik denetimi uygulanabilir. Örneğin tek sayı olan veri baytında yüksek bit sayısı 3 ise, tek eşlik olması durumunda eşlik biti 1'e ayarlanacak ve çift eşlik kontrolü durumunda sıfıra ayarlanacaktır. Bu eşlik biti, verici cihaz tarafından veri paketine kodlanır.

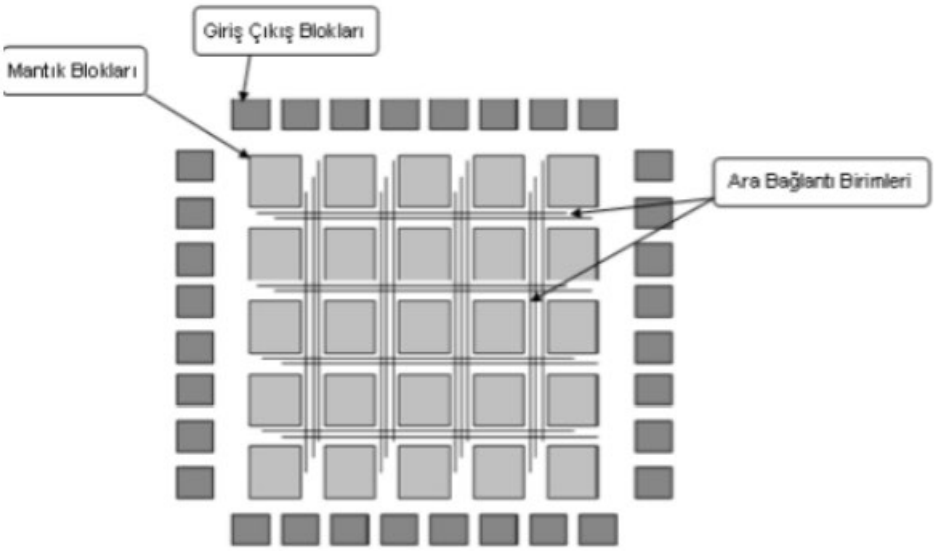
Şimdi alıcı veri paketini aldığında, veri baytındaki yüksek bitlerin sayısını da sayar ve bunu eşlik biti ile karşılaştırır. Eşlik biti veriyi onaylarsa, daha fazla işlenir ve veri baytındaki eşlik biti ve yüksek bit sayısı eşleşmezse veri bozuk olduğu için atılır ve hata sayacı artırılır. Bu hata sayacı belirli bir değeri aşarsa düşük hız pahasına veri kalitesini artıracak olan baud hızının düşürülmesi tavsiye edilir. Eşlik bitinden sonra, alıcıya veri paketinin sonunu bildirmek için kullanılan durdurma biti verilir. Ayrıca iletişim hızını yavaşlatmak için alıcıya 2 durdurma biti göndermeyi ve hata oranını düşürmek için ardışık veri paketleri alma sırasında yeterli zaman vermeyi seçebiliriz.

UART haberleşme protokolünü özetlersek, verici UART verileri paralel olarak veri yolundan alır. Verici UART, veri paketine başlangıç, eşlik ve durdurma bitlerini ekler. Paketin tamamı, verici UART'tan alıcı UART'a seri olarak gönderilir. Yapılandırılmış baud hızını kullanarak alıcı UART veri paketini örnekler. Alıcı UART, verileri orijinal formuna geri dönüştürür ve ardından kullanabileceği ve görselleştirebileceği veri yoluna aktarır. Verici ve alıcı UART cihazların düzgün çalışması için aynı bit, eşlik biti ve durdurma bitlerine ayarlanması gerekir. UART, yaygın olarak kullanılabilen bir yöntemdir. Dezavantajlarından bazıları ise, veri çerçevesi 9 bit ile sınırlıdır ve verici ve alıcı UART baud hızları yüzde on içinde olmalıdır.

2.4 Alanda Programlanabilir Kapı Dizisi

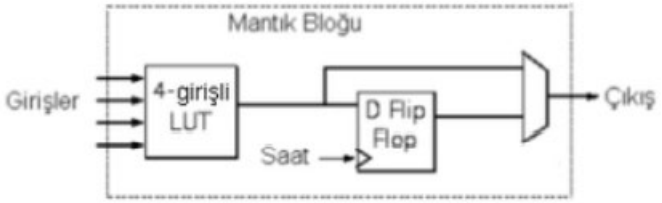
Alanda programlanabilir kapı dizileri (Field Programmable Gate Array – FPGA), programlanabilir yarı iletken devrelerdir. Uygulamaya özel tümleşik devre (Application Specific Integrated Circuit – ASIC) tasarımı kolaylaştırmak ve maliyeti düşürmek amacıyla geliştirilmiş ve günümüze kadar da geliştirilmesi devam etmiştir. Günümüzde FPGA'ler birçok alanda yaygın olarak kullanılmaktadır.

FPGA entegre devreleri Şekil 2.6'daki gibi programlanabilir mantık blokları (Configurable Logic Block – CLB) ve bu blokları birbirine bağlayan, değiştirilebilir ara bağlantılardan oluşur.



Şekil 2.9. FPGA iç yapısı

Burada CLB yapıları Şekil 2.7'de görüldüğü üzere eski nesil FPGA'ler için dört girişli bir doğruluk tablosu (Lookup Table – LUT), bir D flip flop ve çoklayıcıdan oluşmuştur. 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. Dört girişli bir LUT 16'lık bir doğruluk tablosunu hafızada tutabilirken altı girişli LUT'lar 64'lük bir doğruluk tablosunu hafızasında tutabilir.



Şekil 2.10. 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(Layout) 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 elemanların 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 ardından oluşturulan bit uzantılı dosya yardımıyla FPGA programlanır ve tasarlanan sistem FPGA üzerinde çalıştırılır.

Ayrıca bu işlemlerin her birinin ardından FPGA ‘ye yükleme yapmaksızın gerçekleme yapmak mümkündür. Bu gerçeklemeler davranışsal ve PAR sonrası gerçekleme olarak ikiye ayrılır. Davranışsal gerçekleme yazılan VHDL 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 grafiksel gecikmeler gözlenmektedir.

2.5 VHDL Nedir?

Çok yüksek hızlı tümleşik devre donanım açıklama dili (Very High Speed Integrated Circuit Hardware Description Language – VHDL), donanımı açıklamak için kullanılan bir açıklama dilidir. Elektronik tasarım otomasyonunda, entegre devreler(Integrated Circuit – IC) ve FPGA gibi karışık sinyal ve dijital sistemleri ifade etmek için kullanılır. VHDL’ye genel amaçlı bir paralel programlama dili olarak da kullanabiliriz. Tasarım bir sentez programı tarafından işlenir, süreçteki bir sonraki adım, mantık tasarımını test etmek için bir simülasyon programını içerir. Bu adımda, tasarımla arayüz oluşturan mantık devrelerini karakterize etmek için simülasyon modellerini kullanırız. Bu simülasyon modelleri koleksiyonunu bir test tezgahı (Testbench) olarak adlandırıyoruz.

Tipik olarak bir VHDL simülatörü olay tetiklemeli bir simülatördür, yani her bir işlemi belirli bir zamanlanmış zaman için bir olay kuyruğuna eklediğimiz anlamına gelir. Sıfır gecikmeye izin verilse de, yine de planlanmalıdır ve bu senaryolar için bir delta gecikmesi kullanıyoruz. Bir delta gecikmesi, sonsuz küçük bir zaman adımını temsil eder.

VHDL’in sistem tasarımında kullanımıyla ilgili kritik avantajı, tasarımın sentez araçlarının gerçek kapılara ve donanıma dönüştürülmesinden önce temel sistem davranışının doğrulanmasına ve modellenmesine izin vermesidir. VHDL kullanımı, kullanıcıya eşzamanlı sistem açıklamaları sağlar. VHDL bir veri akış dilidir, yani yürütme için her ifadeyi aynı anda dikkate alabilir. Bu C, Assembly ve Basic gibi programlama dillerinin tam tersidir. Bu dillerin her biri, hem sıralı olarak hemde bir seferde tek bir talimat olmak üzere bir dizi ifadeyi çalıştırır. VHDL’in en önemli avantajlarında biri, projelerinin çok amaçlı olmasıdır; projeyi bir kez oluşturursunuz ve

hesaplama bloğunu diğer çeşitli projelerde kullanabilirsiniz. Bu parametreler, eleman, bellek boyutu, ara bağlantı yapısı ve kapasite gibi çeşitli değişiklikler yapabileceğiniz anlamına gelir.

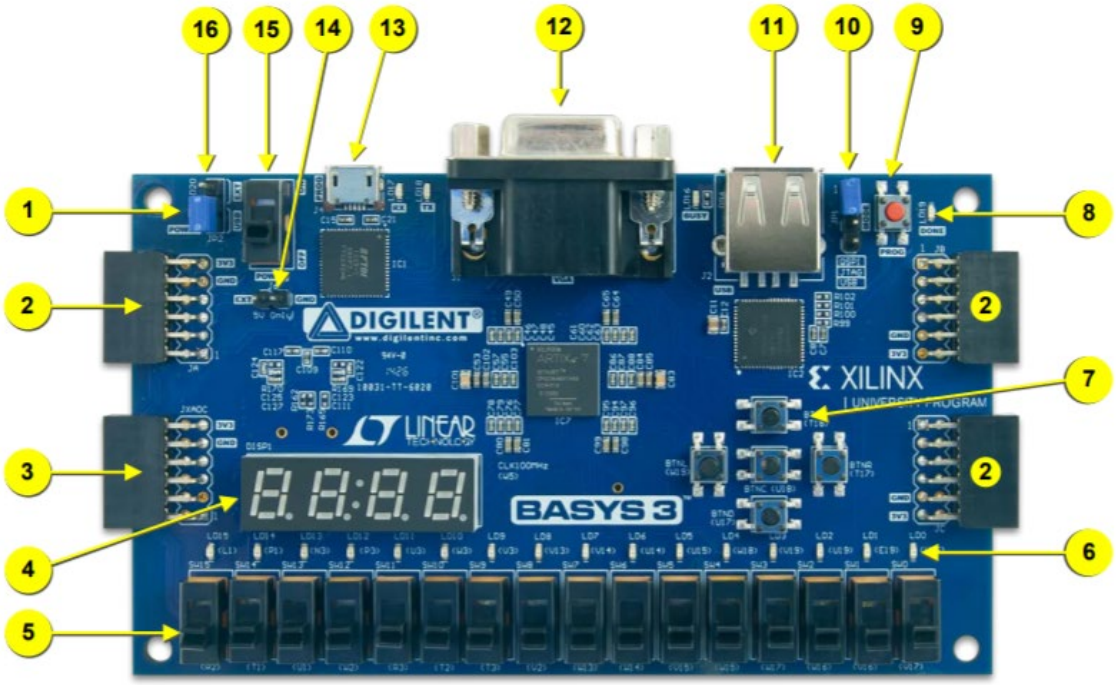
2.6 VHDL ve Verilog Farkları

VHDL güçlü tiptir (Strongly Type), Verilog ise zayıf tip (Weak Type) burada güçlü ve zayıf tipten kasıt, dilin derleme sırasında daha katı yazım kurallarına sahip olması veya olmamasıdır. Programcılar arasında teknik bir tanımı yoktur. Genel olarak, programlama dilinde değişkenlerin içeriklerinin türlerine uygun olduğunun sıkı kontrol edilme özelliğidir. Örneğin, C ve Java örneği verilebilir. VHDL'in anlaşılması kolaydır Verilog'da daha az kod ile yazılabilir. VHDL kullanımda daha doğaldır Verilog'da ise daha çok donanım modelleme ile ilgilidir. VHDL sözü utan bir yapısı vardır Verilog'un ise kısa ve özdür. VHDL, C'ye benzemeyen bir sözdizimine sahiptir Verilog ise C diline daha çok benzer. VHDL'de değişkenler veri türüne göre tanımlanmalıdır Verilog'da ise daha düşük seviyede programlama yapıları kullanılır. VHDL, FPGA ve askeri uygulamalarda daha çok kullanılırken Verilog, donanım modelleme konusu daha iyi başarır. VHDL öğrenilmesi daha zordur, Verilog'un ise öğrenilmesi daha kolaydır. Hangi dilin daha üstün olduğu konusunda farklı görüşler var, ancak bu kişisel tercihlere bağlıdır.

Verilog, daha çok gerçek bir donanım modelleme dili olduğu için daha kısa ama etkilidir. Verilog donanım modellemede üstün bir kavrayışa ve daha düşük düzeyde programlama yapısına sahiptir. Verilog, kompakt yapısını açıklayan VHDL kadar sözlü değildir. VHDL ve Verilog benzer olmasına rağmen farklılıkları benzerliklerinden daha ağır basma eğilimindedir.

2.7 BASYS3 FPGA Geliştirme Kartı

Bu projenin tasarlanması konusunda Digilent firması tarafından üretilmiş olan Şekil 2.8'deki Basys 3 FPGA geliştirme kartı baz alınmıştır.

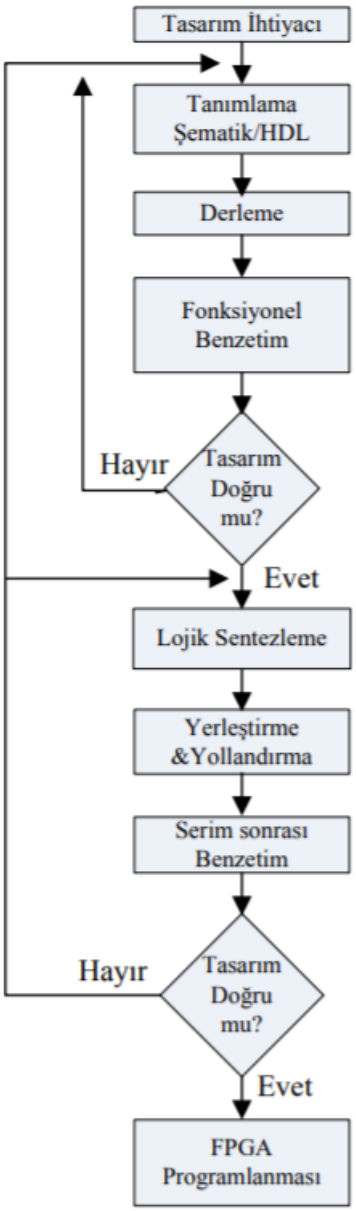


1	Power good LED	9	FPGA configuration reset button
2	Pmod connector(s)	10	Programming mode jumper
3	Analog signal Pmod connector (XADC)	11	USB host connector
4	Four digit 7-segment display	12	VGA connector
5	Slide switches (16)	13	Shared UART/ JTAG USB port
6	LEDs (16)	14	External power connector
7	Pushbuttons (5)	15	Power Switch
8	FPGA programming done LED	16	Power Select Jumper

Şekil 2.11. Basys3 FPGA geliştirme kartı pin ve komponent tanımlama

Basys3 FPGA kartı, en son Artix-7'yi temel alan eksiksiz, kullanıma hazır bir dijital devre geliştirme platformudur. Artix-7'nin içerdiği elemanlar 5200 adet dilimde 33.280 lojik hücre (her dilim 4 adet 6 girişli LUT ve 8 adet flip-flop içerir), 1.800 Kbits hızlı blok RAM, 5 adet saat yönetim başlığı, 90 adet dijital sinyal işlemcisi (Digital Signal Processing - DSP) birimi, 450 MHz'i aşan dahili saat hızı, dahili örneksel sayısal çevirici (XADC) özelliklerine sahiptir. Basys3 kartı, Xilinx Artix 7-FPGA mimarisine sahip,

Vivado Design Suite için dizayn edilmiş, giriş seviyesinde bir FPGA kartıdır. Basys3 kartı popüler Basys başlangıç kartları serisinin en yeni üyesidir. Bütün Basys kartlarında bulunan, tamamen kullanıma hazır donanım, geniş yelpazedeki dahili giriş ve çıkışlar, bitün FPGA yardımcı devreleri ve ücretsiz geliştirme araçları gibi özelliklere sahiptir.



Şekil 2.12. FPGA kullanılarak tasarım sürecinin genel akış diyagramı

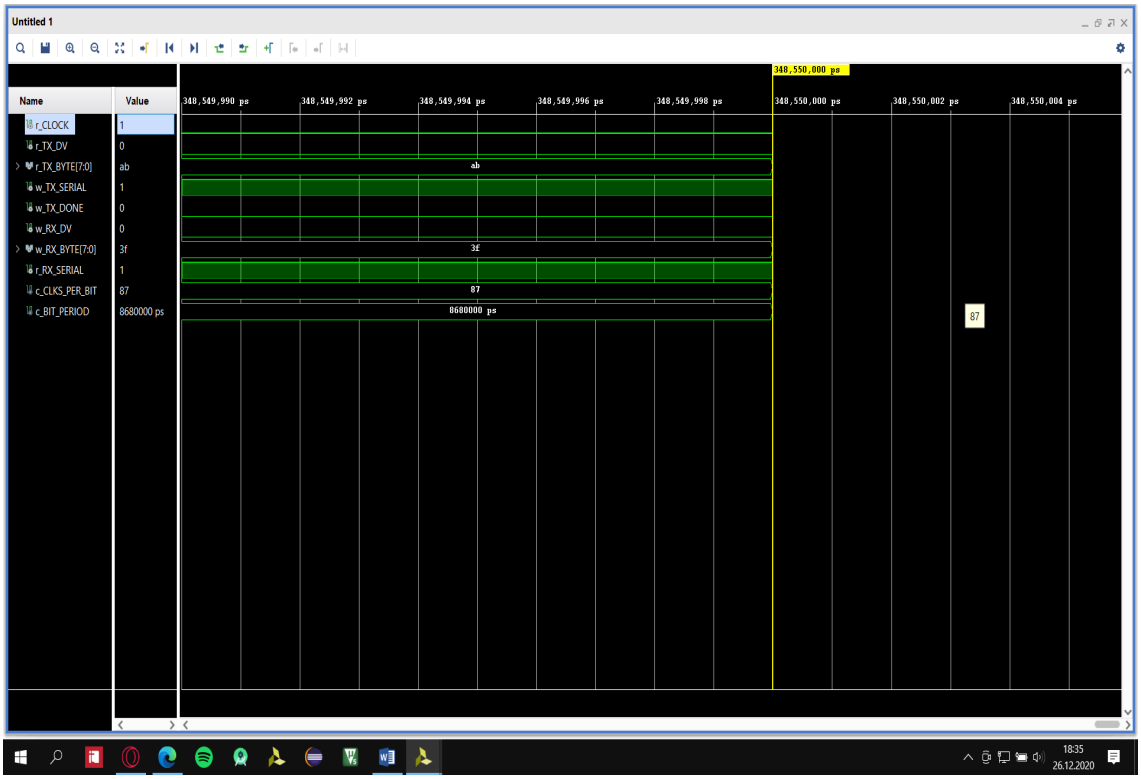
3. KAYNAK ARAŞTIRMASI

Bu projede kaynak olarak çoğunlukla internet sitelerinden, youtube videolarından ve VHDL bazlı FPGA kitaplarından yararlanılmıştır. Öncelikli SPI için değinilen temel kaynak bu protokolün esaslarını belirleyen firma olan Motorola'nın yayınlamış olduğu veri sayfalarıdır. Bu konuda bakılan bir diğer kaynak bu konuda yazılmış tezlerdir. Bunlardan ilki "SPI Implementation on FPGA- Trupti D. Shingare, R. T. Patil" çalışmasında giriş bilgileriyle bölüm açıklanmış sonra SPI blok şemalarıyla açıklanmıştır. Adım adım işlem basamakları verilmiştir. En sonunda tasarım blok diyagramıyla algoritma oluşturulmuştur. UART için kaynak taramasında Çavuşlu., A.Ç., 2017, "VHDL ile SAYISAL TASARIM ve FPGA UYGULAMALARI" Kodlab Yayınevi, İSTANBUL. Bu kaynak ile hem VHDL öğrenmede hem de UART projesinde danışılan bir kaynak olmuştur. Bir diğer VHDL ve FPGA için kaynak KELLECİ., B., 2017, "VHDL ve VERİLOG ile sayısal tasarım" Seçkin Yayıncılık, ANKARA. Bu kaynak ise hem piyasa çokça yararlanılan bir başucu kitabı görevini görmüştür. Kitaplar ile VHDL örnekleri yapılmış, simülasyon ve tasarım arayüzlerine çalışılmıştır. Projeler örneklerle desteklenmiştir.

Proje sadece simülasyon düzeyinde gerçekleşmiştir. FPGA ortamı Vivado Design Suite üzerinden sağlanmıştır. Projenin fiziksel olarak gerçeklemesini gördüğüm bazı kaynaklarda Microblaze işlemci ile diğer mikrodenetleyicilerle haberleşmesi sağlanmış ve VHDL kodu ile değil üzerinde özelleşmiş birimlerle IP'ler kullanılmıştır. Bir lojik analizör ile çıkışların izlenmiş olduğu görüldü. Youtube videoları ile haberleşme protokolleri hakkında örnekler üzerinde çalışma prensipleri araştırılmıştır. Xilinx Community ise çok iyi bir danışma platformu sorduğunuz sorulara cevap veren moderatörler, sorulara hızlıca cevap verip soru işareti kalmaması açısından caziptir.

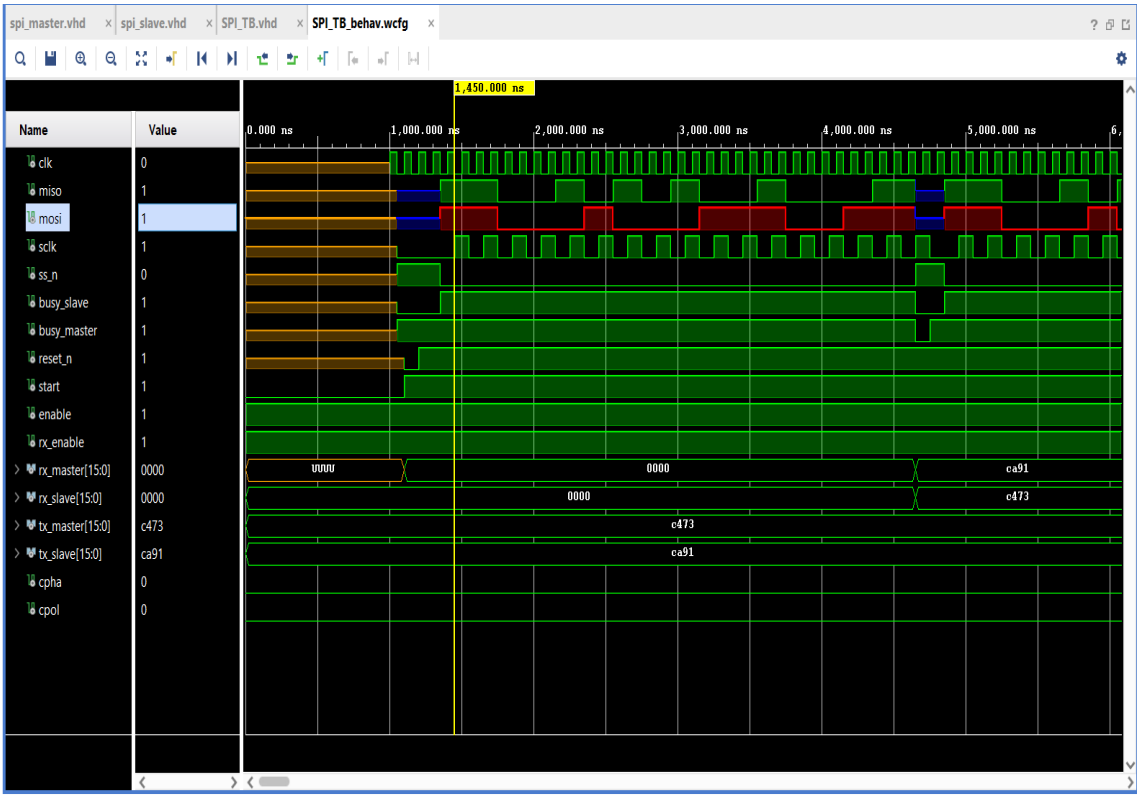
4. ARAŞTIRMA SONUÇLARI

İlk projemiz olan UART protokolü için VHDL kodu yazılmış ve bu kodun sonuçları aşağıda ekran görüntüleri paylaşılmıştır. UART, paralel ve seri formlar arasında veri çevrimi yapan bilgisayar donanım parçalarından biridir. UART'lar RS-232, RS-485 gibi yaygın iletişim standartları ile birlikte kullanılır. Aşağıda VHDL dilinde UART modülü ile veri alma ve gönderme işlemlerinin gerçekleştirildiği örneğe ilişkin ekran görüntüleri paylaşılmıştır. Yazdığımız kodlardan bazı yerlerin açıklanması gerekmektedir. Bir CLK_BIT değişkenimiz tanımlanmış olsun bu değişken her bir bit değeri için gerekli saat darbe sayısını vermektedir. Ana saat frekansının UART'ın frekansına bölümü ile bu değer bulunabilir. Örneğin 10 MHz saat frekansımız olsun UART frekansı 115200 olarak belirlenirse CLK_BIT değerimiz 87 olarak bulunur. İlk yapılan gönderim işleminde gönderdiğimiz bit "AB" değerini göndermiş olalım aldığımız veri ise "3F" olsun. Aşağıdaki ekran görüntisini elde ederiz.



Şekil 4.1: UART simülasyon sonucu

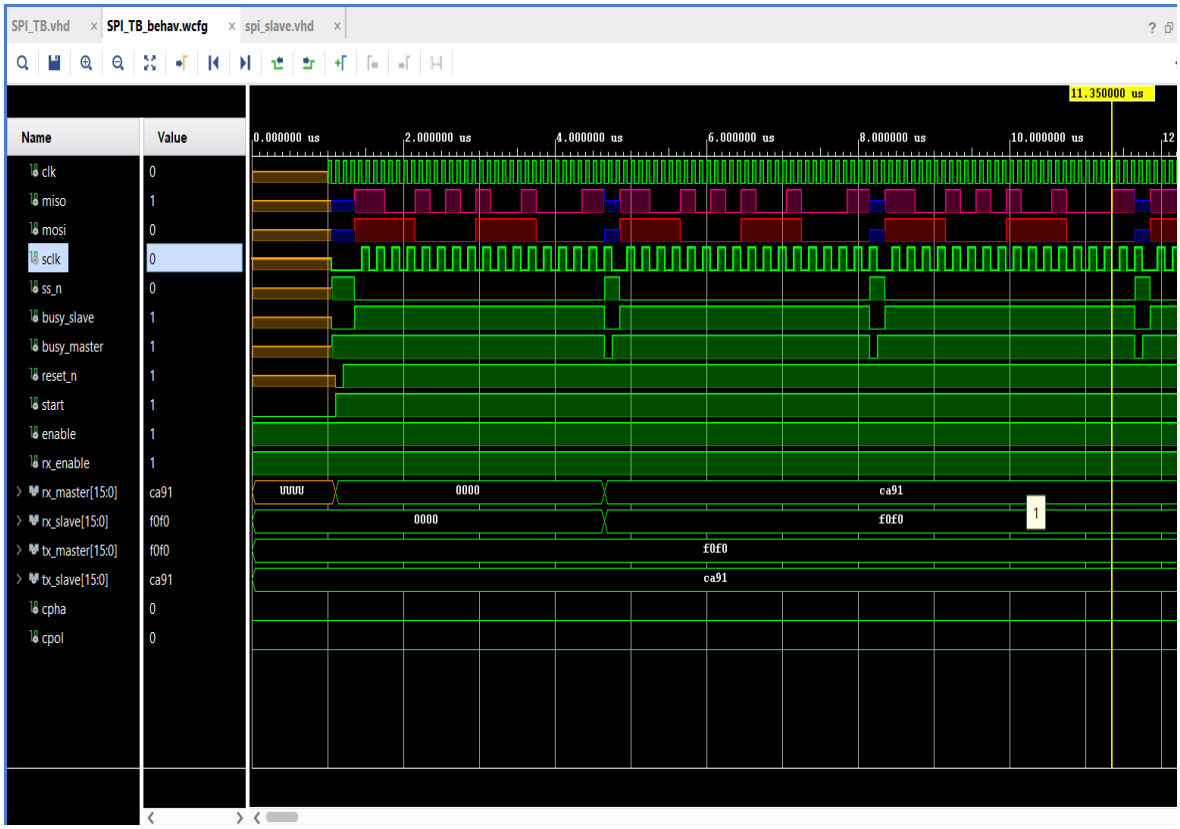
Şimdi de geriye SPI protokolü için simülasyon sonuçları paylaşılacaktır. Projemiz simülasyon ortamındaki sonuçları başarılı bir şekilde çıkışlarda gözlemlenmiştir. Simülasyon da ana cihaz için gönderilecek veri 16 bitlik "1100010001110011" verisini belirledik. Simülasyon sonucunda ise MOSI hattında en önemli bitinden başlayarak seri olarak gönderilmesi gerektiğini biliyoruz. Bu bilgilere dayanarak simülasyon sonucumuz,



Şekil 4.2. Ana cihaz için "1100010001110011" bitinin MOSI hattından izlenimi

Buna göre ana cihaz için yaptığımız simülasyon başarıyla gerçekleştirilmiştir. Buna göre farklı bit değeri için simülasyon bir kez daha tekrarlanırsa yine aynı sonuçlara ulaşılacaktır. Projemiz bu noktadan sonra bir diğer aşama olan PAR (Place and Routing) kısmına başarılı bir şekilde geçmiştir. Proje CPHA '0' VE CPHA '0' için doğru cevaplar verecektir. Girilen bit değeri arttırılarak örneklerimiz çeşitlendirilebilir ve geliştirilebilir.

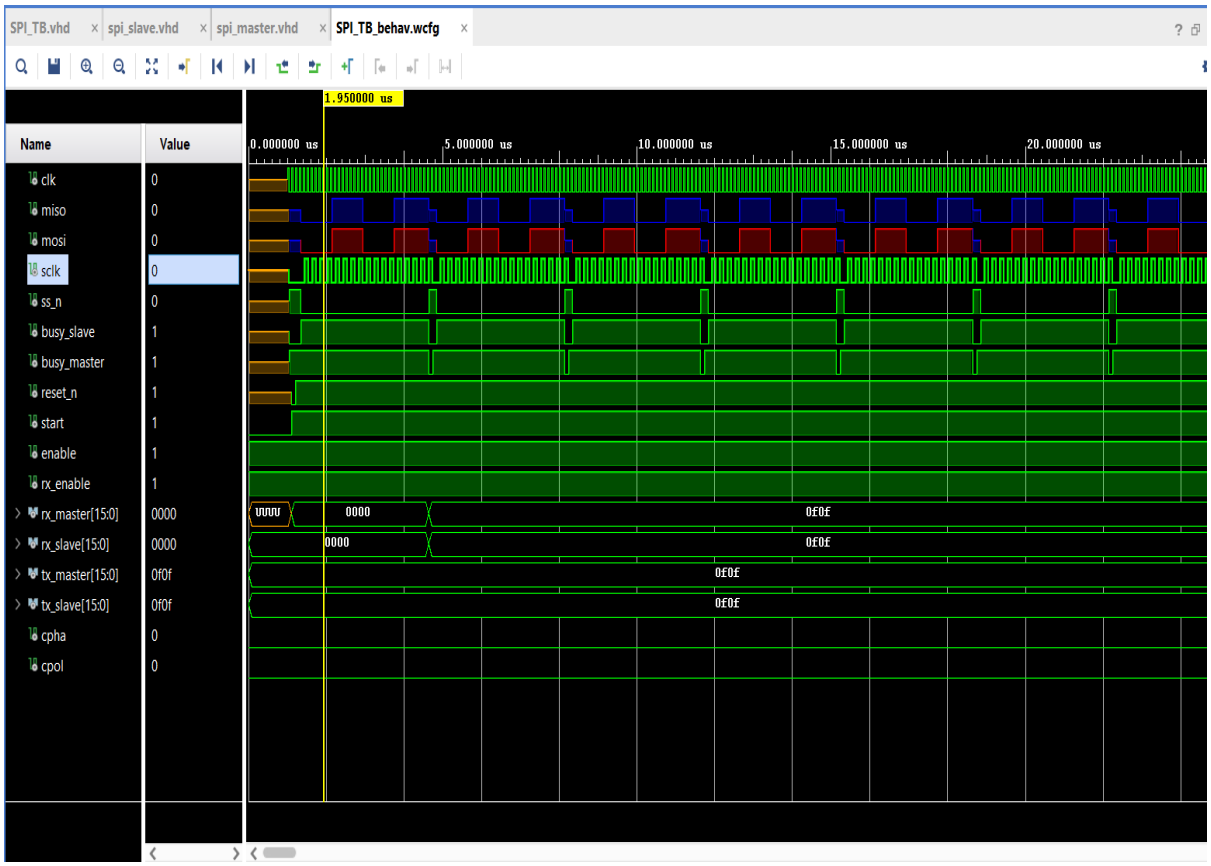
Şimdi MOSI hattından gelen veriyi değiştirip bir başka simülasyon sonucu aşağıda verilmiştir Bu sefer MOSI hat verisi olarak “1111000011110000” verisi için simülasyon sonucunu verilmiştir. Simülasyondan da görüldüğü gibi işlem başarıyla gerçekleşmiş, teorik olarak elde edilen verilerin doğruluğu ispatlanmıştır.



Şekil 4.3. Ana cihaz için “1111000011110000” bitinin MOSI hattından izlenimi

Yukarıda verilen MOSI hat verisi için sonuçlar gösterilmektedir. Verilerin doğruluğu bir kez daha ispatlanmış oldu.

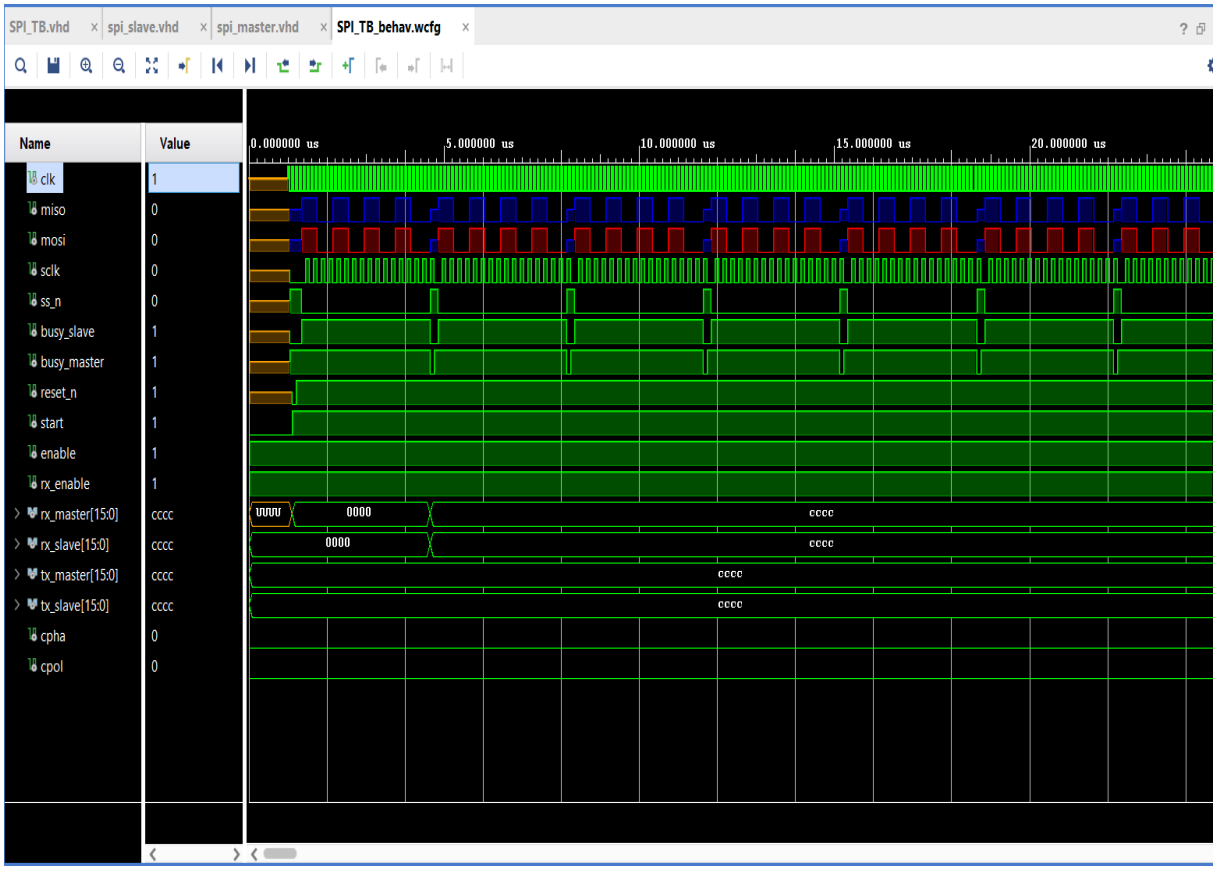
Şimdi MOSI hattından gelen veriyi değiştirip bir başka simülasyon sonucu aşağıda verilmiştir Bu sefer MOSI hat verisi olarak “0000111100001111” verisi için simülasyon sonucunu verilmiştir. Simülasyondan da görüldüğü gibi işlem başarıyla gerçekleşmiş, teorik olarak elde edilen verilerin doğruluğu ispatlanmıştır.



Şekil 4.4. Ana cihaz için “0000111100001111” bitinin MOSI hattından izlenimi

Yukarıda verilen MOSI hat verisi için sonuçlar gösterilmektedir. Verilerin doğruluğu bir kez daha ispatlanmış oldu.

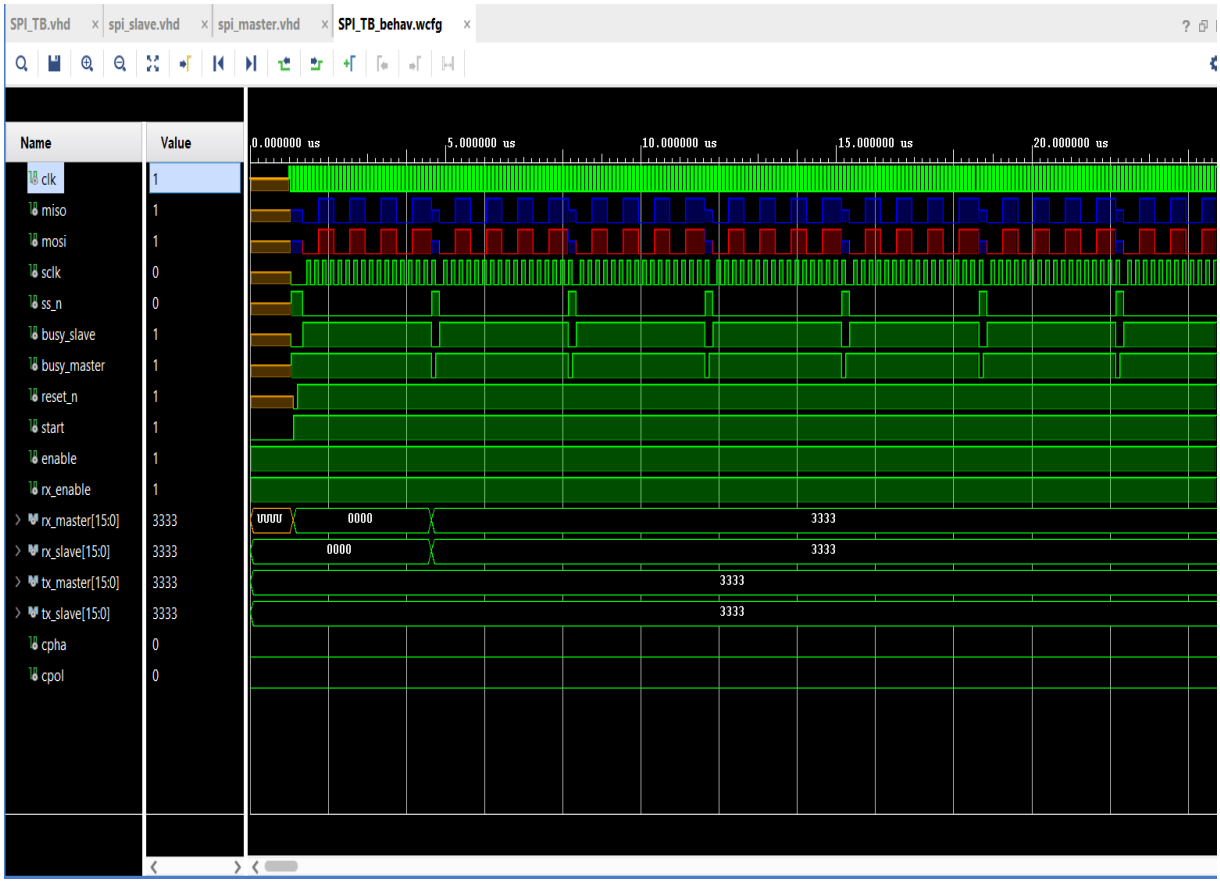
Şimdi MOSI hattından gelen veriyi değiştirip bir başka simülasyon sonucu aşağıda verilmiştir Bu sefer MOSI hat verisi olarak “1100110011001100” verisi için simülasyon sonucunu verilmiştir. Simülasyondan da görüldüğü gibi işlem başarıyla gerçekleşmiş, teorik olarak elde edilen verilerin doğruluğu ispatlanmıştır.



Şekil 4.5. Ana cihaz için “1100110011001100” bitinin MOSI hattından izlenimi

Yukarıda verilen MOSI hat verisi için sonuçlar gösterilmektedir. Verilerin doğruluğu bir kez daha ispatlanmış oldu.

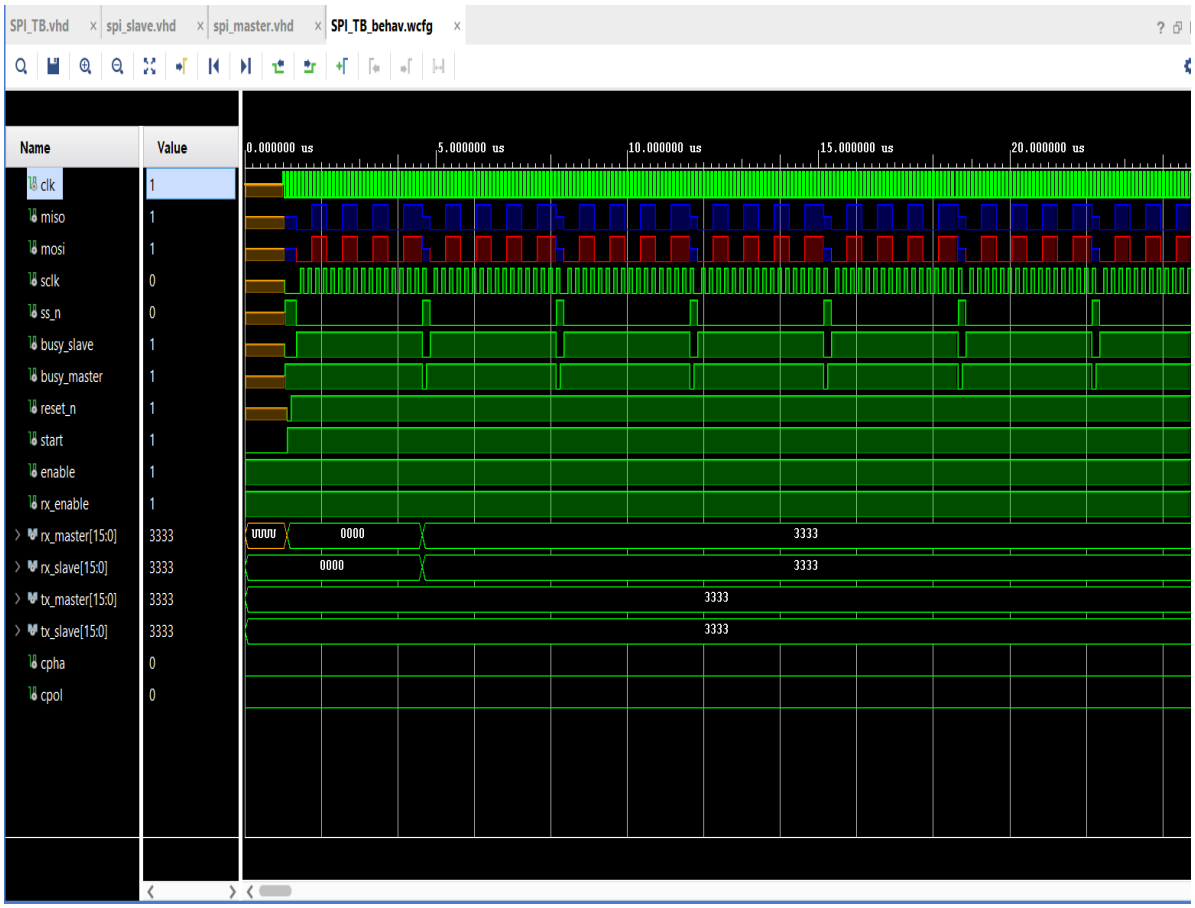
Şimdi MOSI hattından gelen veriyi değiştirip bir başka simülasyon sonucu aşağıda verilmiştir Bu sefer MOSI hat verisi olarak “0011001100110011” verisi için simülasyon sonucunu verilmiştir. Simülasyondan da görüldüğü gibi işlem başarıyla gerçekleşmiş, teorik olarak elde edilen verilerin doğruluğu ispatlanmıştır.



Şekil 4.6. Ana cihaz için “0011001100110011” bitinin MOSI hattından izlenimi

Yukarıda verilen MOSI hat verisi için sonuçlar gösterilmektedir. Verilerin doğruluğu bir kez daha ispatlanmış oldu.

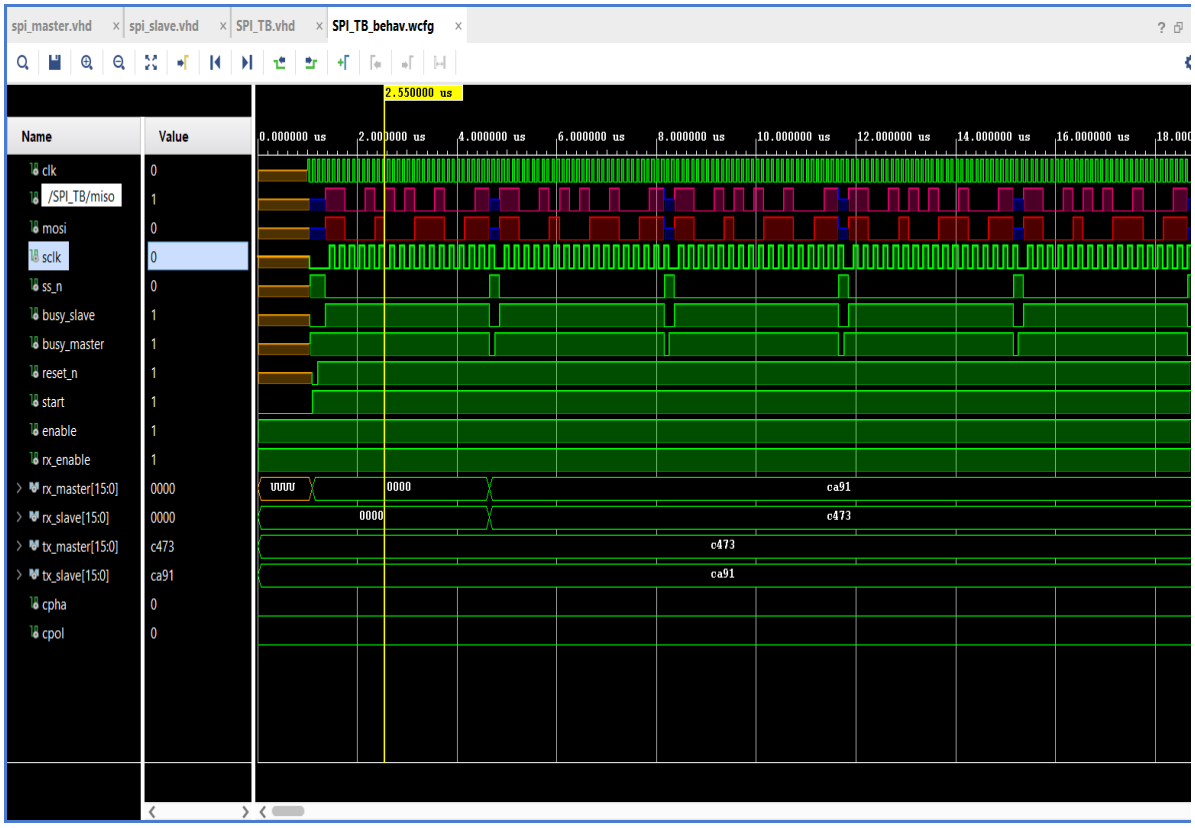
Şimdi de projemizde MISO hattından gelen veriyi kontrol etmek amacıyla simülasyon sonucunu içeren ekran görüntüsü aşağıda gösterilmiştir. Ekran görüntüsü, teorik olarak elde ettiğimiz bilgileri doğrular niteliktedir.



Şekil 4.7. Bağımlı cihaz için “0011001100110011” bitinin MISO gösterimi

Teorik olarak elde ettiğimiz veri ile simülasyon sonucu elde edilen MISO hat veri bitleri hatasız bir şekilde gönderilmiştir.

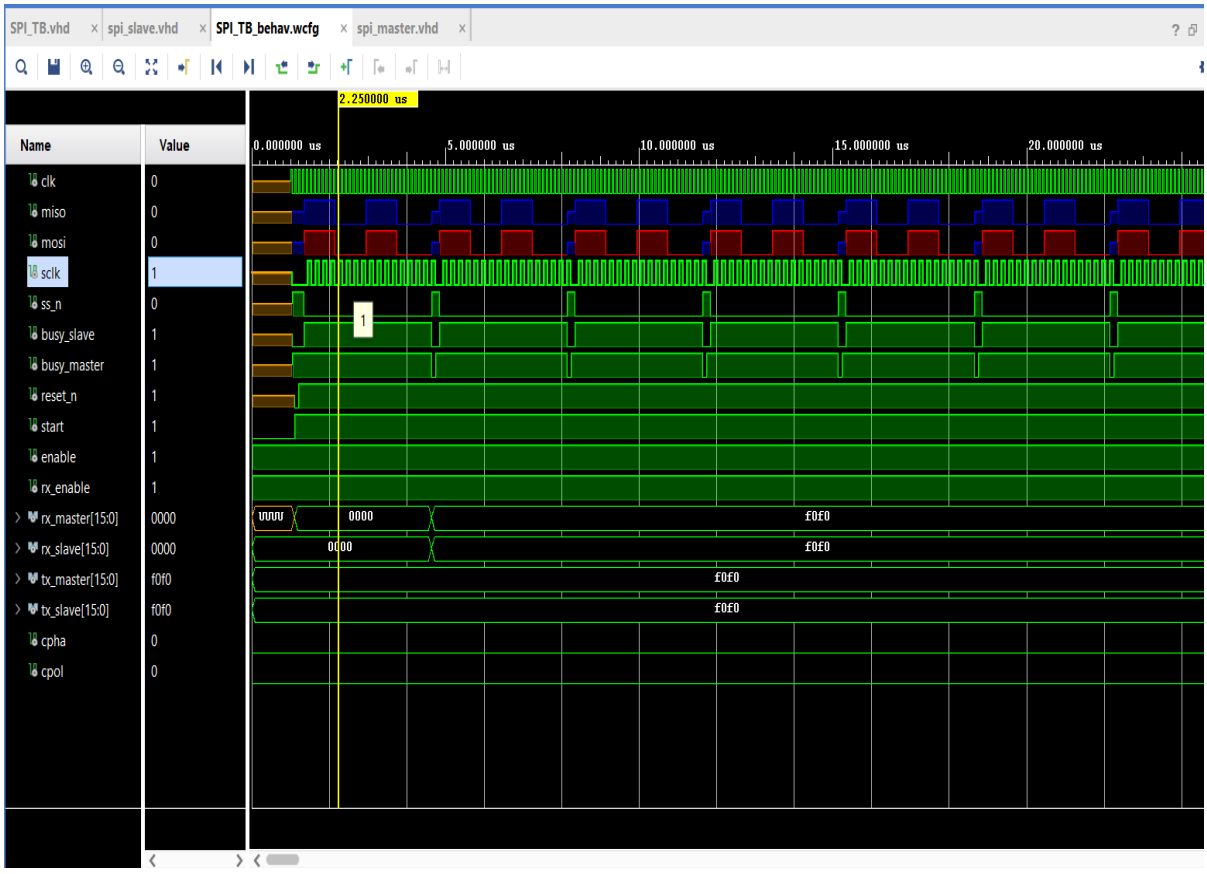
Şimdi de projemizde MISO hattından gelen veriyi kontrol etmek amacıyla simülasyon sonucunu içeren ekran görüntüsü aşağıda gösterilmiştir. Ekran görüntüsü, teorik olarak elde ettiğimiz bilgileri doğrular niteliktedir.



Şekil 4.8. Bağımlı cihaz için "1100101010010001" bitinin MISO hat gösterimi

Test ortamında kullandığımız 16 bitlik "1100101010010001" bitinin MISO hattından izlenimi paylaşılmıştır. Testimiz başarıyla gerçekleştirilmiştir. Burada dikkat edilmesi gereken MOSI ve MISO hattından alınan veya verilen verilerin aynı olmadığı görülmektedir. SPI tam çift yönlü olduğundan dolayı ana cihaz veriyi gönderirken aynı zamanda bağımlı cihazdan sürülen 16 bitlik veriyi göndermektedir. Bir SPI saat ile ana saat frekansı, girilen bit sayısına göre bir frekans bölümü gerçekleştirilmektedir. Bu işlem sonucunda istediğimiz saat frekansı elde edilecektir.

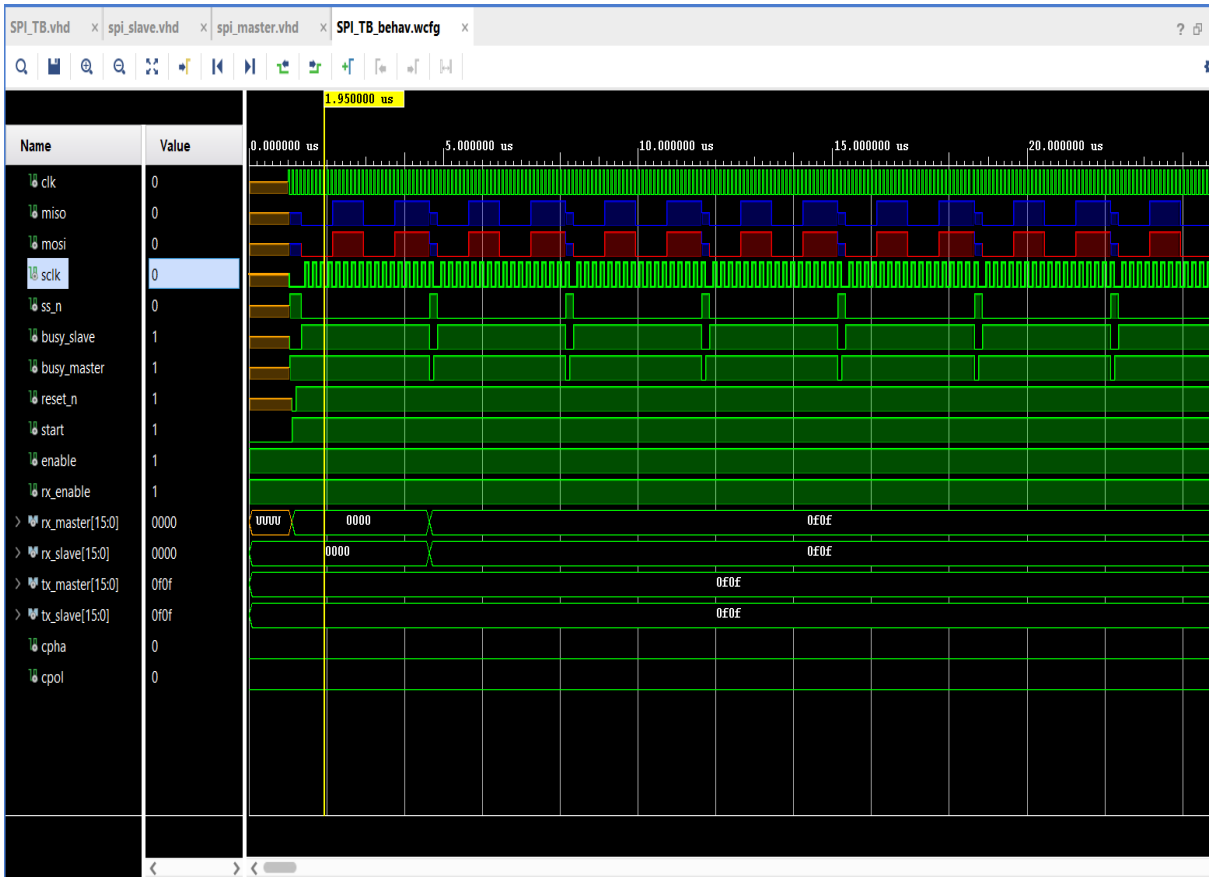
Şimdi de MISO hattından gönderilen veri ile ilgili bir simülasyon sonucu aşağıda verildiği gibidir. Bu kez MISO hattı için veri "1111000011110000" yani önceki örnekteki simülasyon sonucunda gösterildiği gibi yanı hem MOSI hemde MISO aynı veriyi alıp göndermiş olsun. Buna göre aşağıda gösterilen simülasyon sonucuna ulaşmış oluruz.



Şekil 4.9. Bağımlı cihaz için "1111000011110000" bitinin MISO hat gösterimi

Teorik olarak elde ettiğimiz veri ile simülasyon sonucu elde edilen MISO hat veri bitleri hatasız bir şekilde gönderilmiştir.

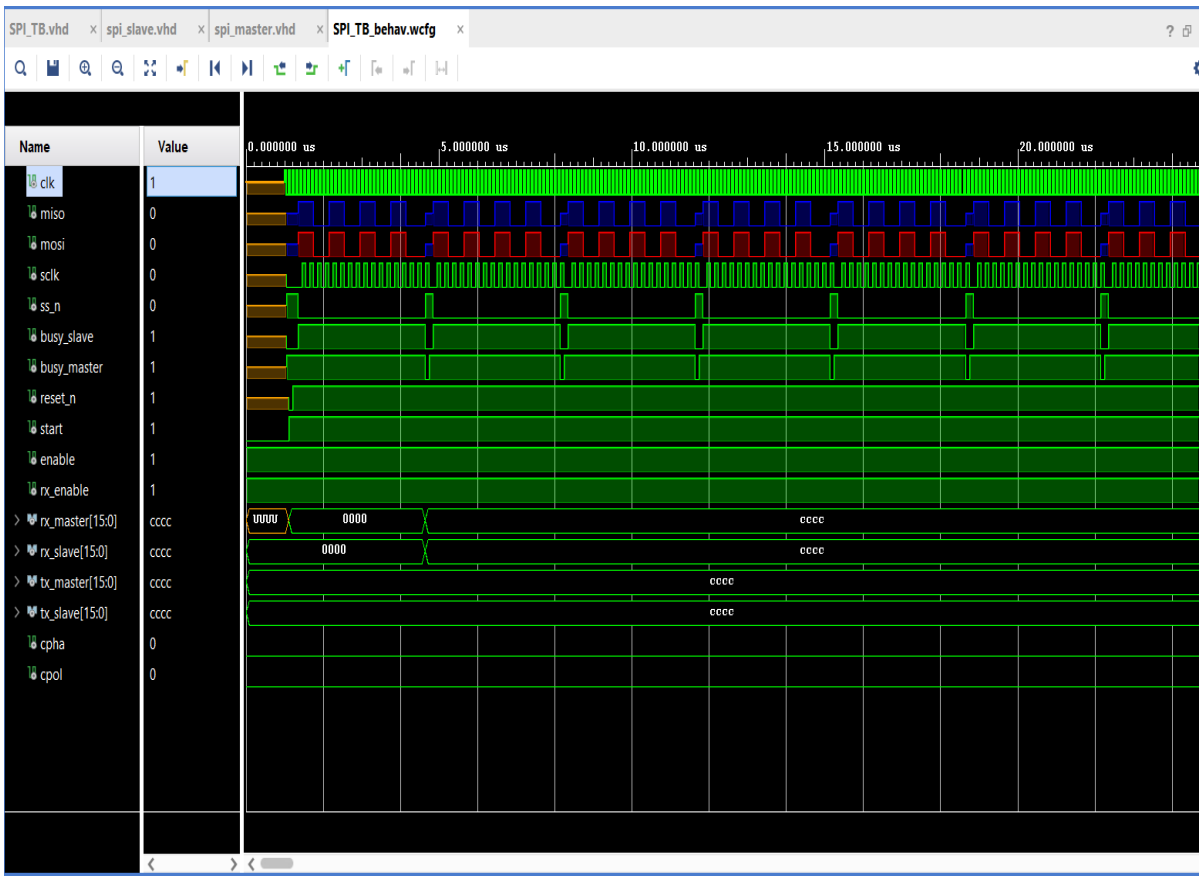
Şimdi de MISO hattından gönderilen bir başka veri ile ilgili bir simülasyon sonucu aşağıda verildiği gibidir. Bu kez MISO hattı için veri “0000111100001111” yani önceki örnekteki simülasyon sonucunda gösterildiği gibi yanı hem MOSI hemde MISO aynı veriyi alıp göndermiş olsun. Buna göre aşağıda gösterilen simülasyon sonucuna ulaşmış oluruz.



Şekil 4.10. Bağımlı cihaz için “0000111100001111” bitinin MISO hat gösterimi

Teorik olarak elde ettiğimiz veri ile simülasyon sonucu elde edilen MISO hat veri bitleri hatasız bir şekilde gönderilmiştir.

Şimdi de MISO hattından gönderilen bir başka veri ile ilgili bir simülasyon sonucu aşağıda verildiği gibidir. Bu kez MISO hattı için veri “0000111100001111” yani önceki örnekteki simülasyon sonucunda gösterildiği gibi yanı hem MOSI hemde MISO aynı veriyi alıp göndermiş olsun. Buna göre aşağıda gösterilen simülasyon sonucuna ulaşmış oluruz.



Şekil 4.11. Bağımlı cihaz için “1100110011001100” bitinin MISO hat gösterimi

Teorik olarak elde ettiğimiz veri ile simülasyon sonucu elde edilen MISO hat veri bitleri hatasız bir şekilde gönderilmiştir.

5. TARTIŞMA

Bu çalışma sonunda simülasyon çıktıları bize gösteriyor ki, teorik olarak elde ettiğimiz verilerle sonuçlar arasında bir uyum sağlanmıştır. Böylelikle bir sonraki aşamalar olan plan ve yerleştirme aşamasına geçimi sağlıklı bir şekilde gerçekleştirilecektir. UART ve SPI günümüzde en çok kullanılan protokollerinden bazılarıdır ve bunu projede kullanmak hem hız hem de enerji verimliliği konusunda bize önemli avantaj sağlayacaktır. Bunda sonraki aşamalarda UART ve SPI kullanan sensörler ve modüller ile birlikte kullanılarak proje geliştirilebilir. FPGA'in günümüzde kullanımı gittikçe önem kazanması bundan dolayıdır. Uzay ve savunma sanayide kullanımı ile birlikte bu projelerin FPGA ile kullanımı daha önemli bir hal almıştır. Haberleşmedeki dijital modülasyonların neredeyse FPGA ile kullanılması bu fikri doğrular niteliktedir. Enerji verimliliğinin özellikle açık ara en önemli olduğu alan olan uzay alanında ilerlemeler diğerlerinden ayrışmıştır.

Bir diğer faktör olan yüksek frekans gereksinimi FPGA'yi daha cazip kılmıştır. Günümüzde işlemcilerin gigahertz seviyelerinde olması haberleşmede frekans ölçeğinin yükseltilmesi FPGA kullanımını yaygınlaştırmıştır.

Bu projenin daha ileri adımları birden fazla mikrodenetleyicinin, sensörün, modülün, hafıza elemanlarının ve bunların arasındaki veri bit akışını inceleyen tüm kompleks işlemlerde kullanılması bu alanda çalışmanın önemini ortaya koymaktadır. Gömülü sistemin bu ileri düzey konularında çalışmak günümüzdeki çalışmaların ne kadar ileri bir seviyeye geldiğinin kanıtıdır.

Projemizin diğer çalışmalar ile benzerlikleri bulunmaktadır. FPGA'de SPI haberleşme sadece VHDL programlama dili ile değil, aynı zamanda Vivado ile birlikte gelen özelleşmiş bloklardan oluşan bir IP (İntellectual Property) hazır bloklarla proje geliştirilebilir. Bir diğer ileri aşama FPGA ile bir başka mikrodenetleyici ile haberleşmesi Microblaze işlemci kullanılarak haberleşmenin yapılması günümüzde en yaygın projelerin temelini oluşturur.

Bir SPI modülü kullanılarak yine SPI temelli sensörler ile modüller kullanılarak, örneğin anten modülü ile haberleşmenin büyük önemi vardır. Projenin ilerisi için bu konuları temel alan projeler baz alınacaktır. Projemiz FPGA kullanımı ile hem hız hem de enerji verimliliği konusunda diğer projelerden pozitif olarak ayrışmaktadır. Simülasyon yani diğer adı benzetim olan ortamda bir sorunla karşılaşmamamız sonucu olarak diğer aşamalara geçimine de başarıyla adım atılacaktır. Benzetim ortamında saat gecikmelerimizi göremiyoruz daha gerçekçi bir model oluşturmak veya daha sağlıklı sonuçlar için VHDL bekleme komutları eklenerek daha başarılı bir model oluşturulabilir.

6. KAYNAKÇA

BAŞAK, S., Tarih yok, “FPGA ile Gömülü Sistem Tasarımına Giriş”

ÇAVUŞLU., A., Ç., 2017, “VHDL ile SAYISAL TASARIM ve FPGA uygulamaları” Kodlab Yayınları, İSTANBUL

KELLECİ, B., 2017, “VHDL ve VERİLOG ile sayısal tasarım” Seçkin Yayıncılık, ANKARA.

SPI Block Guide V03.06 (Motorola Inc,21 Ocak 2000)

UZUN, S., CANAL, M., R., KAÇAR, M., C., “VHDL Programlama Dili ve Sayısal Elektronik Devrelerin FPGA Tabanlı Uygulaması. International Advanced Technologies Symposium (2011)

URL 1. https://resources.pcb.cadence.com/blog/2020-hardware-description-languages-vhdl-vs-verilog-and-their-functional-uses (2021)

URL2.https://www.kaizen40.com/senkron-ve-asenkron-iletisim nedir/#:~:text=Senkron%20ve%20Asenkron%20İletişim%20olarak,için%20sürekli%20olarak%20karakterleri%20gönderir. (2021)

URL3.https://www.circuitbasics.com/basics-of-the-spi-communication-protocol/#:~:text=SPI%20is%20a%20common%20communication,can%20be%20transferred%20without%20interruption. (2021)

URL4.https://microcontrollerslab.com/uart-communication-working-applications/ (2021)

URL5.https://microcontrollerslab.com/introduction-to-spi-communication-protocol/ (2021)

URL 6. https://aticleworld.com/spi-communication-protocol/ (2021)

URL7. https://www.mcu-turkey.com/wp-content/uploads/2012/02/VHDL-Tutorial.pdf (2021)

URL8. https://en.wikipedia.org/wiki/Serial_Peripheral_Interface (2015)

7. TEŞEKKÜR

Bu çalışmada teorik haberleşme ve FPGA teknolojilerine ilgi duymamı sağlayan, ileride bu alanda çalışmaya devam etme kararımı vermeme vesile olan hocam Dr. Öğr. Üyesi Abdurrahman Günday öğretmenime,

Mantık devreleri dersini öğrenmemi sağlayan öğretmenim Öğr. Gör. Dr. Gökhan Yenikaya'ya,

FPGA kartını ve teknolojileri hakkında bilgi aldığım Digilent ve Xilinx firmalarına,

Çok teşekkür ederim.

8.ÖZGEÇMİŞ

8.1 Ahmed Melih Ulusoy

1995 yılında Malatya'da dünyaya gelmiştir. İlk ve orta öğrenimini Malatya Mehmet Emin Bitlis İlköğretim okulunda tamamlamıştır. Liseyi Kernek Anadolu Lisesinde okumuştur. 2015 yılında Uludağ Üniversitesi Elektrik Elektronik Mühendisliği kazanmıştır. Şu an Uludağ Üniversitesi öğrencisi olarak aktif eğitim hayatına devam etmektedir.