Düşük Maliyetli FPGA'lar için Hafifsıklet SPI Master IP Tasarımı ve Gerçekleştirilmesi

Design and Implementation of a Lightweight SPI Master IP for Low Cost FPGAs

Yazarlar Gizlenmiştir

***Özetçe*—SPI (Serial Peripheral Interface) protokolü 1980'lerde geliştirilerek özellikle gömülü sistem uygulamalarında IC'lerle haberleşmede kendisine geniş bir kullanım alanı bulmuştur. SPI protokolünün FPGA'da tasarımı ve gerçekleştirilmesi çok ağır bir tasarım zamanı getirmemekle birlikte, özellikle düşük maliyetli ve düşük kaynaklara sahip FPGA'lar için daha az lojik kaynağı harcayan SPI tasarımları kritik olmaktadır. Bu makalede tasarlanan SPI master IP modülü lojik kaynak kullanımı açısından hem literatürdeki daha önceki SPI tasarımlarıyla hem de açık kaynak kod olarak internetten elde edilen SPI protokol tasarımlarıyla karşılaştırılmıştır. Yapılan çalışma, diğer tasarımlardan daha az lojik kaynağı harcamakla birlikte modülerlik ve fonksiyonalite olarak da SPI protokolünün bütün gereksinimlerini sağlamaktadır. Tasarlanan modül, açık-kaynak olarak opencores komünitesinde herkesin kullanımı ve değerlendirmesine açık olarak yüklenmiştir.**

Anahtar Kelimeler — düşük maliyetli FPGA; hafifsıklet FPGA tasarımı; SPI protokolü, açık-kaynak.

***Abstract*—SPI protocol was developed in 1980s and it is used vastly to communicate ICs especially in embedded systems applications. Although designing and implementing SPI protocol on FPGA does not require a long design time, SPI designs using less resources are essential for low cost FPGAs having small amount of logic resources. The SPI master IP designed in this paper compared both to the previous works in the literature and open source implementations of SPI IPs that can be found on the internet. The proposed work utilizes less amount of logic resource than other designs while keeping modularity and functionality and satisfy all the requirements of SPI protocol. The designed module is uploaded to opencores community for public usage and evaluation.**

Keywords — low-cost FPGA; lightweight FPGA design; SPI protocol, open-source.

I. GİRİŞ

SPI (Serial Peripheral Interface) haberleşme protokolü, ilk olarak 1979 yılında Motorola'nın 68000 mikroişlemcisiyle birlikte piyasaya çıkmıştır ve o günden bugüne kadar özellikle yongalar arasında haberleşme amacıyla kullanılmaktadır [1]. Günümüzde birçok ADC, DAC ve çeşitli sensör yongaları ile SPI protokolüyle haberleşilmektedir. SPI spesifikasyonunun haklarına resmi olarak Motorola'yı satın almış olan Freescale firması sahip olmakla birlikte yaklaşık 40 yıldır yoğun bir kullanım alanına sahip olmasından dolayı *de facto* standart olarak yorumlanmaktadır [2].

SPI protokolü master-slave ilişkisine göre tasarlanmıştır. Tek bir master birden çok slave ile haberleşebilmektedir. Haberleşme çift-yönlü ve seri şekilde gerçekleşmektedir. Elektronik sistemlerde kullanılan pek çok ADC, DAC ve diğer yongalar SPI slave olarak tasarlanmışlardır. Bu yongalarla veri iletişimi sağlamak için SPI master sürücüsüne sahip olan mikrodenetleyici, FPGA vb gibi bir akıllı birime ihtiyaç duyulmaktadır. Mikrodenetleyicilerde bir ya da birden fazla SPI master sürücü donanımı bulunmaktadır ve kullanıcı, bu sürücüye ilgili bilgileri yazılımda girerek SPI slave yongayla haberleşebilmektedir. Yonga üreticileri genellikle SPI slave arayüzüne sahip ürünlerinin yazılım platformları için sürücülerini (.h ve .c dosyaları) ücretsiz ve açık kaynak kod olarak paylaşmaktadırlar. Mikrodenetleyici veya mikroişlemci ile SPI slave yonga ile haberleşecek olan kullanıcının bu sürücüleri projesine ekleyip kullanması yeterli olmaktadır.

FPGA (field programmable gate array) yongaları, 1980'li yılların ortalarında piyasaya girdikten sonra özellikle CPU, GPU gibi yongaların tasarım aşamasında silikona dökümden önce prototip amacıyla, ASIC'e alternatif olarak, uzay ve savunma sanayiinde, haberleşme sistemlerinde ve tüketici elektroniği gibi alanlarda kendisine yer bulmuştur. Mikrodenetleyicilerden farklı olarak HDL (hardware description language) dilleriyle tasarımı gerçekleştirilmektedir. Günümüzde yüksek sığaya sahip FPGA'ların yanında düşük sığaya sahip FPGA'lar da hala düşük maliyetli sistemlerde kullanılmaktadır [3,4]. Bu tarz sistemlerde maliyet ön plana çıktığından dolayı gerçekleştirilecek olan fonksiyonların az lojik kaynağı tüketmesi önem arz etmektedir.

Bu çalışmada, düşük kaynak tüketimine yönelik bir hafifsıklet (lightweight) SPI master IP modülü (LW SPI) VHDL dilinde tasarlanmış ve simülasyon ortamında test

edilmiştir. Modülün lojik kaynak tüketimi, geçmiş çalışmalarla ve internet ortamında açık kaynak kod olarak bulunan diğer SPI IP modüllerinin kaynak tüketimiyle karşılaştırılmıştır. 2. bölümde SPI protokolünün detayları anlatılmış, 3. bölümde de FPGA'da LW SPI Master IP tasarımı açıklanmıştır. 4. bölümde simülasyon sonuçlarıyla beraber tasarlanan LW SPI Master IP ile geçmiş çalışmalardaki ve açık kaynak kod olarak bulunan modüllerin FPGA lojik kaynak tüketimleri karşılaştırılmıştır.

II. SPI HABERLEŞME PROTOKOLÜ

A. *Spi mimarisi*

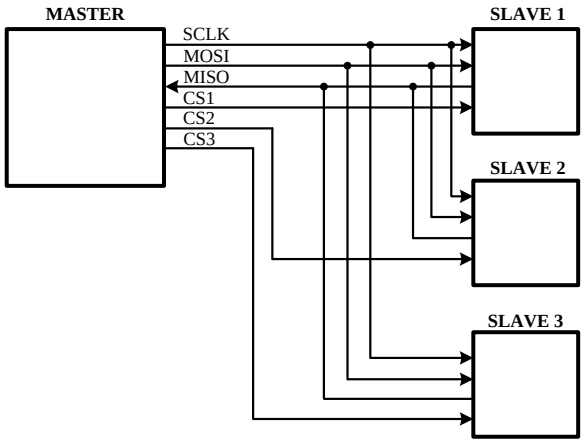
SPI protokolü, bir adet master cihazın, bir ya da daha fazla slave cihazla full-duplex olarak haberleşmesini sağlayan seri ve senkron bir protokoldür [1]. SPI protokolünde 4 adet pin mevcuttur: CS veya SS (chip/slave select), SCLK (serial clock), MOSI (master out slave in), MISO (master in slave out). Düşük performanslı bazı slave yongalar 3 pinli SPI protokolünde half-duplex olarak haberleşmektedir fakat günümüzde kullanılan SPI protokolüne sahip yongalar istisnalar dışında 4 pinle haberleşmektedir ve 3 pin destekleyen spi master tasarımı pek bulunmamaktadır. 3 pin SPI protokolünde MOSI ve MISO pinleri tek bir giriş-çıkış özelliğine sahip pinde ortaklanmıştır [2][4].

Şekil. 1'de, master ve 3 adet slave yongadan oluşan bir SPI bağlantı mimarisi gösterilmiştir.

B. *Spi sinyalleri*

CS, MOSI ve SCLK pinleri master cihaz tarafından sürülürken MISO pini slave cihaz tarafından sürülmektedir. CS pini, haberleşilmek istenilen slave cihazını aktif hale getirir. Master cihaz veri aktarımı sırasında SCLK ve MOSI pinlerini sürmeye başlar. Slave cihaz da SCLK pinine bağlı olarak master cihazına MISO hattı üzerinden veri aktarımını gerçekleştirir. Master cihaz, MISO hattında aynı anda tek bir slave cihazdan veri geldiğini garantilemek için aynı anda yalnızca tek bir slave cihazın CS pinini düşük seviyede tutmak zorundadır.

SCLK hattının saat polarizasyon (CPOL) ve saat faz (CPHA) bilgisi tanımlanmasına göre master ve slave tarafından verinin örneklenmesi ile ilgili dört farklı konfigürasyon tanımlanmıştır. Tablo 1'de CPOL ve CPHA bitlerine göre SPI modları verilmiştir.



Şekil. 1: Çoklu slave yapısında SPI haberleşme mimarisi

TABLO 1: SPI MOD KONFİGÜRASYONLARI

SPI MOD	CPOL	CPHA
Mod 0	0	0
Mod 1	0	1
Mod 2	1	0
Mod 3	1	1

CPOL biti, SCLK hattının aktif-yüksek ya da aktif-düşük çalışmasını ayarlar. CPOL '0' olduğu zaman hatta bir haberleşme gerçekleşmiyorsa SCLK pini düşük seviyede bulunur; '1' olduğu zaman ise aktif-yüksek olarak çalışır ve boş durumda yüksek seviyede bekler.

Saat faz biti ise ilk SCLK değişikliğinde nasıl davranılacağını tanımlar. Eğer CPHA biti '0' ise, SCLK hattındaki ilk değişiklikte hem master ham de slave tarafından verinin örneklenmesi gerçekleşir. CPHA biti '1' olarak ayarlandıysa SCLK hattındaki ilk değişiklikte veri hazır hale getirilir ve bir sonraki değişiklikte veri örneklenir. Şekil. 2'de kırmızı noktalı çizgiler CPHA biti '0' iken örnekleme notkalarını, mavi noktalı çizgiler de CPHA biti '1' olduğu zamanki örnekleme noktalarını göstermektedir.

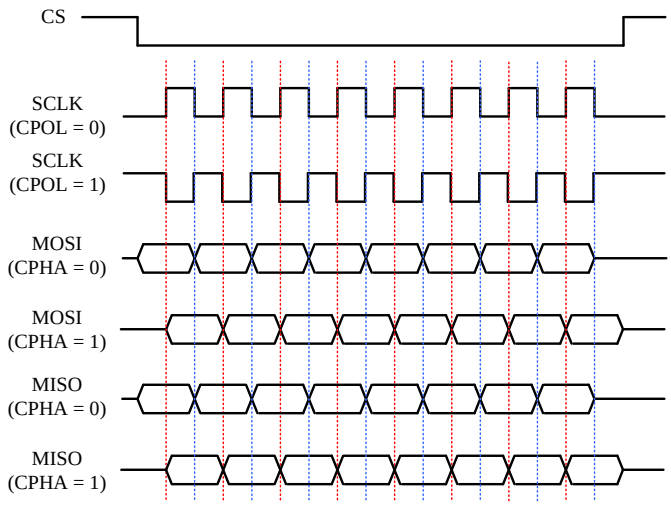
Veri transferi başlamadan önce master ve slave arasında CPOL ve CPHA bilgilerinin aynı olması gerekmektedir.

SPI protokolünde veri iletişimi bir ya da daha fazla bayt içerecek şekilde olabilirken en anlamlı bit önce iletilecek şekilde transfer gerçekleştirilir. Birden fazla bayt veri transferi yapılacağı zaman CS pini düşük seviyede tutularak slave cihazı aktif halde tutulur ve diğer hatlar aynı şekilde çalışmaya devam eder.

III. LW SPI MASTER IP TASARIM DETAYLARI

A. *Kullanıcı arayüzü I/O sinyalleri*

LW SPI Master IP tasarlanırken FPGA lojik kaynaklarını az harcaması hedeflenirken, kullanım kolaylığı da ön planda tutulmuştur.



Şekil. 2: Farklı CPOL ve CPHA için SPI sinyalleri

Kullanım kolaylığını arttırmak için literatürde mevcut bulunan IP'lerden çok daha az sayıda ve sadece kullanılması zorunlu giriş-çıkış sinyalleri tanımlanmıştır. SPI protokolünün CS, SCLK, MOSI ve MISO sinyalleri dışında IP'yi kontrol etmek için yalnızca EN (enable), DATA_READY, MOSI_DATA ve MISO_DATA olarak 4 adet giriş-çıkış sinyalleri tanımlanmıştır. Tablo 2'de kontrol sinyalleri ve açıklamaları verilmiştir.

TABLO 2: LW SPI MASTER IP KULLANICI ARAYÜZ SİNYALLERİ

mosi_data_i (7-0)	SPI slave cihazına iletilmek istenilen veri
miso_data_o (7-0)	SPI slave cihazından okunan veri
en_i	İletişimi başlatan kontrol sinyali
data_ready_o	İletişimin tamamlandığını bildirir

SPI protokolü haberleşmesinde yalnızca masterın veri gönderdiği, yalnızca slave cihazın veri gönderdiği, ya da her iki cihazın da aynı anda veri gönderdiği durumlar olmak üzere 3 farklı türde haberleşme şekli gerçekleşebilir. Tasarlanan IP tek bir kontrol sinyali ile üç durumu da desteklemektedir. SPI master IP kullanıcısı, transfer işlemini başlatmak için slave cihazına iletmek istediği 8-bit veriyi *mosi_data_i* giriş sinyaline kaydedip *en_i* kontrol sinyaline '1' yazmalıdır. Slave cihazının göndermiş olduğu veri, *data_ready_o* çıkış sinyali '1' olduğu zaman *miso_data_o* çıkış sinyali okunarak elde edilir. Eğer birden fazla bayt veri transferi gerçekleştirilmek istendiği zaman *data_ready_o* sinyali '1' olduğunda *en_i* giriş sinyali '1' yapılarak veri transferine devam edilebilir. Kullanıcı veri transferini sonlandırmak istediği zaman *en_i* sinyalini '0' seviyesine çekmesi yeterli olacaktır.

Kullanıcının sentezleme öncesinde atama yapabileceği dört adet *generic* parametre tanımlanarak da kullanım kolaylığı ve modülerite sağlanmaya çalışılmıştır. Kullanıcı, IP saat hızını, SPI hattı SCLK saat hızını ve SPI hattı saatinin polarite ve faz bilgilerini sırasıyla *c_clkfreq*, *c_sclkfreq*, *c_cpol* ve *c_cpha* parametrelerine girerek sentezleme öncesinde gerekli ayarlamaları yapabilmektedir. Ayrıca LW SPI master modülünde herhangi bir firmaya ait IP kullanılmamış ve sadece VHDL dili ve sentaksı kullanılarak tasarım gerçekleştirilerek FPGA üreticilerden bağımsız bir IP geliştirilmiştir.

B. *Tasarım detayları*

LW SPI master modülünde 4 adet process yapısı kullanılmıştır. Bunlardan iki tanesi combinational olarak çalışırken (SAMPLE_EN, RISEFALL_DETECT) iki tanesi de sequential devre olarak çalışmaktadır (MAIN, SCLK_GEN). Modülde 2 adet durum tanımlanmıştır: IDLE ve TRANSFER. IDLE durumundayken *en_i* sinyalinin '1' olmasıyla birlikte TRANSFER durumuna geçilir ve veri transferi tamamlandıktan sonra tekrar IDLE durumuna geçiş yapılır.

TRANSFER durumunda *sclk_o* sinyalinin üretimi için generic olarak tanımlanan clkfreq ve sclkfreq parametrelerinden (1)'e göre edgecntr sinyali için sınır tanımlanmış olunur.

$$c\_edgecntrlimdiv2 = c\_clkfreq / (c\_sclkfreq * 2) \quad (1)$$

IDLE durumundayken *en_i* sinyali '1' olup TRANSFER durumuna geçildiğinde *sclk_en* sinyali '1' olur ve *sclk* sinyali, sclkfreq parametresi ile tanımlanan frekans bilgisine göre edgecntr isimli sayaç sinyali aracılığıyla SCLK_GEN process bloğu içerisinde üretilmiş olunur.

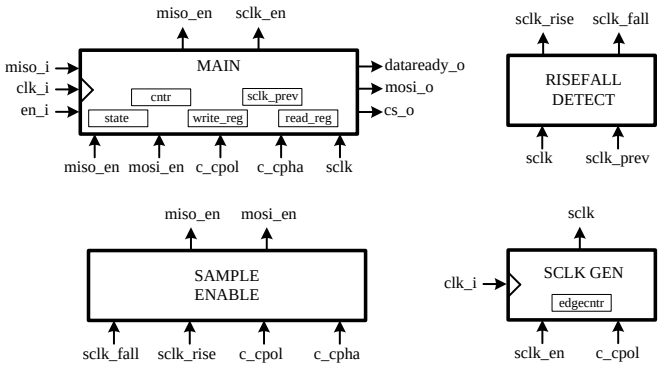
sclk_prev isimli bir sinyale her saat vuruşunda *sclk* değeri atanır. Bu iki flip-flop kontrol edilerek sclk sinyalinde yükselen ya da düşen kenar tespit edilen lojik RISEFALL_DETECT isimli combinational process yapısı içerisinde kodlanmıştır.

pol_phase sinyaline generic parametre olan cpol ve cpha sinyalleri atanmıştır. pol_phase sinyalinin aldığı değere göre master cihazın mosi_o çıkışında veriyi ne zaman değiştireceği ve miso_i girişini ne zaman örnekleyeceği mosi_en, miso_en isimli iki sinyalle SAMPLE_EN combinational process bloğunda tanımlanmıştır.

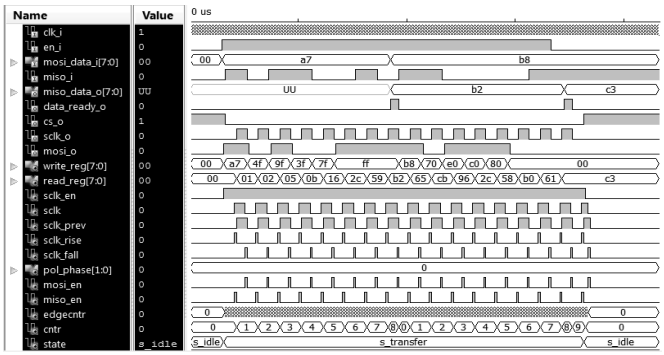
MAIN process bloğu içerisinde IDLE ve TRANSMIT durumları gerçekleştirilmiştir. TRANSMIT durumunda cpha generic parametresinin aldığı değere göre iki farklı tasarım yapısı tanımlanmıştır. *cntr* isimli bir sayaç bit sayısını sayarken SAMPLE_EN process bloğu çıktısı olan *mosi_en* ve *miso_en* sinyallerinin değerlerine göre *mosi_o* çıkışına basılacak olan veri ve *miso_i* girişindeki değerin örneklenmesi gerçekleştirilir. Okunan veri hazır olduğunda *data_ready_o* sinyali '1' olur. Son *sclk_o* değişimi gerçekleştiğinde *en_i* sinyali '1' ise *cntr* sıfırlanır ve bir bayt daha veri transferi devam eder, aksi takdirde IDLE durumuna geçilir. LW SPI master IP'nin process yapıları Şekil. 3'de giriş-çıkış sinyalleriyle birlikte gösterilmiştir.

IV. SONUÇLAR

LW SPI master IP tasarımı Xilinx ISE 14.1 yazılımında synthesis ve implementation aşamalarıyla gerçekleştirilmiş ve herhangi bir hata ya da uyarı almadan sonuçlanmıştır. FPGA olarak da Spartan 6 ailesinin en düşük hacimli ürünü olan XC6SLX4 seçilmiştir [13]. Bu FPGA'nın seçilme nedeni, düşük maliyetli olmasıdır. Karşılaştırılacak olan diğer tasarımlar da aynı FPGA üzerinde ve aynı synthesis ve implementation ayarlarıyla derlenmiştir. Simülasyon sonuçları da yine ISE'de mevcut olan ISIM'de gerçekleştirilmiştir. Şekil. 4'de verilen simülasyon sonuçlarında slave cihaza 0xA7 ve 0xB8 verileri yazılırken 0xB2 ve 0xC3 verileri okunmuştur. clkfreq parametresi 50 MHz, sclkfreq parametresi 5 MHz ve cpol ve cpha parametreleri '0' olarak girilmiştir.



Şekil. 3: LW SPI master IP process blokları

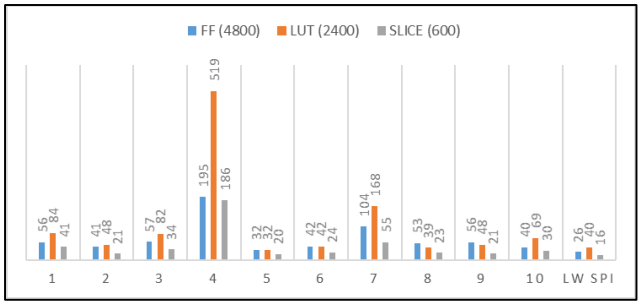


Şekil. 4: Simülasyon sonuçları

SPI protokolünün FPGA üzerinde tasarımı ve gerçekleştirilmesine dair pek çok çalışma bulunmaktadır [5, 6, 9, 10, 11, 14]. Bu çalışmalardan FPGA kaynak tüketimi verenler ve internette açık-kaynak kod olarak bulunan SPI master tasarımları ile kaynak tüketimi karşılaştırılması yapılmıştır. Oudjida ve Anand yaptıkları çalışmalarda Virtex5 FPGA ailesini kullandıkları için onların sonuçlarıyla karşılaştırmak adına LW SPI master IP modülü de Virtex5'te de sentezlenmiştir. Oudjida'nın çalışmasında 232, Anand'ın çalışmasında 31 adet slice kullanılmışken LW SPI master IP 15 adet slice kullanmıştır [8, 11]. Saha'nın çalışmasında 4 girişli LUT bulunan Spartan3 FPGA ailesi ile 34 adet slice tüketmiştir. LW SPI master IP modülü bu ailede sentezlendiğinde 32 adet slice kullanmaktadır.

İnternet ortamında açık-kaynak SPI protokolü tasarımları araştırılıp, bulunan modüller Spartan6 ailesinde sentezlenmiştir. Değerlendirilmeye alınan tasarımlarda SPI protokolünü eksiksiz gerçekleştirebilmeleri ve SCLK saat hızını, CPOL ve CPHA konfigürasyon bitlerini parametrik olarak ele almalarına önem verilmiştir. İlk 6 adet SPI master modülü opencores açık-kaynak komünitesinden alınmıştır [2]. Tasarımlar opencores'da sırasıyla *AnotherSPIController, asynchronous_master_spi, simple_spi, spi, spi_master_slave ve tiny_spi* olarak kayıtlıdır. Diğer tasarımlar da internette açık olarak bulunabilen SPI master modüllerinden elde edilmiştir [15-18].

Sonuçlar değerlendirildiğinde LW SPI master IP modülünün en az sayıda flip-flop ve slice kaynaklarından tükettiği, minimal LUT tüketiminde de ikinci sırada olduğu gözlenmiştir. İmplementasyon sırasında slice tüketimini azaltmak için *PlanAhead* yazılımında 10 adet slice *Pblock* kısıtı seçilerek LW SPI master IP yeniden implement edildiğinde 10 adet slice içerisine de sığabildiği gözlenmiştir.



Şekil 5: SPI master modülleri FPGA kaynak tüketimleri

LW SPI, Spartan6 ailesinin en düşük kapasiteli XC6SLX4 cihazında yalnızca 10 adet slice tüketimiyle toplam slice kapasitesinin sadece %1.66 kadarını kullanmıştır.

V. DEĞERLENDİRME

Bu çalışmada ucuz ve düşük kapasiteli FPGA'lar için az kaynak tüketen ve kullanımı kolay bir hafifsıklet SPI master IP VHDL dilinde tasarlanmış ve geçmiş tasarımlarla kaynak tüketimi karşılaştırılarak avantajı gösterilmiştir. Yapılan çalışma **https://opencores.org/GİZLENMİŞTİR** adresinde açık kaynak olarak yayınlanarak tasarımcı ve araştırmacıların kullanımına ve değerlendirmesine açılmıştır.

KAYNAKLAR

[1] Motorola Inc., "SPI Block Guide V03.06," February 2003.

[2] Leens. F, "An Introduction to I2C and SPI Protocols" *IEEE Instrumentation & Measurement Magazine*, pp. 8-13, February 2009.

[3] https://opencores.org/

[4] http://www.ti.com/lit/ds/symlink/lm74.pdf

[5] D. N. Oruganti and S. S. Yellampalli, "Design of power efficient SPI interface," *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, New Delhi, 2014, pp. 2602-2606.

[6] Tianxiang Liu and Yunfeng Wang, "IP design of universal multiple devices SPI interface," *2011 IEEE International Conference on Anti-Counterfeiting, Security and Identification*, Xiamen, 2011, pp. 169-172.

[7] A. K. Oudjida, M. L. Berrandjia, A. Liacha, R. Tiar, K. Tahraoui and Y. N. Alhoumays, "Design and test of general-purpose SPI Master/Slave IPs on OPB bus," *2010 7th International Multi- Conference on Systems, Signals and Devices*, Amman, 2010, pp. 1-6.

[8] Anand N, G. Joseph, S. S. Oommen and R. Dhanabal, "Design and implementation of a high speed Serial Peripheral Interface," *2014 International Conference on Advances in Electrical Engineering (ICAEE)*, Vellore, 2014, pp. 1-3.

[9] T. Praveen Blessington, B. Bhanu Murthy, G. V. Ganesh and T. S. R. Prasad, "Optimal implementation of UART-SPI Interface in SoC," *2012 International Conference on Devices, Circuits and Systems (ICDCS)*, Coimbatore, 2012, pp. 673-677.

[10] M. Koushik, R. Anushree, B. J. Sowmya and N. Geethanjali, "Design of SPI Protocol with DO-254 Compliance for Low Power Applications," *2017 International Conference on Recent Advances in Electronics and Communication Technology (ICRAECT)*, Bangalore, 2017, pp. 186-190.

[11] A. K. Oudjida, M. L. Berrandjia, R. Tiar, A. Liacha and K. Tahraoui, "FPGA implementation of I2C & SPI protocols: A comparative study," *2009 16th IEEE International Conference on Electronics, Circuits and Systems - (ICECS 2009)*, Yasmine Hammamet, 2009, pp. 507-510.

[12] S. Saha, M. A. Rahman and A. Thakur, "Design and implementation of SPI bus protocol with Built-in-self-test capability over FPGA," *2014 International Conference on Electrical Engineering and Information & Communication Technology*, Dhaka, 2014, pp. 1-6.

[13] https://www.xilinx.com/support/documentation/data_sheets/ds160.pdf

[14] Shingare, Trupti D., and R. T. Patil. "*SPI implementation on fpga.*" *International Journal of Innovative Technology and Exploring Engineering (IJITEE)* 2.2 (2013): 7-9.

[15] https://www.digikey.com/eewiki/pages/viewpage.action?pageId=4096096

[16] https://surf-vhdl.com/how-to-design-spi-controller-in-vhdl/

[17] https://www.volkerschatz.com/hardware/vhdocl-example/sources/spi-master.html

[18] https://github.com/Nematollahi/SPI-FPGA-VHDL/blob/master/hdl/spi_master.vhd