Seri Çevresel Arayüzü için Evrensel Doğrulama Metodu ile Test Ortamının Oluşturulması (Creating Test Environment with UVM for SPI)

Buse Ustaoğlu, Ahmet Çağrı Bağbaba, Berna Örs
Elektrik ve Elektronik Mühendisliği Bölümü
İstanbul Teknik Üniversitesi
İstanbul,Türkiye
{ustaoglubu, bagbaba, siddika.ors}@itu.edu.tr

İnan Erdem
Anka Mikroelektronik Sistemler
İstanbul,Türkiye
inan.erdem@ankasys.com

***Özetçe* —Güvenilir bir sayısal sistem gerçeklemek için doğrulama ortamı hazırlanarak testlerinin yapılıp, açıklarının bulunması tasarımlar karmaşıklaştıkça büyük bir önem kazanmaktadır. Standartlaşmış ve çip endüstrisinde yaygın olarak kullanılan Evrensel Doğrulama Yöntemi ile yeniden kullanılabilir, etkili doğrulama ortamı hazırlamak mümkün olmaktadır. Bu çalışmada tümleşik devrelerin haberleşmesinde yaygın olarak kullanılan Seri Çevresel Arayüz protokolünün uydu devresinin donanım tanımlama ve doğrulama dili olan SystemVerilog ile tasarımı yapılmış, UVM yöntemi ile de test ortamı oluşturulmuştur.**

Anahtar Kelimeler—seri çevresel arayüz, evrensel doğrulama yöntemi, test, simülasyon

***Abstract*—In order to implement reliable digital system, it is becoming important making tests and finding bugs by setting up a verification environment. It is possible to set up effective verification environment by using Universal Verification Methodology which is standardized and used in worldwide chip industry. In this work, the slave circuit of Serial Peripheral Interface, which is commonly used for communication integrated circuits, have been designed with hardware description and verification language SystemVerilog and creating test environment with UVM.**

Keywords—serial peripheral interface(SPI), universal verification methodology(UVM), test, simulation.

I. GİRİŞ

Dijital tasarımlar günden güne karmaşıklaştıkça fonksiyonel doğrulamaların önemi hızla artmıştır. Tasarım açıklarının bulunması ve devrelerin doğru bir şekilde gerçeklenebilmesi için testlerinin yapılması, bu testler yapılırken doğru bir plan yapılarak ilerlenmesi gerekmektedir.

Günümüzde sıklıkla kullanılmakta olan Evrensel Doğrulama Metodu (Universal Verification Method, UVM), SystemVerilog dili kullanılarak gerçekleştirilen fonksiyonel doğrulamadır ve genelde entegre devre tasarımlarında kullanılan standartlaşmış bir metottur. Temelleri Açık Doğrulama Yöntemi'ne (Open Verification Methadology) dayanmaktadır. SystemVerilog gibi doğrulama dilleri her ne kadar gittikçe popülerlik kazansa da var olan doğruluk sistemleri 2011 yılında Accellera firmasını sayesinde birleştirilmiş ve UVM ismini almıştır [11]. UVM systemverilog tabanlı sınıf kütüphaneleri içermektedir ve bu sınıf kütüphaneleri uyarıcı içeren diziler ve doğrulama ortamını oluşturan yapısallar arasında iyi bir ayrım yapmaktadır. Bu sınıflar kullanılarak tasarımcı test ortamı oluşturabilir ve uyarıcılar üretebilir. Doğrulama ve modellemenin tüm ihtiyaçlarını karşılayabilmek amacıyla aynı zamanda Genel Amaçlı Nesne Yönelimli programlama yeteneklerine de sahiptir [6].

Sayısal devrelerin haberleşmeleri için yaygın olarak kullanılabilen haberleşme protokollerinden biri olan Seri Çevresel Arayüzü (Serial Peripheral Interface-SPI) protokolünün devre tasarımı VHDL, Verilog gibi donanım tanımlama dilleri ile yapılabilmektedir. Testlerinin ve doğrulamasının yapılması bir üst seviye sistemde devrelerin sorunsuz bir şekilde haberleşmeleri için önemlidir.

Bu çalışmanın geri kalan kısmı şu şekilde organize edilmiştir. İkinci başlıkta Genel Kavramlar olan Seri Çevresel Arayüz haberleşme protokolünün ve UVM'in ayrıntılı açıklaması yapılmış, üçüncü kısımda bu protokolün dijital tasarımına yönelik UVM ile test ortamı oluşturulmuştur, son kısım olan sonuç kısmında ise yapılan çalışma özetlenmiştir.

II. GENEL KAVRAMLAR

A. Seri Çevresel Arayüzü

Seri Çevresel Arayüzü (Serial Peripheral Interface-SPI), sayısal tümdevrelerin seri haberleşmeleri için geliştirilmiş verimli haberleşme standartlarından biridir. SPI ismi Motorola tarafından bulunumuştur. SPI protokolünün bir diğer adı ise Microwire'dır ve bu Natiaonal Semiconductor firmasının tescilli ticari markasıdır.

SPI, eş zamanlı çift yönlü (full duplex) çalışabilen, senkron (verinin saat darbeleriyle birlikte eşzamanlı olarak aktarıldığı) bir seri haberleşme standardıdır ve pek çok tümdevre tarafından donanımsal olarak desteklenmektedir. Bu haberleşmede veri transferi ana-uydu(master-slave) ilişkisi ile gerçekleşir. Ana cihaz veri haberleşmesini başlatan cihazdır. Transfer başlatıldıktan sonra veri her iki yönde de eş zamanlı olarak aktarılabilir. Alınan baytın anlamlı olup olmadığı bilgiyi alan cihaz karar vermektedir [10].

Basit bir donanım arayüzü olması, bit transferinin 8 bit ile sınırlandırılmaması, çip seçme hattı dışındaki diğer işaretlerin paylaşılması gibi avantajları bulunurken, sadece kısa mesafeli

haberleşmeler için kullanılabilmesi, standart bir protokol olmaması, geri besleme mekanizmasının olmaması gibi dezavantajları mevcuttur.

SPI veri haberleşmesi için 4 hat kullanılır.

1) SDO: Veri çıkış hattı(SerialDataOut)
2) SDI: Veri giriş hattı(SerialDataIn)
3) SCK: Saat sinyali hattı
4) CS(SS): Uydu seçme hattı(Chip-Slave Select)

Tipik kullanım Şekil 1'de gösterilen bağımsız uydu kurulumudur. Bağımsız uydu seçme hattı vardır ve uyduların MISO pinleri birbirlerine bağlı olduğundan üç durumlu lojik pinlerine ihtiyaç duyarlar.

İletişime başlamak için ana cihazdaki veri hattı saat işaretini ayarlar. Uydu seçme hattına lojik 0 göndererek ilgili uyduyu aktif hale getirir.

- Ana cihaz MOSI hattına bir bit gönderir, uydu aynı hattan veriyi okur.
- Uydu MISO hattına bir bit gönderirken ana cihaz aynı hattan veriyi okur.

Veri iletimi Şekil 2'de gösterildiği gibi iki adet ötelemeli kaydedici kullanılarak dairesel döngü şeklinde gerçekleşir [2].

SPI için 4 ayrı çalışma modu tanımlanmıştır ve bu modlar Tablo I'da verilmiştir. Bu çalışma modları için saat polaritesi (clock polarity-CPOL) ve saat fazı(clock phase-CPHA) olarak iki adet parametre tanımlanmıştır. Ana cihaz saat frekansını ile birlikte bu iki parametreyi ayarlamalıdır. Eğer saat fazı 0

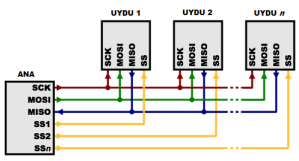
SPI Modu	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

Tablo I: SPI modları

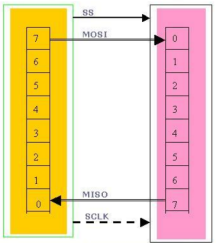
ise; saat polaritesi 0 iken veri saatin yükselen kenarında saat polaritesi 1 iken veri saatin düşen kenarında örneklenir. Saat fazı 1 olduğunda polariteler ters çevrilir.

B. Evrensel Doğrulama Metodu

Evrensel Doğrulama Metodu (UVM) Accelera Girişim Sistemleri tarafından Açık Doğrulama Yöntemi (OVM) baz alınarak geliştirilmiş bir standarttır. Sayısal donanımların işlevsel doğrulamasının yapılması için temel olarak simülasyonları kullanır. Yeniden kullanılabilir test bileşenlerinin oluşturulması sağlanır [9]. UVM açık kaynak kodludur, sınıf kütüphanelerinden oluşur ve dili SystemVerilog'dur [5].



Şekil 1: Tipik SPI Veriyolu [3]

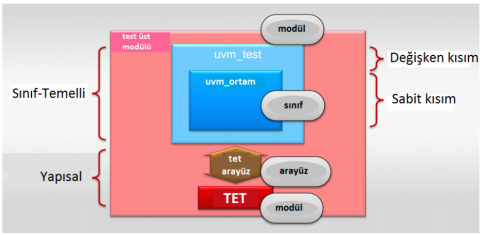


Şekil 2: Ötelemeli kaydediciler ile donanım ayarı [4]

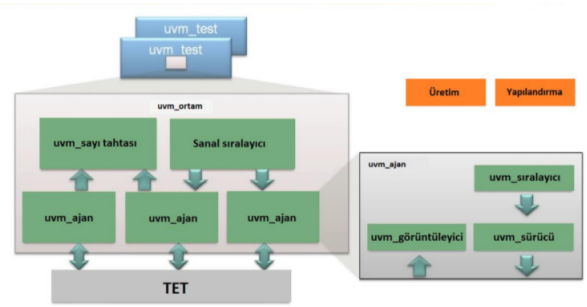
1) UVM Doğrulama Elemanları:

UVM test düzeneği temel eleman sınıfından türemiş iç içe yer alan sınıf serilerinden oluşur. Şekil 3'ten görülebildiği gibi sınıf temelli ve yapısal kısımdan oluşur.

- **Test Üst Modülü (Top Level):** Test edilen tasarımı, uvm_testi ve arayüzü kapsayan en üst modüldür.
- **Test Edilen Tasarım (Design Under Test):** Doğrulanması amaçlanan tasarımdır. Genellikle Donanım Tanımlama Dilleri (Verilog, VHDL) ile fonksiyonelliği tanımlanır.
- **Arayüz (Interface):** Doğrulama ortamı ile TET'in haberleşmesini sağlar. Bu arayüz TET'in pin seviyesi tanımını yapar.
- **Sanal Arayüz (Virtual Interface):** Tasarımdaki gerçek işaretlerin soyut modellerden ayrılmasını sağlayan bir mekanizma sağlar. Tasarımın farklı bölümlerindeki bir örneğin veya bir alt programın çalışmasına ve aynı verinin tüm bileşenlere geçmesine izin verir.
- **uvm_test:** Tasarıma göre yapılandırılbilen, değişken kısımdır.
- **uvm_ortam (Environment):** Doğrulama elemanlarının üst seviye bileşenidir. Bir veya daha fazla ajan içerebilir. Sabit kısımdır ve yeniden kullanılabilir bir yapının oluşturulmasını sağlayan kurulum özelliklerini içerir.
- **İşlemler (Transactions):** Testlerde üretilen ve TET'e sürücü aracılığıyla gönderilen işlemleri ifade eder. İşlemler SystemVerilog kısıt yapısı kullanılarak rastgele hale getirilir ve testler bu şekilde üretilir.



Şekil 3: UVM test düzeneği [8]



Şekil 4: UVM genel resim [7]

Tasarım Birimi	UVM Düzeneği Karşılığı	SystemVerilog kodu karşılıkları
top	test üst modülü	top.sv
spi_arayüz	arayüz	spi_if.sv
spi_uydu	TET	spi_slave.sv dut0-dut1- dut2-dut3
test_bayt test_kelime test_çift_kelime	uvm_test	testByte.sv testWord.sv testDoubleWord.sv
spi_ajan	uvm_ajan	spi_ajent.sv
spi_görünteleyici	uvm_görünteleyici	spi_monitor.sv
spi_sürücü	uvm_sürücü	spi_driver.sv
spi_sıralayıcı	uvm_sıralayıcı	spi_sequencer.sv
spi_sıraögeleri	uvm_sequence	spi_seq_lib.sv

Tablo II: SPI Test Düzeneği

- **uvm_ajan (Ajent):** Tasarımların çoğu değişik işaret arayüzlerine sahiptir. Ajanın amacı kullanıcıya pin seviyesi işlemleri üretip bunları görüntüleyebilmesi için doğrulama elemanlarının üretilmesini sağlar.
- **uvm_görünteliyici (Monitor):** Ajan içerisinde yer almaktadır. Pin seviyesi aktiviteleri görüntüleyerek işlevsel kapsamın hesaplanacağı kapsam toplayıcıya gönderilecek gözlemleri sıra ögelerine dönüştürür.
- **uvm_sürücü (Driver):** Ajan içerisinde yer almaktadır. Veriyi sıra ögelerinden pin seviyesi işlemlere dönüştürür. Sıralayıcıdan giriş alır ve bu girişi TET'e arayüz aracılığıyla yollar.
- **uvm_sıralayıcı (Sequencer):** Ajan içerisinde yer almaktadır. Ürettiği işlemleri sıralayıcıya aktarır. Eğer birden fazla değişken sürülmüş ise sıra ögeleri üreterek bunları sürücüye aktarır.

2) UVM Fazları:

Fazlar sıralı çalışma adımlarıdır. Bütün sınıflar simülasyon fazlarını içermektedir. En önemli fazların açıklaması:

- **İnşa fazı (build_phase):** Elemanların hiyerarşik olarak yapılandırılmasını sağlar. Örneğin ajanın inşa fazı görüntüleyici, sürücü ve sıralayıcıyı yapılandırır.
- **Bağlama fazı (connect_phase):** Farklı alt elemanları birbirine bağlayan fazdır. Ajanın bağlama fazı sürücüyü sıralayıcıya görüntüleyiciyi de dışarıdaki bağlantı noktasına bağlar.
- **İşletme fazı (run_phase):** Uygulamanın ana fazıdır ve simülasyonun asıl kodlarının koşturulduğu kısımdır.
- **Raporlama fazı (report_phase):** Simülasyon sonuçlarının gösterildiği fazdır [1].

III. Seri Çevresel Arayüz Test Ortamı

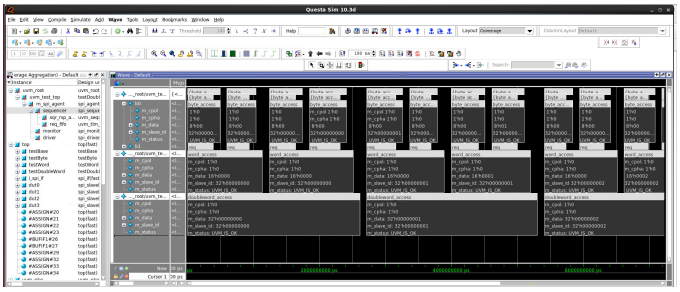
UVM ile SPI test ortamı tasarımı yapılmış SPI uydu devresi için oluşturulmuştur. UVM elemanları karşılık gelen SystemVerilog kodları Tablo II'da verilmiştir. Burada sınıf temelli kısıma düşen SPI test elemanları UVM elemanlarından türetilmiştir. Yani sınıf temelli UVM elemanları baz alınarak test edilecek tasarıma özgü kodlar yazılmıştır.

- Test üst modülünde *spi_arayüz* ile *spi_uydu*ları donanımsal olarak ve *uvm_test* tarafına ise arayüzlerin UVM veritabanı içerisine konularak bağlanır.
- *uvm_test* içerisinde öncelikle inşa fazı ile *spi_ajan* yapılandırılır. *spi_ajan*daki bağlama fazı bittikten sonra koşma fazı başlatılarak UVM veritabanından spi_arayüzü çekilir içerisinde bulunan *spi_sürücü* ve *spi_görünteleyici*ye atanır, *spi_uydu*ları için giriş değerleri üretilir *spi_sıralayacı* çağırılır.
- *spi_ajan* inşa fazı ile sürücü, *spi_görünteleyici* ve *spi_sıralayıcı*yı kurar bağlama fazı ile de bu elemanların bağlantılarını gerçekleştirir.
- *spi_sıralayıcı spi_uydu*larına üretilen giriş değerleri için sıra ögeleri oluşturup sürücüye gönderir. Bu test ortamında sıra ögelerinin yapacağı işlemler bir *spi_sıraögeler*inde tutulmuştur.
- *spi_sürücü spi_sıralayıcı*dan gelen işlemleri pin seviyesi sinyallere çevirerek *spi_uydu*lara gönderir.
- *spi_görüntüleyici*, *spi_uydu*lar ile haberleşerek, onların aldığı girişlere göre hangi çıkışları ürettiğini görüntüler.
- Bu aşamaların normal seyrinde gerçekleştiği ya da herhangi bir hata meydana geldiğini bildiren raporlama fazı ile son bulur.

A. Bayt, Kelime ve Çift Kelime Erişimi

SystemVerilog nesne yönelimli programlama dilidir ve bu dilin önemli özelliklerinden biri de miras kavramıdır. Buna göre bir sınıfın örnekleri,ata sınıfın örneklerini devralarak,ilgili nesneye kendine özgü yeni özelliklerini ilave ederek yeniden ortaya çıkar. Bu kısımda miras kavramından faydalanılarak bir türetilebilen testler yazılmıştır.

SPI uydu devresi için mesaj ve uydu numarası giriş değerleri üretilir. Mesaj bayt(8-bit), kelime(16-bit) ve çift kelime(32-bit) halinde gönderilir. *testWord.sv*, *testBayt.sv*'den yani ata sınıftan *testDoubleWord.sv* ise *testWord.sv*'den dolaylı olarak *testBayt.sv*'den doğrudan miras alarak türetilebilir. uvm_test değişken kısım olduğu için farklı testler yazılabilir ve ilgilenilecek test, test üst modülünde çağrılır. Testlerin simülasyonu Mentor Graphics firmasının ürettiği QuestaSim yazılımını 10.3d versiyonunda yapılmıştır. Şekil 5'de çift kelime erişimi işlemi içerisinde iki adet kelime ve dört adet bayt erişimi işlemi



Şekil 5: Bayt, Kelime ve Çift Kelime Erişimi

görülmektedir. Dört uydu devresine de mesajlar gönderilerek hepsinin test edilmesi sağlanır. Artık pin seviyesi değil bir üst seviye olan işlem seviyesinde grafiklerin gözükmekmesi tasarımın daha kolay bir şekilde doğrulanmasını sağlamaktadır.

IV. SONUÇLAR

Bu çalışmada sayısal tasarımların ağırlıklı olarak simülasyon ortamında test edilip açıkların bulunması ve doğrulanması için geliştirilen bir yöntem olan UVM ile yaygın olarak kullanılan SPI iletişim protokolünün gerçeklenmiş uydu devresi için test ortamı oluşturulmuştur. UVM'in sınıf temelli bir yapıya dayanması sabit kısmı bir kere yazıldıktan sonra farklı testler uygulanarak tekrar tekrar kullanılabilmesini sağlamaktadır. Testlere miras özelliği uygulanıp SPI uydu devresine farklı boyutlarda mesajlar gönderilip tasarımın davranışı gözlenebilmektedir.

KAYNAKÇA

[1] Defining the verification environment. http://colorlesscube.com/uvm-guide-for-beginners. [Accessed: January-2015].

[2] Serial peripheral interface bus. http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus. [Accessed: January-2015].

[3] Serial peripheral interface (spi). https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi. [Accessed: January-2015].

[4] Spi bus interface. http://www.eeherald.com/section/design-guide/esmod12.html. [Accessed: January-2015].

[5] John Aynsley. Uvm verification primer. https://www.doulos.com/knowhow/sysverilog/uvm/tutorial_0/, June 2010.

[6] Jonathan Bromley. If systemverilog is so good, why do we need the uvm? sharing responsibilities between libraries and the core language. In *Specification Design Languages (FDL), 2013 Forum on*, pages 1–7, Sept 2013.

[7] Tom Fitzpatric. Connecting env to dut. https://verificationacademy.com/sessions/uvm-connecting-env-dut. [Accessed: January-2015].

[8] Tom Fitzpatric. Uvm "hello world". https://verificationacademy.com/sessions/uvm-hello-world. [Accessed: January-2015].

[9] Rakhi Nangia and Neeraj Kr Shukla. Functional verification of i2c core using systemverilog. *International Journal of Engineering, Science and Technology*, 6(4):45–51, 2014.

[10] Ramazan Soral. *Spi nedir? Nasıl Çalışır?*, 2009. http://ramazansural.blogspot.com.tr.

[11] Young-Nam Yun, Jae-Beom Kim, Nam-Do Kim, and Byeong Min. Beyond uvm for practical soc verification. In *SoC Design Conference (ISOCC), 2011 International*, pages 158–162, Nov 2011.