# PUF, DPSR ve Bulandırma Yoluyla Sayısal Yongaların Güvenilir Yapılması

Sezer Gören, H. Fatih Uğurdağ, Özgür Özkurt

Bahçeşehir Üniversitesi Bilgisayar Mühendisliği Bölümü
Çırağan Cad., 34349 Beşiktaş, İstanbul

e-posta: fatih.ugurdag@bahcesehir.edu.tr

## Özetçe

Günümüzde hem ticari hem de askeri uygulamalar, yongaların (özellikle sayısal olanların) güvenilir kılınmasını gerektirmektedir. Yonga güvenilirliğinin 3 önemli alt başlığı bulunmaktadır: 1. Casus/İstenmeyen Modüller, 2. Tersine Mühendislik, 3. Klonlama. Bu bildiride PUF (Physically Unclonable Functions: Fiziksel Klonlanamaz Fonksiyonlar), DPSR (Dynamic Partial Self Reconfiguration: Dinamik Kısmi Kendi kendini Yeniden yapılandırma) ve Bulandırma (Obfuscation) tekniklerini anlattıktan sonra bunları kullanarak bukalemun yongalar (FPGA) üzerinde güvenilir yonga elde edilmesi ile ilgili yeni bir yöntem sunacağız.

## 1. Giriş

"Güvenilir Yongalar" (Trusted ICs), son zamanların revaçta çalışma konularından biridir. Bu konuya ilgi, 2006 yılında Amerikan İleri Savunma Araştırma Projeleri Kurumu'nun (DARPA) [1]'deki çağrısı ile artmıştır.

Yonga güvenilirliği, sadece savunma dünyasında değil ticari dünyada da önemli hale gelmiştir. Ticari dünyada yongalardaki güvenlik eksikliği, yongayı tasarlayan şirkete ciro kaybı olarak geri döner. (Bir araştırmaya göre, 2005 yılında Amerikan şirketleri fikri hırsızlık sonucunda 200 milyar dolarlık ciro kaybına uğramıştır [2].) Öte yandan, savunma uygulamalarında ise güvenlik eksikliği, savunma teknolojilerinde zafiyet doğurur.

Özetçede de belirtildiği gibi yonga güvenilirliği 3 belli başlı konuya ayrılabilir:

1. Casus/İstenmeyen Modüller (Truva Atları)
2. Tersine Mühendislik
3. Klonlama

"Casus/İstenmeyen Moduller" ile kasıt, çok özel sinyal girdisi kombinasyonlarında tetiklenerek yonganın işlevini bozan (veya farklı/istenmeyen bir iş yaptıran) ama standart test yöntemleriyle bulunması zor modüllerdir.

"Tersine Mühendislik" ile kastedilen ise, tasarımı içindeki mühendislik çözümlerini anlamaya yönelik hırsızlık faaliyetidir. Bu bilgiden yola çıkarak, karşı taraf sizin çözümlerinizi kendisine mal edebilir.

"Klonlama" ise tersine mühendislik yapmadan tasarımın aynısını kullanıp satma veya kendi sistemlerinde kullanma faaliyetini mümkün kılar.

Güvenilir yonga çözüm ve yaklaşımları, yonganın gerçeklenme türüne

1. ASIC
2. FPGA

göre farklılık gösterir.

Biz bu bildiride Türkiye'de bukalemun yonga da tabir edilen FPGA'ler (Field Programmable Gate Array: Sahada Programlanabilir Kapı Dizileri) üzerine yukarıda bahsedilen 3 alt konuda çözüm önerileri sunacağız. Öte yandan fikir verme amaçlı olarak, ASIC'ler (Application Specific Integrated Circuit: Özel Amaçlı Tümleşik Devre) üzerine bugüne kadar öneriler bazı çözümleri aşağıda özetliyeceğiz.

## 2. ASIC'lerde Yonga Güvenilirliği

Bir yonganın özel amaçlı oluşu işlevsel ve/veya fiziksel açıdan olabilir. Bir ASIC, hem işlevsel hem de fiziksel açıdan özel amaçlı, yani belli bir uygulamaya hizmet eden bir yongadır. Öte yandan, bir FPGA'e özel bir işlevi gerçekleyen bir bit dizini (bitstream,) yani bir tasarım, indirdiğimizde elimizde işlevsel açıdan özel amaçlı bir yonga oluşmasına rağmen, fiziksel açıdan hala genel amaçlı bir yonga kullanıyor oluruz.

Günümüzde, sayısal yonga üretimi (özellikle ASIC'ler için) tasarımdan tamamen ayrı bir fonksiyon haline gelmiştir. Bu nedenle üretim yapmayan yonga şirketleri kurulmuş (fabless semiconductor companies) ve bunların ciroları milyarlarca doları bulmuştur (Qualcomm ve Broadcom gibi.) Bugün sayısal yonga satan şirketler içinde Intel gibi aynı yongadan onlarca milyon yapan şirketler dışında kalanlar, üretim tesislerinin maliyetini karşılayamamakta ve yonga üretimini fason olarak çoğu Uzakdoğu'da olan (TSMC gibi devasa) şirketlere yaptırmaktadırlar. Bu sektördeki fason üreticiler Tayvan, Çin, Japonya, Singapur ve Kore'dedir. Bir-iki tane de Avrupa ve Amerika'da oyuncu vardır. Yarı-iletken sektöründe iş yapan büyük Amerikan ve Avrupa şirketlere "fabless" modele kayarken ufak şirketlerin başka bir model benimsemesi düşünülemez. Düşük üretim hacimlerinde, bırakın kendi üretim tesisinizi kurmanın ekonomik olmasını (en az bir-kaç milyar dolar gerektirir,) size özgü bir ASIC ürettirmeniz bile güçtür (yonga başına milyon dolar civarı NRE gerekir.) Askeri uygulamalarda da üretim hacmi düşük olduğu için (milyonlarca yonga değil, en fazla binlerce yonga) "fab" (yonga üretim fabrikası) gerektiren bir model aşırı maliyetlidir ve bir süre sonra yatırım eksikliği nedeniyle en son teknolojinin gerisinde kalmaya mahkumdur.

Şu noktada ASIC yapıyorsak, yonga tasarımımızı geometrik seviyede de olsa (layout) bir üretim fabrikasına gönderiyoruz demektir. Bu demektir ki, en büyük güvenlik zafiyeti üretim şirketidir. Bu şirkette yongamızın içine casus modül de konulabilir, tasarımımız tersine mühendislik veya daha kolayı klonlama yoluyla çalınabilir de. FPGA bit dizinleri gibi yongamızın tasarım geometrisini (bir GDSII dosyası) şifreleme imkanımız yoktur. Çünkü şifreli bir geometri dosyasını şirket üretilmiş yongalara dönüştüremez.

Casus modüllerin ayıklanması noktasında imdadımıza üretim test (Manufacturing Test veya VLSI Test denilen) metotları gelir. Yonganın tasarımından çıkarılan bazı sayısal

imzalar (digital signatures) üretilen yonganın sayısal imzaları ile karşılaştırılır [3, 4].

Klonlama konusunda ise yardımımıza yazılım sektöründe de kullanılan bulandırma (obfuscation) teknikleri gelir. [5] nolu referanstaki bulandırma tekniğinin tasarımınıza eklediği ek donanım, aynı casus modüller gibi belli bir sinyal girdi kombinasyonunda (bir çeşit sayısal imza/lisans/sertifika) tetiklenir. Ancak casus modüller tetiklendiğinde tasarımınız yanlış davranırken, bu teknikte tetiklenme noktasından sonra tasarımınız işlevini görmeye başlar. Bu teknik tasarımın çeşitli noktalarına XOR kapıları koyarak tasarımınızı anlaşılmaz kıldığı gibi, XOR'lara sıfır değeri beslenmedikten sonra tasarım çalışmaz (yani boot etmez.)

Yukarıdaki bulandırma yönteminin belli bir yonga için kullandığı sayısal imzayı yonganın kullanıcısına veririz. Yongayı kendimiz kullanmak üzere ürettiriyor isek, bu sayısal imza oldukça güvenilir ellerde demektir. Eğer ticari müşterilerimiz var ise, bu sayısal imzanın sadece müşteriler tarafından bilinmesi ve üretim şirketinin eline geçmemesine dikkat etmemiz gerekir.

Bu sayısal imzayı her üretilen yonga için farklı kılmak oldukça yeni bir teknoloji olan PUF'lar (Physically Unclonable Functions: Fiziksel Klonlanamaz Fonksiyonlar) ile mümkündür. Her müşteri için farklı bir kod kullanmak klonlayan kişilerin işini zorlaştırdığı gibi hırsızlığın ispatlanmasını kolaylaştırır.

PUF'ın ne olduğunu bir sonraki bölümde anlatacağız. PUF'ları biz de FPGA tabanlı yonga tasarımlarının güvenilirliği konusundaki özgün çözümümüzde kullanacağız.

[5]'deki tasarım bulandırma yöntemi klonlama yanında aynı zamanda tersine mühendislik için de kullanılabilir. Çünkü devreye enjekte edilen XOR'lar, tetikleme dizininin üretici tarafından bilinmemesi nedeniyle, üreticiye manasız bir devre gibi gözükür.

Yonga tarafımızdan üretilirse içine üretim safhasında casus modül konulması ihtimali kalkmış olur. Ancak yonga tasarımımızın içine VLSI CAD yazılımları tarafından casus modül konulabilir. Bunu yakalamanın yolu değişik VLSI CAD safhalarında değişik şirketlerden yazılımlar kullanıp, bunların çıktıları arasında formal yöntemlerle karşılaştırma yapıp devremizin aynı kaldığını kontrol etmektir.

Yongayı bizim üretmemiz tersine mühendisliği tamamen engellemez. Masraflı da olsa karşı taraf, yongalarımızdan birinin kapağını açıp kimyasal işlemlerle yonganın geometrik tasarımını çıkarabilir [6]. Tersine mühendislik yapıldıktan sonra da yonga klonlanabilir.

Daha önce de belirttiğimiz gibi, bizim bu çalışmadaki akademik katkımız FPGA tasarımlarının güvenilirliğine yöneliktir. Önerdiğimiz yeni yöntem DPSR, PUF ve devre bulandırma tekniklerini özel bir şekilde bir araya getirmektedir. Bu nedenle, bizim yöntemimizi anlatmadan önce bu 3 yöntemi/teknolojiyi anlatacağız.

## 3. PUF

İngilizce kısaltması ile PUF tabir edilen fiziksel klonlanamaz fonksiyonların amacı, her yonganın kendisine ait kopyalanamaz bir seri numarasına sahip olmasıdır [7]. Bir PUF'ın çıktısı olan sayı o yongaya özgüdür. Bunun nedeni, sayının üzerinde bulunduğu yonganın üretimi sırasında oluşan karakteristik varyasyonlara bağlı olmasıdır. Bu varyasyonlar kontrol altında tutulup belli bir PUF seri nosu hedeflenemez, çünkü zaten tanım gereği bu varyasyonlar üretim ekipmanının toleransının (hata payının) parçasıdır.

Bu bildirinin asıl ilgi alanı FPGA'ler olduğu için, bundan sonraki PUF tartışmamızı FPGA'ler bağlamında yapacağız. PUF'ın FPGA'lerde akla ilk gelen uygulaması, tasarım bit dizininin FPGA'deki PUF'ı kullanarak yaratılmasıdır. Bu sayede ilgili bit dizininin başka FPGA'lerde çalışmasını engelleyebiliriz. Yani klonlamayı engelleyebiliriz. Aynı zamanda belli bir FPGA'e yanlış bir bit dizininin indirilip, FPGA'in içinde bulunduğu sisteme ve sistemin çevresine zarar vermesine engel olabiliriz. Yani PUF sayesinde, ne o FPGA farklı bir bit dizinini kabul eder, ne de bit dizini farklı bir FPGA'i. Ancak tersine mühendislik yolu ile bit dizini içine casus modül konulursa, o zaman değiştirilmiş bit dizini aynı FPGA'e konulabilir.

Bir FPGA'e casus modüllü veya daha geniş deyimiyle içinde "trojan" (Truva atı) olan bir bit dizini indirmeye çalışılmasındaki amaç bir ATM makinasından para çalmak, oy verme makinasından oy vererek seçimleri manipule etmek, kasino (kumar) makinalarını manipule etmek olabilir [9]. Son zamanlarda tüm güvenilir yongalara alanında olduğu gibi FPGA bit dizinlerindeki truva atlarının bulunması konusunda da çalışmalar artmış bulunmaktadır [10].

Bir FPGA bit dizininin orijinali olup olmadığını anlama işlemi aşikardır ki bir otantikasyon (authentication) işlemi olarak düşünülmelidir. Otantikasyonda amaç, diğer şifreleme işlemleri gibi kriptoloji teknikleri kullanmasına rağmen bir tekstin anlaşılırlığını engellemek değildir. Buradaki amaç, tekstin bozulup bozulmadığını (tamper detection) anlamaktır. Bunun için bir çeşit sayısal imza yöntemi kullanılabilir. Bozulan bir tekstin sayısal imzası da değişir. Karşılaştırdığımız imza eğer bir PUF ise, bu imza her FPGA için farklı olacağı için "hacker"ın o FPGA için ayrı bir yanıltma yöntemi kullanması gerekmektedir. Yani geçerli bir yanıltma yöntemi olsa bile, o FPGA'deki PUF'ı öğrenmesi ve yöntemini o PUF için ayrıca şekillendirmesi gerekmektedir. Bu mümkün olsa bile bir FPGA'in içindeki PUF'ı öğrenmek FPGA'in bacaklarına erişim sağlansa bile imkansıza yakındır.

## 4. DPSR

DPSR, İngilizce "Dynamic Partial Self Reconfiguration" teriminin baş harflerinden oluşan bir kısaltma olup "Dinamik Kısmi Kendi kendini Yeniden yapılandırma" manasına gelir. Burda kastedilen, bir FPGA'in çalışırken kendi kendisinin yapısını değiştirmesidir [11]. Dinamik kelimesi ile kasıt, FPGA'in işlevine aralık vermemesidir. Kısmi kelimesi ile kasıt, FPGA'in bir kısmının yapısının değişmesidir. Yeniden yapılandırma ile kasıt, FPGA'in programlanması yani bit dizininin değişmesidir. Kendi kendine ile kasıt, yapılandırılan FPGA yongasının dışındaki bir donanımın bit dizinini beslemesi ve kontrol etmesi yerine FPGA'in içindeki bir modulün bu işi yapmasıdır.

DPSR, DPR teknolojisinin bir üst versiyonu olarak düşünülebilir. DPR ise PR'ın bir üst kademesidir. PR, "Partial Reconfiguration" yani "Kısmi Yeniden yapılandırma" demektir. PR sırasında FPGA işlevini göremez. PR'daki amaç, %100 bir yapılandırmaya göre yapılandırma zamanından ve de bit dizininin bellek büyüklüğünden tasarruf etmektir. DPR da ise (Dynamic Partial Reconfiguration: Dinamik Kısmi Yeniden yapılandırma) amaç, kısmi yapılandırma sırasında FPGA'de bazı kritik işlevlerin (yani FPGA'in statik bit dizini) çalışmaya devam etmesini mümkün kılmaktır. DPSR'da DPR'dan farklı olarak kısmi yapılandırma için bir dış donanım öğesine ihtiyaç olmamasıdır.

PR, platform bağımsız ve kavramsal bir terim gibi kullanılıyor olsa da aslen FPGA sağlayıcı en büyük şirket olan Xilinx kökenli bir terimdir. İkinci büyük FPGA şirketi olan Altera aynı kavram için SPR (Software Programmable Reconfiguration: Yazılım tarafından Programlanabilir Yeniden yapılandırma) terimini kullanmaktadır. Öteki taraftan FPGA

piyasasında daha ufak payı olan diğer şirketlerin çoğu (örneğin: Lattice Semiconductor) PR terimini benimsemiştir.

FPGA'lerin yararlarından biri farklı uygulamalar arasında donanım paylaşımıdır. Bu sayede donanıma yazılım-vari bir özellik kazandırarak (her bit dizini değişikliğinde) davranış değişikliği kabiliyeti kazandırırlar. FPGA'ler bu özelliklerine rağmen bir yazılım çözümü değil tam anlamıyla bir donanım çözümüdürler. Çünkü içlerinde koşan hesaplama birimleri paralel donanım öğelerinden oluşur ve tamamen eldeki uygulamaya özgü bir şekilde tasarlanmış ve oluşturulmuşlardır.

Öteki taraftan PR, FPGA'lerin bu davranış değişikliği kabiliyetinin maliyetini daha ucuz bir hale getirirler. PR özelliği olmayan bir FPGA'e göre PR'lı bir FPGA daha ufak bir flash bellek ile yetinebilir ve bunun yanında davranış değişikliğini daha çabuk gerçekleştirir. DPR, yukarıdakilerin tümünü yapar ve bunların yanında davranış değişikliği FPGA'e yüklenirken FPGA bazı hayati işlevlere devam edebilir, kullanıcısına cevap verebilir. DPSR ise tüm yukarıdakileri yapmanın yanında elektronik sistemin toplam maliyetini daha da düşürür. Bunu da bit dizini değişikliklerini yapmak için PCB (baskılı devre) üzerinde ek bir işlemci gerektirmeyerek başarır.

Donanım paylaşımı gerektiren ve bunun dinamik yani hayati işlevleri devam ettirerek olmasını gerektiren bir uygulama cep telefonlarıdır. Ülkemizde artık bir çok yerde hem GSM hem de GSM'in yeni jenerasyonu olan 3G denilen cep telefonu şebekesi (yani iletişim ağı) vardır. GSM yanında 3G özelliği olan telefonlar, 3G şebekesi olan yerlerde bağlantıyı 3G üzerinden yapabilirler (aynı yerde GSM şebekesi olduğu halde.) Telefonun 3G özelliği kullanılırken GSM ile ilgili donanım modülleri kullanılmamaktadır. GSM özelliği kullanıldığında da 3G modülü kullanılmaz. Bu durumda GSM ve 3G modüllerine donanım paylaşımı yaptırılıp sistem maliyeti düşülebilir. Ama bunun yaparken ilgili yonganın bazı işlevleri yapmaya devam etmesi gerekmektedir. Tüm bunlar DPR veya DPSR özelliğine sahip bir FPGA ile mümkündür. (Aynı donanımı değişik zaman aralıklarında paylaşıma müsait iletişim teknoloji/protokolleri Kuzey Amerika'da daha da fazladır.) Aynı işe yarayan ama farklı ve aynı zamanda kullanılmayan (time shared, time multiplexed) uygulamalar iletişim dünyasında oldukça fazladır (örnek: VoIP'de kullanılan vocoder'lar yani insan sesi için codec'ler).

DPSR'ın diğer faydaları arasında anlık şartlara uyum sağlayabilen donanımlar, uygun maliyetli ve(ya) uzaktan donanım güncelleştirmesi (upgrading) de sayılabilir. DPR özelliği olan bir FPGA'in tasarım akışı, normal FPGA tasarım akışından bir çok noktada farklıdır:

(i) Dinamik ve statik modüller arasına karşılıklı haberleşme için sabit veri yolu makroları yerleştirilir.

(ii) Dinamik modüller projenin genelinden ayrı olarak sentezlenir.

(iii) FPGA için yerleşim planı, sistemin kendisi ile beraber bütün dinamik modüller incelenerek yapılır.

(iv) Dinamik modüllerin değiştirilebilmesi için sistem içerisinde veya dışarıda bir yönetici modüle ihtiyaç vardır.

## 5. Devre Bulanıklaştırma

[5] nolu referansta anlatılan teknik (ObfusFlow) kapı devreleri seviyesinde önerilen ilk bulandırma tekniğidir. Daha önce RTL seviyesinde bulandırma teknikleri önerilmiştir. C gibi bir dilde yazılmış yüksek seviyeli bir kod yazılım dünyası için ne ise, RTL de sayısal donanım dünyası için aynı şeydir. RTL (Verilog veya VHDL dilinde olabilir) sentez aracı tabir edilen bir çeşit derleyiciden geçtikten sonra bir kapı devresine dönüşür. Bu nedenle kapı devreleri de yazılım dünyasındaki "Assembly" kod gibi düşünülebilir.

ObfusFlow ile IP'sini (tasarım kütüphane modüllerine donanım dünyasında verilen isim) bulandırmak isteyen bir tasarım şirketi önce tasarımını ObfusFlow'u kullanmadan tasarlar. Daha sonra sentezlenmiş tasarımı alır ve buna bir FSM (Finite State Machine: Sonlu Durum Makinası) ekler. FSM, bulandırılmaya çalışılan bloğun giriş bacaklarının tümünü veya bir kısmını izler ve bu bacaklardan belli bir dizin geldiğinde artık 1 yerine 0 değerinde bir çıkış vermeye başlar. FSM'in çıkışı tasarımın bazı yerlerine enjekte edilen XOR'ların girişlerinden birini besler. Bloğun girişine o "sihirli dizin" (tetikleme dizini) beslenene kadar, XOR'ların eklendiği devre düğümlerinde yanlış değerler varken, sihirli dizinden sonra XOR "inverter" yerine düz tel gibi davranmaya başlar ve ilgili devre düğümlerinde doğru değerler oluşur.

ObfusFlow'u kullanan şirketler müşterilerine bu sihirli dizini verir ama üreticiye vermez. ObfusFlow, hem ASIC'ler için hem de FPGA'ler için kullanılabilir. ObfusFlow klonlamaya karşı bir çözüm olduğu kadar tersine mühendisliği de bir çözümdür.

Ancak ObfusFlow'un bazı problemleri bulunmaktadır. Eğer sihirli dizin müşterilerden birinden sızar ise yapabilecek bir şey yoktur. ObfusFlow burada bir "digital watermarking" tekniği olarak kullanılabilir. Eğer IP'nizi korsan olarak kullanan sistemin bir kopyasını ele geçirebilir ve orada IP'nizi izole edebilirseniz, bir sihirli dizin kredi kartı numarası gibi bir şey olduğu için IP'nin size ait olduğunu ispatlayabilirsiniz. Böyle bir duruma karşı [5]'teki öneri her müşteriye ayrı bir dizin verilecek şekilde ObfusFlow'un kullanılması, yanı her farklı müşteri şirket için bir IP üzerinde ObfusFlow'un tekrar koşturulmasıdır. ObfusFlow bu durumda sihirli dizinin hangi müşterinizden sızdığını da gösterir.

ObfusFlow, XOR'ları kapı dizinine rastgele koymaz. Devre düğümleri bulandırma etkilerine göre sıralanır. Bunun için bir düğümün "fan-in" ve "fan-out" kapı konilerinin büyüklüklerine bakılır. İki yöne giden iki koni söz konusu olduğu için aslında bu bir "kum saati"dir. Kum saati en büyük olan düğüm alınır ve oraya bir XOR konulur. Daha sonra diğer tüm aday düğümlerin kum saati büyüklükleri güncellenir ve tekrar sıralanır. Sonra bu şekilde ikinci, üçüncü ve daha sonraki (kullanıcı tarafından belirlenen bir sayıya kadar) düğümler seçilir ve XOR'lar bu düğümlere enjekte edilir. [5] nolu referans ObfusFlow'u ISCAS-89 devreleri üzerinde koşturup bu tekniğin devrenin büyüklüğü ve güç tüketimi üzerinde önemli bir artışa neden olmadığı gösterilmiştir. Ayrıca simülasyonlarla ObfusFlow'un bulandırma etkisi tescillenmiştir.

ObfusFlow oldukça etkili bir teknik olmasına rağmen sihirli dizinin sızma ve klonlanan tasarımlardan haberimiz olmama olasılığı rahatsız edicidir. Buna karşı bir sonraki bölümde de anlatacağımız gibi sihirli dizin için PUF kullanmayı öneriyoruz. PUF'ların tasarım akışına monte edilmesi FPGA'lerde daha da kolaydır. Çünkü PUF'ın değerine göre yeni bir bit dizini üretebilirz. Ancak ASIC'lerde ise her yonga için (yani her PUF için) yeni bir ASIC tasarımı (GDSII) üreticiye ürettiremeyiz.

Ayrıca XOR'lar yerine MUX kullanarak ObfusFlow'un tersine mühendisliğe karşı bulandırma etkisini arttırmak mümkündür.

## 6. FPGA Güvenliği için Yeni Yöntemler

Yukarıdaki 3 tekniği özel bir şekilde birleştiren yöntemlerimizle FPGA güvenliğini tehdit eden 3 unsuru ayrı ayrı tartışacağız.

### 6.1. Casus/İstenmeyen Modüller ve Truva Atları

Yukarıda az önce 5 nolu bölümde zaten bunu nasıl yapabileceğimizden bahsettik. Çözümümüz [5] nolu referanstaki ObfusFlow isimli bulandırma yöntemini ve PUF'ları kullanmaktan geçiyor. ObfusFlow'un sihirli dizini her müşteriye değil her FPGA'e özgü olacak. Sihirli dizini dışarıdan değil PUF'tan alacak şekilde ObfusFlow'u değiştirmeyi öneriyoruz.

Ayrıca XOR'ları MUX'lara çevirmeyi ve bunların farklı kontrol değerleri ile sürülmesini öneriyoruz. Bu devrenin tersine mühendisliğe karşı dayanıklılığını arttıracaktır.

[5]'te düşünülmeyen başka bir konu da, sihirli dizin beslenene kadar devrenin çevresine zarar vermesini önlemek için eklenmesi gereken ekstra devre elemanlarıdır. Bu, her devre çıkışına bir MUX (bulandırma amaçlı değil) konularak yapılabilir. Ancak bu MUX'lar ObfusFlow'un eklediği FSM'den sürülürse, tersine mühendisliğe başlangıç noktaları verdiğimiz için tersine mühendisliği kolaylaştırmış oluruz. Bu nedenle bu MUX'ları bir "timer" ile kontrol etmeyi öneriyoruz.

Bir müşterinin veya başka kişilerin önerdiğimiz otantikasyon mekanizmasını atlatabilmesi için klonlama için bile tersine mühendislik yapması ve tasarımımız içinde PUF'ı üreten alt-modülü bulması gerekmektedir. Bu da oldukça "intractable" bir iştir. Yalnız ObfusFlow çalıştırılmadan önce FPGA'e farklı bir tasarım yüklenip PUF'ın ürettiği değer öğrenilmelidir. Yalnız böyle bir yaklaşım yine tersine mühendislere kolaylık sağlar. Bu nedenle tek bir tasarım ama 2 farklı mod öneriyoruz. İlk modda tasarım bacaklardan birinden dışarıya PUF değerini bildirecektir. Bu değer, her FPGA için IP'nin yani tasarımın sahibi olan şirketin İnternet sunucularından bir sihirli dizine dönüştürülecektir. Bu sihirli dizin (bir çeşit anahtar) sadece hangi FPGA için kesildi ise o FPGA'de çalışacaktır. Farklı bir FPGA farklı bir PUF değeri üretecektir (içsel olarak). Korsan bir firma FPGA'lerine İnternet sunucularımızdan sihirli dizin elde edemeyecektir.

Tüm bu güvenlik önlemlerinin üstüne, DPSR kullanarak PUF ileri aşamalarda kısmi bit dizinleri ile oluşturulup PUF devresinin tersine mühendisliği iyice zorlaştırılabilir.

Bu akış, casus modüllerin varlığında çalışabilir. Çünkü korsanlar, tasarımızın ufak bir kısmını değiştirecekleri için PUF ve onu otantike eden devremiz FPGA'e inecektir. Öteki taraftan korsanlar FPGA'e tamamen farklı bir bit dizini indirip (yani Truva Atı) elektronik sistemimizin istenenden farklı davranmasına neden olabilirler. Bu durumda sistemimize ekleyebileceğimiz içinde flash bellek olan bir CPLD (ufak non-volatile bir FPGA) tüm diğer önlemlerimize ek olarak FPGA'in üretmesi gereken PUF'ı doğrulayabilir.

### 6.2. Tersine Mühendislik

Klonlama, bazı kabiliyeti yüksek korsanlar tarafından tercih edilmez. Çünkü sayısal imza (digital watermarking/signatures) teknikleri ile tasarımın bir klon olduğu ispatlanıp kanuni işlem başlatılabilir. Bu nedenle bazı korsanlar, tersine mühendislik yoluyla tasarımı çözüp, değiştirip, kendilerine mal edebilirler. Askeri uygulamalar da dahil olmak üzere bazı kontekstlerde amaç zaten korsanlık ve kaçak üretim/satış değil, çeşitli sırları çalmaktır.

Tersine mühendisliğe karşı çoğu şirket bugün şifrelenmiş bit dizinleri ve bunları "on-the-fly" çözen FPGA'ler kullanmaktadır [12]. Ancak bu FPGA'lerin arka-kapıları (back-door) olabilir. Bu arka-kapılar FPGA'i tasarlayan şirket ve o ülkenin istihbarat şirketine açık olabilir. Bu arka-kapıyı açan bir sinyal dizisi FPGA'e uygulandığında, FPGA tasarım bit dizinini dışarıya besleyebilir. Bu nedenle %100 güvenli olmak istiyorsak şifreli FPGA'ler bize fayda sağlamaz.

Yalnız böyle bir arka-kapı, bizim bulandırma ve PUF kullanan yöntemimizi de tersine mühendisliğe karşı savunmasız duruma getirir. Korsanlar, sihirli dizi tasarımımızı normal çalışma moduna getirdikten sonra arka-kapıdan hem bit dizinini hem de flop ve LUT içeriklerini alabilirler. Buna karşı DPSR kullanabilir ve belli bir andaki bit dizini ve iç lojik değerlerinin bilinmesini yetersiz kılabiliriz. Bu durmda da eğer gizli bir "freeze" sinyali var ise tersine mühendislik müteakip birden çok freeze ve bit dizini sorguları ile çözülebilir. Yalnız not edilmelidir ki, buradaki varsayım korsanın elinde ürünümüzün çalışır vaziyette olan bir kopyasının olmasıdır. Aksi durumlar için yöntemimiz tersine mühendisliğe karşı çok güvenilirdir.

Bir çözüm, sihirli bit dizininin FPGA'imizi içeren elektronik sistemin kullanıcısı tarafından dışarıdan (örneğin bir keypad'den) beslenmesidir. Yalnız bu durumda bit dizininin dışarıya sızma tehlikesi vardır.

En garanti çözüm, yine flash-tabanlı ek bir CPLD ile FPGA'imizin çıkışlarını takip edip freeze durumunu tespit edebiliriz. Böyle bir durumda FPGA'i resetleyebiliriz.

### 6.3. Klonlama

Bulandırma, PUF ve DPSR içeren yöntemimiz kullanıldığında klonlama yapılması tersine mühendislik yapılmadan mümkün değildir.

### Kaynakça


[1] D. Collins, *Trust RFI Announcement*, www.darpa.mil /mto/solicitations/sn06-25, 11 Nisan, 2006.

[2] M. Moran, *How to implement high-security in low-cost FPGAs*, Programmable Logic DesignLine (online okunabiliyor,) 4 Aralık, 2006.

[3] F. Wolff, C. Papachristou, S. Bhunia ve R.S. Chakraborty, *Towards Trojan-Free Trusted ICs: Problem Analysis and Detection Scheme*, Proc. DATE Konferansı, 2008.

[4] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi ve B. Sunar, *Trojan Detection Using IC Fingerprinting*, Proc. IEEE Security and Privacy Sempozyumu, 2007.

[5] R.S. Chakraborty ve S. Bhunia, *Hardware Protection and Authentication Through Netlist Level Obfuscation*, Proc. ICCAD Konferansı, s. 674-677, 2008.

[6] R.J. Anderson ve M.G. Kuhn, *Tamper Resistance - a Cautionary Note*, Proc. USENIX Workshop on Electronic Commerce, s. 1-11, 1996.

[7] B. Gassend, D. Clarke, M. van Dijk ve S. Devadas, *Silicon Physical Random Functions*, Proc. Computer Communication Security Konferansı, s. 148–160, 2002.

[8] S. Kumar, J. Guajardo, R. Maes, G.J. Schrijen ve P. Tuyls, *The Buttefly PUF: Protecting IP on every FPGA*, Proc. IEEE International Workshop on Hardware Oriented Security and Trust, 2008.

[9] J. McDonald, *Intellitech: FPGAs and Security (CEO C.J. Clark ile röportaj)*, www.fpga-report.org/intellitech.htm, 25 Ekim, 2008.

[10] S. Dutt ve L. Li, *Trust-Based Design and Check of FPGA Circuits Using Two-Level Randomized ECC Structures*, ACM Trans. on Reconfigurable Technology and Systems (TRETS), vol. 2, no. 1, 2009.

[11] S. Bayar ve A. Yurdakul, *Dynamic Partial Self-Reconfiguration on Spartan-III FPGAs via a Parallel Configuration Access Port (PCAP)*, Proc. HiPEAC Workshop on Reconfigurable Computing, 2008.

[12] A. El-Ashmawi, *Designers can protect IP with FPGAs and bitstream encryption*, Military & Aerospace Electronics (online okunabiliyor,) Nisan, 2007.