İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

MANTIKSAL KİLİTLEME TEKNİĞİ İLE ÜÇÜNCÜ TARAF FİKRİ MÜLKİYETLERİN KORUNMASINA YÖNELİK KRİPTO MİMARİ VE FPGA GÜVENLİK MODÜLÜ TASARIMI

YÜKSEK LİSANS TEZİ

Teyyar AÇAR

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Elektronik Mühendisliği Programı

OCAK 2026

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

MANTIKSAL KİLİTLEME TEKNİĞİ İLE ÜÇÜNCÜ TARAF FİKRİ MÜLKİYETLERİN KORUNMASINA YÖNELİK KRİPTO MİMARİ VE FPGA GÜVENLİK MODÜLÜ TASARIMI

YÜKSEK LİSANS TEZİ

Teyyar AÇAR
(504981121)

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Elektronik Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Sıddıka Berna ÖRS YALÇIN

OCAK 2026

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

DESIGN OF A CRYPTOGRAPHIC ARCHITECTURE AND FPGA SECURITY MODULE FOR PROTECTING THIRD-PARTY INTELLECTUAL PROPERTY USING THE LOGIC LOCKING TECHNIQUE

M.Sc. THESIS

Teyyar AÇAR
(504981121)

Department of Electronics and Communication Engineering

Electronics Engineering Programme

Thesis Advisor: Prof. Dr. Sıddıka Berna ÖRS YALÇIN

JANUARY 2026

İTÜ, Lisansüstü Eğitim Enstitüsü'nün 504981121 numaralı Yüksek Lisans Öğrencisi Teyyar AÇAR, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "MANTIKSAL KİLİTLEME TEKNİĞİ İLE ÜÇÜNCÜ TARAF FİKRİ MÜLKİYETLERİN KORUNMASINA YÖNELİK KRİPTO MİMARİ VE FPGA GÜVENLİK MODÜLÜ TASARIMI" başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Prof. Dr. Sıddıka Berna ÖRS YALÇIN** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Prof. Dr. Mustafa ALTUN** İstanbul Teknik Üniversitesi	.....
	Prof. Dr. Umut Engin AYTEN Yıldız Teknik Üniversitesi	.....
		.....

Teslim Tarihi : **16 Ekim 2025**
Savunma Tarihi : **14 Ocak 2026**

Eşime ve çocuklarıma,

ÖNSÖZ

Yüksek lisans çalışmalarım kapsamında danışmanlığımı üstlenen, benimle tecrübe, bilgi ve deneyimlerini paylaşan değerli hocam Prof. Dr. Sıddıka Berna ÖRS YALÇIN'a sonsuz teşekkürlerimi sunarım.

Çalışmalarım sırasında bana destek olan iş arkadaşım Fatih TİRYAKİOĞLU'na teşekkür ederim.

Hayatımın her anında bana destek olan sevgili eşim Özgül GÖKTAŞ AÇAR'a, varlıklarıyla beni mutlu eden kızlarım Latife Bahar AÇAR, Betül AÇAR ve Begüm AÇAR'a sonsuz sevgilerimi sunarım.

Son olarak beni bu günlere getiren ve her konuda destek olan aileme teşekkür ederim.

Ocak 2026 Teyyar AÇAR

İÇİNDEKİLER

Sayfa

ÖNSÖZ ..... **ix**
İÇİNDEKİLER ..... **xi**
KISALTMALAR ..... **xiii**
SEMBOLLER ..... **xv**
ÇİZELGE LİSTESİ ..... **xvii**
ŞEKİL LİSTESİ ..... **xix**
ÖZET ..... **xxi**
SUMMARY ..... **xxiii**
1. GİRİŞ **1**
2. ÖN BİLGİLER VE LİTERATÜR TARAMA **5**
2.1 Problem Tanımı 5
2.2 ASIC Tümdevreler 5
2.2.1 ASIC yaşam döngüsü aşamaları, tehditler ve tehlikeler 5
2.2.2 ASIC IP için tehlikelere karşı önlem yöntemleri 8
2.3 FPGA ve FPGA IP 9
2.3.1 FPGA IP yaşam döngüsü aşamaları ve tehditler 9
2.3.2 FPGA IP koruma yöntemleri 10
2.3.3 3PIP için koruma yöntemleri 11
3. MANTIKSAL KİLİTLEME TEKNİĞİ VE SALDIRI YÖNTEMLERİ **13**
3.1 Mantıksal Kilitleme Tekniği 13
3.2 Mantıksal Kilitleme Tekniğine Yapılabilen Saldırı Yöntemleri 14
3.2.1 SAT saldırısı 15
3.2.2 SAT saldırısına dirençli mantıksal kilitleme tekniği çalışmaları 17
4. KRİPTO MİMARİDE KULLANILAN ALGORİTMALAR **19**
4.1 Anahtarlı HMAC Algoritması 19
4.2 SHA256 Özet Algoritması 20
4.3 Sayısal İmza Algoritması 20
4.3.1 RSA sayısal imza algoritması 21
5. ÖNERİLEN FPGA GÜVENLİK MODÜLÜ VE KRİPTO MİMARİ TASARIMI **27**
5.1 Önerilen Sistemin Genel Yapısı 27
5.1.1 3PIP tasarımı ve mantıksal kilitleme uygulanması 27
5.1.2 Sistemde uygulanacak kripto mimarinin tasarımı 28
5.1.3 FPGA güvenlik modülü tasarımı 30
5.2 Önerilen Sistemin Uygulama Adımları 32
5.2.1 3PIP tasarımcısının yapacağı işlemler 32
5.2.2 FPGA üreticisinin yapacağı işlemler 32
5.2.3 3PIP kullanıcısının yapacağı işlemler 32
5.3 Önerilen Sistemin Doğru Çalıştığının İspatlanması 33
5.3.1 Örnek 3PIP devresine mantıksal kilitleme uygulanması 33
5.3.2 P_{km} ve P_{ka} parametrelerinin elde edilmesi 34
5.3.3 Mantıksal kilitlenmiş devrenin çalıştırılması 36
5.4 Mantıksal Kilitleme Tekniğinin 3PIP Devrelere Etkisi ve LUL Modülünün Performans Özellikleri 39
5.4.1 Mantıksal kilitleme tekniğinin 3PIP devrelerin FPGA kaynak gereksinimlerine etkisinin incelenmesi 39

5.4.2 LUL modülünün FPGA kaynak gereksinimi ve 3PIP devrelere etkisinin incelenmesi 41
6. GÜVENLİK ANALİZİ VE KARŞILAŞTIRMA **45**
6.1 Sistemde Bulunan Kritik Varlıklar ve Tehdit Senaryoları 45
6.1.1 K_{FPGA} anahtarının elde edilme durumu 45
6.1.2 K anahtarının elde edilme durumu 46
6.1.2.1 Sahte 3PIP-L kullanım senaryosu 46
6.1.2.2 Öykünücü 3PIP-L kullanım senaryosu 46
6.1.2.3 Truva atı ve 3PIP-L içeren T-3PIP-L kullanım senaryosu 47
6.1.2.4 SAT saldırısı senaryosu 47
6.2 Önerilen Yöntemin Alternatif Yöntemle Karşılaştırılması 47
7. SONUÇLAR VE ÖNERİLER **51**
KAYNAKLAR **53**
ÖZGEÇMİŞ **57**

KISALTMALAR

3PIP : Üçüncü Taraf Fikri Mülkiyet
ASIC : Uygulamaya Özgü Entegre Devre
CA : Sertifika Yetkilisi
DSA : Sayısal İmza Algoritması
ECDSA : Eliptik Eğri Sayısal İmza Algoritması
EdDSA : Edwards Eğrisi Sayısal İmza Algoritması
EMSA : İmza Algoritmaları için Kodlama Yöntemi
FPGA : Alanda Programlanabilir Kapı Dizileri
FLL : Hata Tabanlı Mantıksal Kilitleme (Fault-based Logic Locking)
HDL : Donanım Tanımlama Dili
HLS : Yüksek Seviyeli Sentez
HMAC : Özet Tabanlı Mesaj Doğrulama Kodu
IDE : Tümleşik Geliştirme Ortamı
IEEE : Elektrik ve Elektronik Mühendisleri Enstitüsü
IETF : İnternet Mühendisliği Görev Gücü
IP : Fikri Mülkiyet
LUL : Logic Unlocker (Mantık Kilitleme Açıcı)
LUT : Look-Up Table
RLL : Rastgele Mantıksal Kilitleme (Random Logic Locking)
RFC : Yorum Talebi Belgesi (Request For Comments)
RSA : Rivest–Shamir–Adleman (Açık Anahtarlı Şifreleme Algoritması)
RTL : Kütük Transfer Seviyesi (Register Transfer Level)
SAR : SAT Tabanlı Saldırıya Dayanıklılık (Satisfiability-based Attack Resistance)
SAT : Karşılanabilirlik (Satisfiability)
SFLL : İşlevsellik Ayıklamalı Mantıksal Kilitleme (Stripped Functionality Logic Locking)
SLL : Güçlü Mantıksal Kilitleme (Strong Logic Locking)
SRAM : Statik Rastgele Erişimli Bellek (Static Random Access Memory)
T-3PIP-L : Truva atı ve 3PIP-L devresi içere saldırı devresi
VHDL : VHSIC Donanım Tanımlama Dili (Very High Speed Integrated Circuit HDL)
XNOR : Özel DEĞİL VEYA Mantık Kapısı (Exclusive NOR)
XOR : Özel VEYA Mantık Kapısı (Exclusive OR)

SEMBOLLER

3PIP : Orijinal 3PIP devresi
3PIP_L : Mantıksal kilitleme yapılmış 3PIP devresi
b : Özet fonksiyon blok büyüklüğü
(d,e) : RSA özel anahtarı
dP : **P**'nin CRT üs değeri
dQ : **Q**'nun CRT üs değeri
EM : Kodlanmış mesaj
emLen : EM'nin uzunluğu
$\mathbf{I_0}$: Başlangıç giriş vektörü
ipad : İç dolgu sabiti
K : Anahtar (HMAC, DSA ve Mantıksal kilitleme için ayrı)
$\mathbf{K_0}$: Başlangıç anahtarı
$\mathbf{K_{acik}}$: FPGA üreticisinin sisteme özel imza açık anahtarı
$\mathbf{K_{FPGA}}$: FPGA üreticisinin önerilen sistemde özel anahtarı
$\mathbf{K_{ozel}}$: FPGA üreticisinin sisteme özel imza özel anahtarı
k : Oktet cinsinden **s**'nin uzunluğu
M : Mesaj
m : Temsili mesaj değeri
modBits : **n**'in uzunluğu (bit cinsinden)
n : LUL modülü çalışma süresi işaret sayısı
(n,e) : RSA açık anahtarı
$\mathbf{O_{0n}}$: $n\times$ (3PIP çıkış büyüklüğü) uzunluğunda çıkış verisi
$\mathbf{O_{1n}}$: $n\times$ (3PIP çıkış büyüklüğü) uzunluğunda çıkış verisi
opad : Dış dolgu sabiti
(P,Q) : **n**'in asal çarpanları
Pka : 3PIP devresi doğrulama parametresi
Pkm : Anahtar maskeleme parametresi
qInv : CRT katsayısı
S : RSA ile elde edilen imza
s : Temsili imza değeri
$\mathbf{S_{3PIP\text{-}L}}$: 3PIP-L dosyasının imza değeri
T : HMAC ile elde edilen etiket

ÇİZELGE LİSTESİ

Sayfa

Çizelge 3.1: Farklı Düzeyde Uygulanan Mantıksal Kilitleme Özellikleri..... **13**
Çizelge 5.1: Performans Karşılaştırılmasında Kullanılan Benchmark Devrelerin İçerdiği Yapısal Eleman Sayıları. **39**
Çizelge 5.2: Test Edilen Benchmark Devrelerin Zamanlama Analizi Sonuçları... **40**
Çizelge 5.3: Test Edilen Benchmark Devrelerin Orijinal Durumu, Kilitli Durumu ve LUL Modülü ile Birlikte Kullanım Durumlarına Ait FPGA Donanım Kaynağı Gereksinimleri. **42**
Çizelge 5.4: Test Edilen Benchmark Devrelerin Hesaplanan Güç Tüketim Değerleri. **43**
Çizelge 6.1: Alternatif Yöntem ve Önerilen Yöntemin FPGA Donanım Kaynağı Gereksinimi Artış Oranları. **49**
Çizelge 6.2: Alternatif Yöntem ve Önerilen Yöntemin FPGA Üzerinde Çalışır Duruma Gelme Süreleri. **50**

ŞEKİL LİSTESİ

Sayfa

Şekil 2.1: Tasarımdan kullanıcıya kadar geleneksel ASIC üretim süreci **6**
Şekil 2.2: ASIC ürünler için yaşam döngüsü süreçlerinde tehdit unsurları **7**
Şekil 2.3: Klasik ve Modern FPGA IP tasarım akışları **10**
Şekil 2.4: Modern FPGA tasarım aşamalarında tehditler **10**
Şekil 2.5: Alternatif 3PIP koruma yöntemi **12**
Şekil 3.1: Kullanılan kapı türüne göre mantıksal kilitleme teknikleri **14**
Şekil 3.2: Mantıksal Kilitleme uygulamasının örnek devre üzerinde gösterimi **15**
Şekil 3.3: Mantıksal kilitli örnek devrenin farklı anahtar değerleri ile işlevsellik durumu **15**
Şekil 3.4: SAT saldırısının akış şeması **16**
Şekil 3.5: SAT öncesi ve sonrası mantıksal kilitleme çalışmaları ve saldırılar . **17**
Şekil 4.1: HMAC Algoritmasının işlevsel diyagramı **19**
Şekil 4.2: İmzalama ve imza doğrulama süreci akış şeması. **21**
Şekil 4.3: EMSA-PSS kodlama işlem akışı..... **24**
Şekil 5.1: 3PIP ve mantıksal kilitleme yapılmış 3PIP şematik gösterimi. **28**
Şekil 5.2: LUL Modülü yapısının şematik gösterimi..... **31**
Şekil 5.3: 3PIP-L 'nin LUL modülü ile birlikte kullanımı. **33**
Şekil 5.4: b01.vhd ve b01_enc.vhd devrelerinin test edilmesi. **35**
Şekil 5.5: LUL Modülünde b01enc devresinin doğru anahtar elde edilerek çalışması..... **37**
Şekil 5.6: LUL Modülünde b01enc devresinin yanlış anahtar elde edilerek durdurulması. **38**

MANTIKSAL KİLİTLEME TEKNİĞİ İLE ÜÇÜNCÜ TARAF FİKRİ MÜLKİYETLERİN KORUNMASINA YÖNELİK KRİPTO MİMARİ VE FPGA GÜVENLİK MODÜLÜ TASARIMI

ÖZET

Günümüzde donanım tasarımları kullanıcıya sunduğu kolaylıklar, çok çeşitli kapasiteleri ve ucuz maliyetleri nedeniyle Alanda Programlanabilir Kapı Dizileri (Field Programmable Gate Arrays – FPGA) üzerinde gerçekleştirilmektedir. FPGA tasarımcıları tasarımlarının fikri mülkiyet hakkını (Intellectual Property - IP) korumak amacıyla genellikle şifreleme veya gizleme teknikleri kullanmaktadır. Birçok FPGA modeli üzerindeki konfigürasyonu korumak amaçlı bir sistem mimarisine sahiptir.

FPGA geliştirme ortamında tasarımcı donanım tanımlama dili (hardware definition language - HDL) ve hazır kütüphaneler kullanarak gerçekleştirdiği IP'sini sentezleyerek FPGA üzerine yüklenebilecek hale getirir. Sentezleme sırasında tasarım önce netlist biçimine, ardından bitstream dosyası biçimine çevrilir. Tersine mühendislik yoluyla bitstream dosyası formatından netliste ve ardından tasarımın orijinal haline erişilebilmektedir. FPGA geliştirme aracı bitstream dosyasını oluştururken şifreleme seçeneği kullanılırsa bitstream dosyası şifreli olarak üretilir. Bitstream dosyasının şifrelenmesi önemli ölçüde güvenlik sağlamaktadır. FPGA üreticileri bitstream dosyasının elde edilebilmesine yönelik saldırı ve önlemlerle ilgili gelişmeleri takip ederek yeni güvenlik önlemlerini sistemlerine dahil etmektedir.

FPGA ile gerçekleştirilmiş çözümlerin yaygınlaşması ile birlikte tasarımcılar IP'lerini oluştururken iş gücü ve zaman maliyetlerini düşürmek amacıyla tasarımın bazı özel tanımlı kısımlarını hazır IP olarak kullanmak istemektedir. Bazı işlemler için geliştirilmiş IP'ler hazır olarak FPGA geliştirme aracı ile sunulurken, bazı IP'lerin lisanslı olarak dış kaynaktan tedarik edilmesi gerekebilir. Dış kaynaktan tedarik edilmesi durumunda üçüncü bir taraf söz konusu olduğu için bu hazır IP'ler üçüncü taraf fikri mülkiyet hakkı (Third Party Intellectual Property - 3PIP) olarak anılmaktadır. Bazı 3PIP'ler FPGA üreticisi desteği ile geliştirme aracı ile katalog olarak sunulurken, bazısı doğrudan tasarımcı tarafından kullanıcıya sunulmaktadır. 3PIP'lerinin şifrelenmesi ve yönetimi için IEEE tarafından yayınlanmış tavsiye edilen uygulama standardı bulunmaktadır. FPGA üreticileri de bu standarda uymaktadır. 3PIP'lerin telif haklarını korumak için şifreleme yöntemi veya ayrık donanım bazlı çözümler kullanılabilir. Ayrık bir donanım kullanılması 3PIP için maliyeti artırırken aynı zamanda 3PIP kullanıcısının tasarımını zorlaştırmakta ve PCB maliyetini de yükseltmektedir. 3PIP şifreli olsa bile 3PIP içeren FPGA tasarımı sentezlenip bitstream dosyası üretilirken şifrelenmiş IP orijinal haline dönmektedir. Şifrelenmeden üretilmiş bitstream dosyasına tersine mühendislik yapılarak 3PIP tasarımının orijinal hali elde edilebilir. 3PIP tasarımının orijinal haline FPGA üzerine yüklendikten sonra gelmesi sağlanabilirse 3PIP tasarımı için güvenli bir koruma sağlanmış olacaktır.

Benzer problemler uygulamaya özgü tümleşik devreler (Application Specific Integrated Circuit - ASIC) içinde söz konusu olup farklı önlem yöntemleri geliştirilmiştir. Bu önlemlerden mantıksal kilitleme tekniği ASIC ürünlerin orijinal devre yapısına ilave anahtar girişleri ve mantık kapıları ekleyerek tasarım sonrası süreçlerde ASIC IP'sinin korunmasını sağlamaktadır. ASIC devre doğru anahtar değeri güvenli belleğe yüklenerek son kullanıcı aşamasında orijinal tasarımı gibi çalışmaktadır.

Bu çalışmada amaç ASIC IP tasarımları için kullanılan mantıksal kilitleme tekniğinin 3PIP'lerde kullanılması sağlayacak bir sistem tasarımı yapmaktır. Bu amaçla FPGA üreticilerinin FPGA modellerine ekleyebilecekleri bir güvenlik modülü ve uygulanacak kripto mimari tasarımı yapılmıştır. 3PIP orijinal netlist yapısına mantıksal kilitleme uygulayarak ilave anahtar girişleri ve mantık kapıları eklenecektir. Kilitlenmiş 3PIP devresine ait doğru anahtar değeri FPGA içinde bulunan güvenlik modülü yardımıyla kripto mimari adımlar uygulanarak elde edilecek ve 3PIP orijinal haline dönerek çalışacaktır. Önerilen sistemde FPGA üreticileri güvenilir çözüm ortaklarıdır. Bu sistem uygulandığında 3PIP sadece FPGA üzerinde orijinal halinde bulunacak, güvenilmeyen taraflara karşı korunmuş olacaktır.

DESIGN OF A CRYPTOGRAPHIC ARCHITECTURE AND FPGA SECURITY MODULE FOR PROTECTING THIRD-PARTY INTELLECTUAL PROPERTY USING THE LOGIC LOCKING TECHNIQUE

SUMMARY

Today, hardware designs are predominantly implemented on Field Programmable Gate Arrays (FPGAs) due to their ease of use, broad range of capabilities, and cost-effectiveness. In order to safeguard the intellectual property (IP) rights of their designs, FPGA developers commonly employ encryption or obfuscation techniques. Many FPGA platforms are equipped with architectural mechanisms specifically designed to protect configuration data.

A designer utilizing an FPGA development tool synthesizes the created Intellectual Property (IP) using a Hardware Description Language (HDL) and predefined libraries, thereby making it deployable on the FPGA. During the synthesis process, the design is initially converted into a netlist and subsequently transformed into a bitstream file. Through reverse engineering techniques, it is possible to extract the netlist from the bitstream format and ultimately reconstruct the original design. If the encryption option is enabled during bitstream generation, the resulting bitstream file is produced in an encrypted form. Encrypting the bitstream file significantly enhances security. FPGA manufacturers closely monitor developments related to attacks and countermeasures aimed at unauthorized bitstream access, and continuously integrate new security mechanisms into their systems.

With the increasing adoption of FPGA-based solutions, designers often seek to reduce labor and time costs by utilizing pre-designed Intellectual Property (IP) cores for specific parts of their designs. While certain IP cores are readily available through FPGA development tools, others may need to be licensed from external sources. When such cores are sourced externally, they are referred to as Third Party Intellectual Property (3PIP). Some 3PIPs are provided within the toolchain by FPGA vendors as part of an IP catalog, whereas others are delivered directly by independent designers to end users. To address the encryption and management of electronic design 3PIPs, the IEEE has issued a recommended practice standard, which FPGA vendors also adhere to. In order to protect the intellectual property rights of 3PIPs, either encryption techniques or discrete hardware-based solutions may be employed. However, using discrete hardware increases the cost of the 3PIP and complicates the design process for users, in addition to raising the cost of the printed circuit board (PCB). Even when a 3PIP is encrypted, the original IP may be reconstructed during synthesis and bitstream generation. If the bitstream is not encrypted, reverse engineering can be used to retrieve the original 3PIP design. Ensuring that the original design is only recovered after it has been loaded onto the FPGA would provide a secure means of protecting the 3PIP.

In the scope of this study, the feasibility of unlocking a 3PIP on an FPGA using a key has been investigated. It was observed that this can be achieved through the application of logic locking techniques, which are commonly employed in Application-Specific Integrated Circuit (ASIC) products. Similar security challenges also arise in ASIC designs, prompting the development of various countermeasures, with research efforts in this area still ongoing. Logic locking enhances the security of ASIC intellectual property by incorporating supplementary key inputs and logic gates into the original circuit design, ensuring protection in the post-design stages. After fabrication and testing, the correct key value is loaded by the designer to ensure that the circuit functions as intended at the end-user stage. Various attack methodologies targeting logic locking have been studied, and corresponding countermeasures resistant to these attacks have been proposed. Chapter 2 provides a detailed analysis of ASIC and FPGA IP design processes, associated threats, and mitigation strategies.

This thesis aims to adapt the logic locking technique, originally developed for ASIC IP designs, for secure use in 3PIP within FPGA environments. To achieve this goal, a cryptographic architecture and a dedicated security module capable of performing the necessary cryptographic operations have been designed. The proposed system is intended to function as an integrated component of the FPGA security architecture. Ownership of the designed system is attributed to FPGA vendors, who are expected to incorporate the security module as a feature in future FPGA models. In this framework, both 3PIP designers and end-users will utilize the system, while FPGA vendors are assumed to be trusted parties with respect to the protection of 3PIP designs.

Within the scope of this study, the proposed cryptographic architecture incorporates the use of Keyed HMAC-SHA256 and RSA Digital Signature algorithms. The Keyed HMAC algorithm is employed to derive the correct key for the locked 3PIP and to verify its authenticity. On the other hand, the RSA digital signature algorithm is utilized to ensure the integrity and authenticity of the key used to lock the 3PIP, by verifying it prior to the synthesis process within the FPGA design tool. Chapter 4 provides detailed descriptions of the Keyed HMAC and RSA signature algorithms used in the system.

Chapter 5 presents the design of the cryptographic architecture and the associated security module. It describes how the 3PIP designer applies logic locking to the original netlist structure of the 3PIP by inserting additional key inputs and logic gates. Subsequently, the cryptographic steps carried out by the FPGA manufacturer to generate the necessary parameters are outlined. Finally, the procedures for the 3PIP user to utilize the security module and the locked 3PIP are explained. The correct key required to unlock the locked 3PIP circuit is derived through the cryptographic operations performed by the on-chip security module within the FPGA. Once unlocked, the 3PIP will operate in its original form. This approach ensures that the original form of the 3PIP remains confined to the FPGA and is protected from untrusted entities. In addition, this section examines the impact of the logic locking technique on 3PIP circuits and the costs introduced by the LUL module.

Chapter 6 presents the security analysis of the proposed system, including an evaluation of potential threats and attack scenarios. A comparative analysis is conducted against existing studies in the literature that aim to protect 3PIP designs.

Subsequently, the validation procedures of the proposed system are described. In this context, the security module was implemented using AMD's Vivado tool and the VHDL language. The ITC99 benchmark circuits were utilized as representative 3PIPs. Logic locking was applied to the selected benchmark circuits using the Neos tool. After verifying the benchmark circuits through simulation, they were integrated with the developed security module, and the correct functionality of the system was confirmed via simulation.

The conclusion chapter discusses the advantages and disadvantages of the proposed system and outlines potential improvements for future work.

1. GİRİŞ

Fikri mülkiyet hakkı (Intellectual Property - IP) ürünlerin içerdiği fikir ve tasarım bilgilerinin üzerindeki mülkiyet haklarını ifade etmektedir. Yazılım ve donanım ürünlerinde bu haklar telif hakları, patentler, tasarım hakları, ticari sırlar ve marka hakları gibi haklardan oluşur.

Donanım ürünlerinde IP denildiğinde uygulamaya özgü tümleşik devreler (Application Specific Integrated Circuit - ASIC) ya da sahada programlanabilir kapı dizileri (Field Programmable Gate Arrays - FPGA) için donanım tanımlama dili (Hardware Definition Language - HDL) kullanılarak hazırlanan tasarım bilgileri ifade edilmektedir [1]. ASIC çipler uygulamaya özgü olarak geliştirilir ve sadece tanımlanmış işlemleri gerçekleştirler. Güç tüketimi, zaman ve hacim olarak avantajlar içerir. FPGA ler ise tekrar programlanabilmeleri ve farklı işlemleri gerçekleyebilecek yapıda olmaları sebebiyle geliştiricilere donanım seviyesinde tasarımlarını hayata geçirebilme imkanı sunmaktadır. ASIC çipler için korunması gereken IP çipin HDL tasarım bilgisi veya üretime hazır yerleşim düzeni iken FPGA kullanılarak tasarım gerçeklenmesi durumunda IP FPGA de çalışacak HDL tasarımıdır.

FPGA ile gerçeklenmiş çözümlerin yaygınlaşması ile birlikte tasarımcılar IP'lerini oluştururken işgücü ve zaman maliyetlerini düşürmek amacıyla tasarımın bazı özel tanımlı kısımlarını hazır IP olarak kullanmak istemektedir. Bazı işlemler için geliştirilmiş IP'ler hazır olarak FPGA geliştirme aracı ile sunulurken, bazı IP'lerin lisanslı olarak dış kaynaktan tedarik edilmesi gerekebilir. Dış kaynaktan tedarik edilmesi durumunda üçüncü bir taraf söz konusu olduğu için bu IP'ler üçüncü taraf fikri mülkiyet hakkı (Third Party Intellectual Property-3PIP) olarak isimlendirilir [2].

Donanım tasarım IP'lerinin şifrelenmesi ve yönetimi için IEEE tarafından yayınlanmış tavsiye edilen uygulama standardı bulunmaktadır [3]. FPGA üreticileri de bu standarda uymaktadır. FPGA tasarımcıları tasarımlarının fikri mülkiyet hakkını (Intellectual Property - IP) korumak amacıyla genellikle şifreleme veya gizleme teknikleri

kullanmaktadır. Birçok FPGA modeli üzerindeki konfigürasyonu korumak amaçlı bir sistem mimarisine sahiptir. Sistem mimarileri şifreleme, özet alma algoritmaları gibi kriptografik çekirdekler içermektedir. FPGA geliştirme ortamında tasarımcı donanım tanımlama dili (hardware definition language - HDL) ve hazır kütüphaneler kullanarak gerçekleştirdiği IP'sini sentezleyerek FPGA üzerine yüklenebilecek hale getirir. Sentezleme sırasında tasarım önce netlist biçimine, ardından bit dosyası (bitstream) biçimine çevrilir. Tersine mühendislik yoluyla bit dosyası formatından netliste ve ardından tasarımın orijinal haline erişilebilmektedir [4]. FPGA geliştirme aracı bit dosyasını oluştururken şifreleme seçeneği kullanılırsa bit dosyası şifreli olarak üretilir. Bit dosyasının şifrelenmesi önemli ölçüde güvenlik sağlamaktadır. FPGA üreticileri bit dosyasının elde edilebilmesine yönelik saldırı ve önlemlerle ilgili gelişmeleri takip ederek yeni güvenlik önlemlerini sistemlerine dahil etmektedir [5].

3PIP'lerin telif haklarını korumak için şifreleme yöntemi veya ayrık donanım bazlı çözümler kullanılabilir. Ayrık bir donanım kullanılması 3PIP maliyetini artırırken aynı zamanda 3PIP kullanıcısının tasarımını zorlaştırır ve PCB maliyetini de artırır.

Şifrelenmiş 3PIP orijinal haline FPGA'e yüklenmeden önce gelirse tasarım korunması tam olarak gerçekleşmeyecektir. FPGA'ler tarafından sağlanan bit dosyasının şifrelenmesi ve tasarımın FPGA üzerinde orijinal haline gelmesi gibi bir çözüm bulunabilirse 3PIP telif hakları için de güvenli bir koruma sağlanmış olacaktır.

Benzer problemler tümdevre tasarımları içinde söz konusudur. Tümdevre üretiminde çip üreticisi, çip test ekibi gibi tümdevrenin üretim aşamasında yer alan bileşenler güvenilmez olarak kabul edilmektedir [6]. Tümdevre tasarımının korunması amacıyla yapılan çalışmalarda problem çözümü için son zamanlarda önerilen çözümlerden biri Mantıksal Kilitleme (Logic Locking) tekniğidir [7].

Bu tez çalışmasında 3PIP leri mantıksal kilitleme tekniği ile kilitleyerek FPGA içerisinde orijinal 3PIP yapısına ulaşmasını sağlayacak bir FPGA güvenlik özelliği ve kripto mimari tasarımı yapılmıştır.

Literatür taramalarında doğrudan 3PIP 'lerin tasarımının korunmasını hedefleyen sadece bir çalışmaya rastlanmıştır [7].

Tez içinde 2. bölümde donanım IP lerin tasarımdan son kullanıcı aşamasına kadar olan süreçleri ve IP ler ve tasarımcılar için tehdit ve tehlikeler tanımlanacak, ardından önlemler ve bu alanda yapılan çalışmalardan bahsedilecektir. 3. bölümde tasarımda kullanılacak teknik ve algoritmalar tanıtılacaktır. 4. bölümde tasarım detayları ve kullanım işlemleri anlatılacaktır. 5. bölümde tasarım gerçeklemesi ve tasarımın sağladığı güvenlik [7]'de önerilen yöntemle karşılaştırılarak anlatılacaktır.

2. ÖN BİLGİLER VE LİTERATÜR TARAMA

2.1 Problem Tanımı

Elektronik ürünler hayatın her alanında kullanılırken, teknolojideki gelişmelerle ürünler daha az yer kaplamakta, daha az güç tüketmekte ve daha hızlı çalışabilmektedir. Elektronik ürünlere bu özellikleri sağlayan temel bileşenleri tümleşik devrelerdir. Tümleşik devreler ASIC olarak sabit fonksiyonlu üretilebildiği gibi FPGA olarak tekrar programlanabilen biçimde de üretilebilir. Her ikisi de donanım IP olarak tanımlanabilir.

Teknolojik gelişmeler ve gelişen saldırı yöntemleri nedeniyle donanım IP'lerin korunması zorlaşmaktadır. ASIC ve FPGA IP leri için yeni koruma teknikleri çalışmalarıda paralel olarak ilerlemektedir. Bu çalışmada donanım IP'leri için tehditler ve tehlikeler incelenerek FPGA IP'si tasarımlarında önemli yer tutan 3PIP'lerin korunması problemi için bir güvenlik çözümü geliştirilmesi hedeflenmiştir.

Donanım IP ürünlerini ASIC ürünler ve FPGA IP ürünleri olarak iki başlık altında incelenecektir. Ürünlerin tasarımdan son kullanıcıya kadar olan yaşam döngüsü aşamaları ve bu aşamalarda IP ler için tehdit unsurları, tehlikeler ve önlem yöntemlerine değinilecektir.

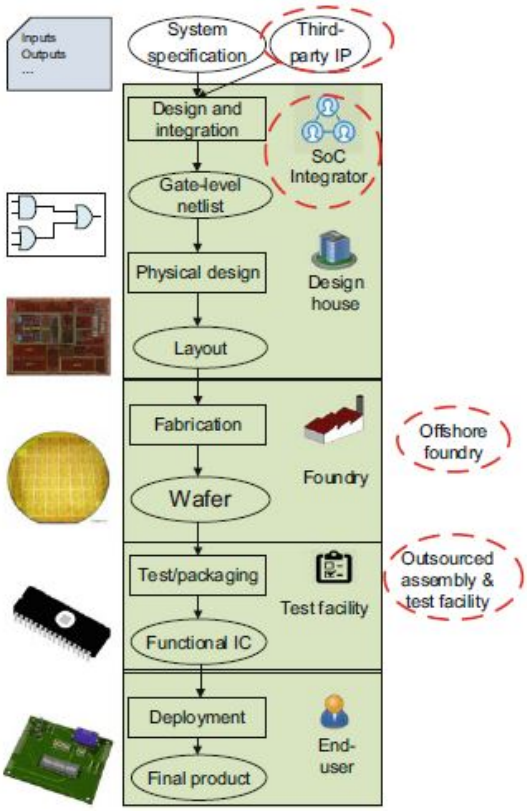
2.2 ASIC Tümdevreler

ASIC tümdevreler uygulamaya özel geliştirilmiş tümleşik devreler olduğundan HDL tasarımı yanı sıra üretime hazır yerleşim düzeni de korunması gereken IP olarak tanımlanabilir.

2.2.1 ASIC yaşam döngüsü aşamaları, tehditler ve tehlikeler

ASIC ürünler için yaşam döngüsü tasarım, üretim, test ve son kullanıcı olarak tanımlanmıştır [8]. Geleneksel ASIC tasarım üretim süreçleri Şekil 2.1'de

gösterilmiştir. Tasarım aşaması IP sahibi tarafından yapılmaktadır. Bilgisayar ortamında gelişmiş tasarım ve benzetim programları yardımıyla tasarım yapmak kolay ve düşük maliyetli olarak gerçekleştirilebilmektedir. Ancak üretim aşaması özel teknolojik cihazlar gerektirmektedir. Üretim tesisi kurmak çok yüksek maliyetli olup uluslararası yaptırımlar gibi sebeblerle ilave zorluklarda bulunmaktadır. Üretimin hassas ve zor olması nedeniyle üretilen her ASIC ürün doğru çalışmamaktadır. Bu nedenle üretim sonrası bir test aşaması bulunmaktadır. Test aşaması da zaman ve test cihazları nedeniyle bir maliyet oluşturmaktadır. Test edildikten sonra ASIC ürün kullanım için hazırdır. ASIC ürün kullanılacağı cihaz ya da ürün üzerine takıldıktan sonra son kullanıcı aşaması başlamaktadır.

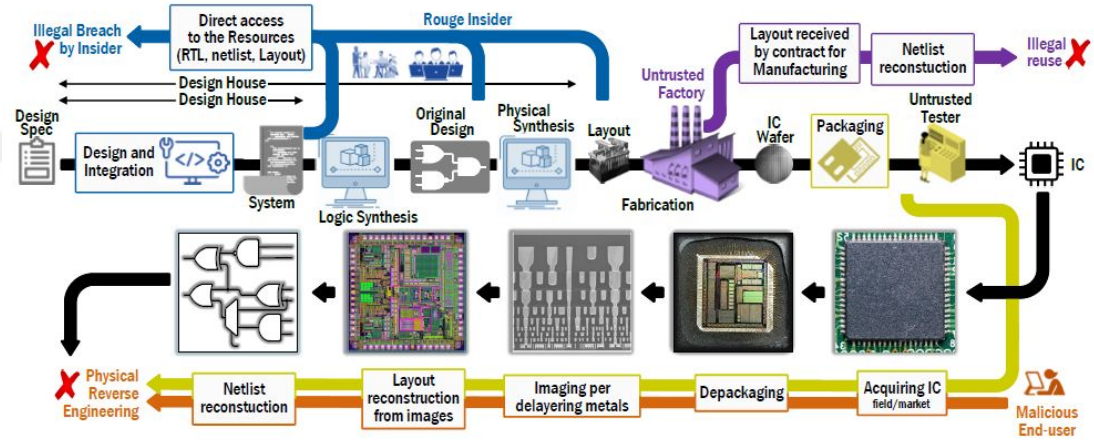


Şekil 2.1: Tasarımdan kullanıcıya kadar geleneksel ASIC üretim süreci [8].

Üretim aşaması için yatırım yapmak maliyetli olması nedeniyle tasarımcılar üretim aşamasını dış kaynak kullanarak çip üretim tesislerine yaptırmayı tercih etmektedir. Test aşamasında da üretim planlaması, lojistik faaliyetlerin planlanması, ilave maliyet ve özellikle zaman yönetimi gibi sebeplerle dış kaynak kullanımı tercih edilmektedir

[9]. Test aşaması üretici firma tarafından yapılabileceği gibi ayrı test merkezlerinde de yapılabilmektedir.

Üretim aşaması ve test aşamasının dış kaynak kullanıldığı durumlar da kapsanarak ASIC ürünler için tehdit unsurları [6] 'te belirtilmiştir. Şekil 2.2'den anlaşılacağı üzere tasarımcı dışında ki tüm bileşenler güvenilmez kabul edilmekte ve tehdit unsuru olabilmektedir. Tasarım geliştirilirken kullanılan araçlar, çalışanlar, dış kaynak üretim tesisi, dış kaynak test merkezi ve son kullanıcı adımında bu tehlikeler gerçekleşebilir. Tehlikeler hem ASIC tasarımcısı için hem de son kullanıcı için geçerli olabilmektedir.



Şekil 2.2: ASIC ürünler için yaşam döngüsü süreçlerinde tehdit unsurları [6].

Geliştirmede kullanılan araçlar veya iç saldırganlar tarafından IP tasarımına donanımsal eklenti (truva atı vb.) yapması ya da internet ortamına erişimi varsa IP tasarım bilgilerinin çalınmasına sebeb olması mümkün olabilir. IP tasarımına yapılacak eklenti son kullanıcı verileri için tehlike oluştururken, bu durumun ortaya çıkması ASIC tasarımcısı için itibar kaybı, ticari kayıp ve ceza durumu gibi tehlikeler oluşturabilir.

Üretim aşaması için üretime hazır ASIC yerleşim düzeni üreticiye teslim edilmelidir. Yerleşim düzeni yukarıda da ifade edildiği gibi IP tasarımını içermektedir. Yerleşim düzeninden HDL tasarım bilgisi elde edilebilir [6]. Üretici değişiklik yaparak ASIC içine truva atı yerleştirebilir. Bu durum geliştirme araçlarının eklenti yapması ile aynı tehlikeleri oluşturmaktadır. Bunun yanında üretici kendisinden istenen miktardan fazla üretim yapabilir. ASIC ürün ticari bir ürün olduğu durumda fazladan ürettiklerini satarak ASIC tasarımcısının pazar kaybına ve maddi kaybına sebep olabilir.

Test aşamasında ASIC ürünler tüm işlevsellikleri doğru yapıp yapmadıkları belirlenecek şekilde test edilmektedir. Bunun için tüm işlevleri test edecek test vektörleri hazırlanmakta ve tarama zincirleri kullanarak ASIC ürüne uygulanmaktadır. Test vektörlerinden ve elde edilen çıktılardan yararlanılarak IP tasarımının yapısı elde edilebilir. Elde edilen tasarım bilgisi ile izinsiz üretim yapılarak tasarımcının kayıplarına sebep olabilir.

Son kullanıcı aşamasında ki ASIC ürün için test aşamasında gerçekleştirilen işlemler uygulanarak veya tersine mühendilik yapılarak benzer tehlikelerin oluşmasına sebep olabilir. Tersine mühendislik yapılması IC paketini açma, katmanları ayırma ve katmanları görüntüleyip bu görüntüleri analiz ederek netlist detaylarının belirlenmesi aşamalarından oluşmaktadır [10].

2.2.2 ASIC IP için tehlikelere karşı önlem yöntemleri

Yukarıda belirtilen tüm tehdit unsurları ve tehlikeler dikkate alınarak ASIC IP sahipleri çeşitli güvenli tasarım (desing-for-trust) teknikleri geliştirmekte ve kullanmaktadır [6]. Bu teknikler filigranlama (watermarking), parmak izi oluşturma (fingerprinting), ölçümleme (metering), yonga gizleme (camouflaging), üretimin bölünmesi (split manufacturing) ve donanım gizleme gibi yöntemleri içermektedir.

Filigranlama ve parmak izi oluşturma teknikleri tasarım içerisine gizli bir iz gömülerek korsancılığın tespit ve takibini yapabilmeyi sağlarlar. Her ikisi de pasif teknikler olup engelleme özelliği içermezler [11] [12].

Yonga gizleme (IC camouflaging) tekniği son kullanıcı aşamasında ki ürün üzerinde yapılabilecek tersine mühendislik saldırılarını engellemeyi amaçlamaktadır [13]. Tasarımda farklı işlevselliğe sahip benzer yapıda mantık kapıları kullanılır. Ancak bu yöntem üretim sonrası tersine mühendislik saldırılarına karşı etkili iken üretim aşamasında ki tehditlere engel olamamaktadır.

Üretimin bölünmesi yöntemi yonganın bazı katmanlarını güvenilmeyen (genellikle yüksek teknolojili) fabrikalarda yaptırırken diğer katmanları güvenilen fabrikalara yaptırmayı önermektedir. Ancak bu durumda güvenilmeyen üretim yeri kaynaklı risk azalırken, son kullanıcı tarafında veya iç tehditlere karşı koruma sağlamamaktadır [14].

Ölçümleme tekniğinde her ASIC ürüne tekil bir kimlik bilgisi vererek üretim sonrası ürünlerin takibini sağlar. Kullanılan teknik pasif ise sadece korsanlığı tespit edebilirken, aktif ölçümleme teknikleri ile sahada ASIC ürünün takibi ve kontrol edilmesi mümkün olabilir [8].

Donanım gizleme yöntemi IP tasarımına tasarımın parçası imiş gibi davranan ilave kapılar ekleyerek IP tasarımının anlaşılmasını engellemeyi hedefler. Donanım gizleme yöntemi örneklerinden en yaygın olanı mantıksal kilitleme tekniğidir. Mantıksal kilitleme tekniği tasarıma ilave anahtar girişleri ve mantık kapıları ekleyerek devrenin işlevsel yapısını değiştirmeyi amaçlar. Uygulama dikkatle yapılırsa tüm aşmalarda yukarıda anlatılan tehditlere karşı güvenlik sağlanabilir [6]. Mantıksal kilitleme yöntemi detaylıca ileride incelenecektir.

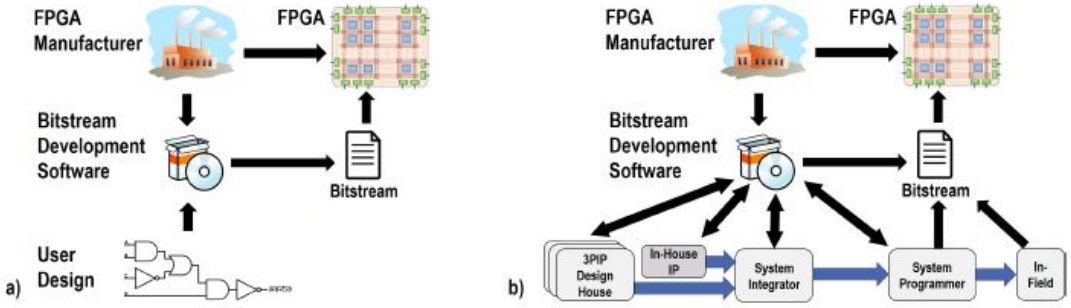
2.3 FPGA ve FPGA IP

FPGA tekrar programlanabilen bir tümleşik devredir. Programlanabilme özelliklerine göre FPGA'ler üçe ayrılır: Tek sefer programlanabilen (One Time Programmable - OTP) FPGA, flash tabanlı FPGA'ler ve SRAM tabanlı FPGA'ler. FPGA IP'si FPGA üzerinde çalışacak sayısal tasarımdır. OTP FPGA'ler bir kez programlandıktan sonra sürekli aynı tasarıma sahip olacağı için ASIC ürünlere benzesede üretim ve test süreçleri diğer FPGA'lerle aynıdır. Flash tabalı FPGA'ler programlandığında IP tasarımı FPGA üzerine kazılı hale gelmekte, tekrar başka bir IP tasarımı yüklenene kadar değişmemektedir. SRAM tabanlı FPGA'ler ise her açıldığında IP tasarımı harici bir bellekten FPGA üzerine tekrar yazılmaktadır.

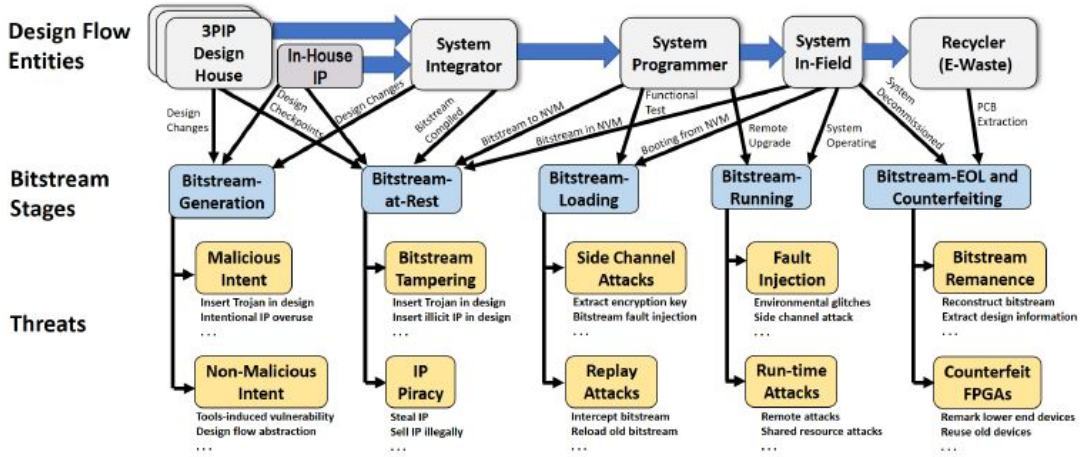
2.3.1 FPGA IP yaşam döngüsü aşamaları ve tehditler

FPGA IP tasarımı ilk başlarda bir tasarımcının FPGA geliştirme aracı (Integrated Development Environment - IDE) kullanarak hazırladıkları IP'nin FPGA'e yüklenmesi aşamalarından oluşurken, günümüzde maliyet ve hızlı tasarım gerekçesiyle tasarımın bazı parçalarını 3PIP olarak temin etmeyi gerektirmektedir [5]. Şekil 2.3'te FPGA tasarım ve işlem akış süreçleri gösterilmektedir. Sistem tümleştirici üçüncü taraftan aldığı 3PIP'leri kendi tasarımı ile birleştirerek FPGA IP'sini (bitstream dosyası) oluşturmaktadır. 3PIP ve FPGA IP tasarımcıları kendileri için güvenilir olacaktır.

Ancak kendilerinin dışındaki aşamalarda tehditlerle karşılaşacaklardır. [5]'de tasarım sürecinde yer alan aktörler ve bitstream dosyasının aşamaları arasındaki etkileşim ile bu aşamalarda gerçekleşebilecek tehditler Şekil 2.4'deki gibi sınıflandırılmıştır.



Şekil 2.3: a. Klasik FPGA IP tasarım akışı b. Modern FPGA IP tasarım akışı [5].



Şekil 2.4: Modern FPGA tasarım aşamalarında tehditler [5].

FPGA bitstream oluşturma sürecinde kötü niyetle zararlı kod (truva atı) ekleme yapılabilir. Bu saldırı [15]'de olduğu gibi sentezleme sonrasında eklenebileceği gibi tasarım süreci içinde iç saldırgan ya da 3PIP tasarımı üzerinden gerçekleşebilir.

FPGA IP için korsanlık şu biçimlerde gerçekleşebilir: IP'nin aşırı kullanımı IP hırsızlığı, IP'nin yeniden kullanımı ve IP'ye tersine mühendislik yapılarak. Tersine mühendislik saldırısı yapılarak orijinal netlist bitstream dosyasından elde edilebilmektedir [4].

2.3.2 FPGA IP koruma yöntemleri

Flash tabanlı FPGA'lerde bitstream dosyası IDE ortamında şifreli üretilebilir ve yüklenmeden önce şifreli olarak bulunabilir. Saha aşamasında ise FPGA üzerinde

olup FPGA güvenlik özellikleri ile korunmuş olmaktadır. SRAM tabanlı FPGA'lerde bitstream dosyası IDE ortamında şifreli üretilebilir ve yüklenebilir. Ancak bitstream dosyası bir bellek üzerinde bulunur ve FPGA her açıldığında şifresi çözülerek yüklenir. Bu nedenle SRAM tabanlı FPGA'lerde yan kanal analizi riski olabilir.

Literatür taraması kapsamında, [16] çalışmasında FPGA üzerinde fikri mülkiyetin (IP) korunmasına yönelik olarak mantıksal kilitleme tekniğinin kullanıldığı bir yaklaşıma rastlanmıştır. Söz konusu çalışmada, makine öğrenmesinde hızlandırıcı uygulamaları içeren IP'ler için mantıksal kilitleme uygulanmış ve bu tekniğin IP üzerindeki etkileri ayrıntılı biçimde incelenmiştir. Ancak çalışma, mantıksal kilitlemenin IP koruması amacıyla uygulanabilirliğini ortaya koymakla sınırlı kalmakta; kullanılan kilitleme anahtarının yönetimi ve güvenliği konusunda herhangi bir mekanizma sunmamaktadır. Kilitli IP'yi içeren bitstream dosyası, FPGA'nın sunduğu bitstream şifreleme hizmeti ile korunmaktadır. Mantıksal kilitleme anahtarının bitstream dosyası içerisine gömülü (kazılı) olarak yerleştirilmiş olması, anahtarın donanım düzeyinde bağımsız bir güvenlik katmanı ile korunmadığını göstermektedir. Bu durum, önerilen yaklaşımın 3PIP koruması açısından bir çözüm sunmasını engellemektedir.

2.3.3 3PIP için koruma yöntemleri

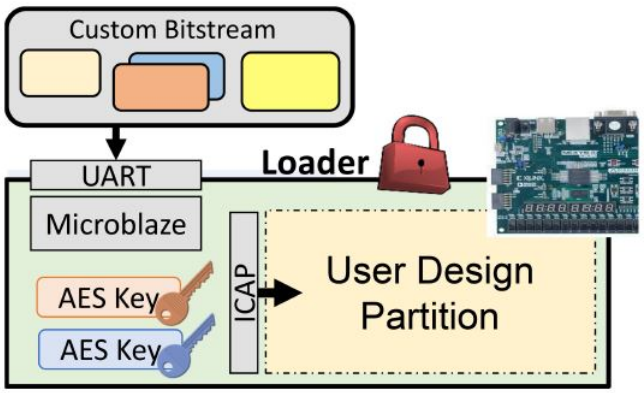
3PIP'ler genellikle FPGA IP'leri için güvenilmez bileşen olduğundan tehdit olarak görülürler. Savunma uygulamaları gibi hassas sektörlerde 3PIP'lerin tasaımın açık ve anlaşılabilir şekilde olması beklenir. ABD Savunma Bakanlığı üç farklı güvence seviyesinde 3PIP gözden geçirme süreci belirlemiştir [17] [18] [19].

3PIP her ne kadar tehdit görülse de içerdiği IP haklarının korunması özellikle ticari sebeblerle gerekmektedir. 3PIP FPGA IP'si içinde şifrelenerek başka taraflara karşı korunmuş olabilir. Ancak FPGA IP tasarımcısı yani 3PIP kullanıcısına karşı bu durum geçerli değildir.

Mevcut koruma yönteminde 3PIP dosyası şifreli olarak ve/veya lisanslama kullanılarak genellikle FPGA üreticileri ile birlikte kullanıcılara sunulmaktadır. Bunun dışında tasarımcılar tarafından doğrudan kullanıcıya da teslim edilebilmektedir. Ayrık

donanım kullanılarak (dongle) koruma sağlamakta mümkün olabilir. Ancak bu durumda hem maliyet artacak hem de kullanıcıya ilave yük getirecektir.

Bu konuda yapılan literatür araştırmalarında 3PIP'nin korunmasına yönelik bir adet çalışmaya rastlanmıştır [7]. Çalışmada 3PIP koruması için sentez sonrası 3PIP nin kullandığı LUT lar belirlenip LUT tipleri kaydedilip başlangıç değerleri alınarak LUT tipi ve başlangıç değerleri değiştirilerek yeni bir bitstream dosyası oluşturulmaktadır. LUT tipleri ve başlangıç değerleri bir dosyaya yazılarak özel bir anahtarla şifrelenmektedir. Çalışmada bir Yükleyici modül tasarımı yapılmıştır. Önerilen kapsamda Yükleyici modül ve özel anahtar(lar) "Güvenilir Anahtar Tutucu" tarafından FPGA'lere daha önce yüklenmelidir. Bu ön yükleme işleminden anlaşılacağı gibi çözüm için "Kısmi Yeniden Yüklenebilir" özellikli FPGA kullanılması gerekmektedir. Ön yükleme yapılmış FPGA üzerine değiştirilmiş bitstream dosyası yüklenirken Yükleyici modül bitstream dosyasını okuyup değişiklik yapılıp yapılmayacağına karar vererek FPGA'e yazmaktadır. Değişiklik yapılacaksa şifreli dosyadan anahtarla çözerek elde ettiği gerekli LUT değişikliğini ve başlangıç değerini yama yaparak FPGA in programlanacak tarafına yazmaktadır. Değişiklik gerekmiyorsa ilgili kısım değiştirilmeden FPGA'e yazılmaktadır. Bu yöntemde gerçekleşen işlemler tasarımcı akışı, kullanıcı akışı ve Yükleyici akışı olarak üç temel süreçte gerçekleştirilmektedir. Ayrıca FPGA üreticisi, 3PIP tasarımcısı ve 3PIP kullanıcısı dışında "Güvenilir Anahtar Tutucu" tanımlamasıyla 4. bir taraf 3PIP tasarım ve kullanım sürecine dahil olmaktadır. Şekil 2.5'te önerilen çözüm sisteminin gösterimi yapılmaktadır.



Şekil 2.5: [7]'de önerilen 3PIP koruma yöntemi.

3. MANTIKSAL KİLİTLEME TEKNİĞİ VE SALDIRI YÖNTEMLERİ

3.1 Mantıksal Kilitleme Tekniği

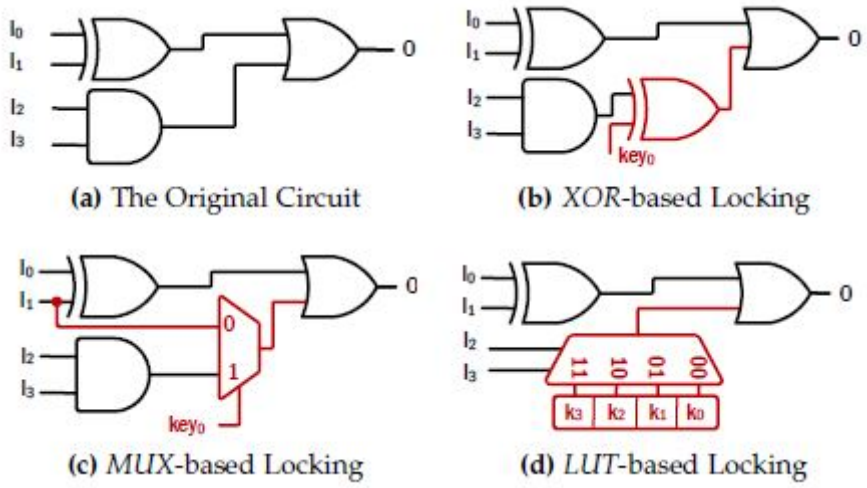
Mantıksal kilitleme, sisteme eklenen kapılarla üretim sonrasında programlanabilme yeteneğinin kazandırılmasını sağlayan bir tekniktir. Eklenen kapılar, "anahtarla programlanabilir kapılar" (key gates) olarak adlandırılır. Mantıksal kilitleme teknikleri, farklı seviyelerde uygulanabilmektedir [6]. Çizelge 3.1, mantıksal kilitlemenin farklı seviyelerde uygulandığında genel özelliklerini göstermektedir [6]. Mevcut durumda, mantıksal kilitleme teknikleri yaygın olarak kapı düzeyinde netlist seviyesinde, RTL (Register Transfer Level) ve HLS (High-Level Synthesis) seviyelerinde kullanılmaktadır.

Çizelge 3.1: Farklı Düzeyde Uygulanan Mantıksal Kilitleme Özellikleri.

Uygulama Düzeyi	Uygulanma Şekli	Getireceği Yük	Uygulama Zorluğu
Yerleşim düzeyi	Bit düzeyinde, kablolama	Sıfıra yakın	Yüksek
Transistör düzeyi	Bit düzeyinde, anahtarlama, kablolama	Düşük	Yüksek
Kapı düzeyi	Bit düzeyinde, mantıksal	Orta	Orta
RTL düzeyi	Bit düzeyinde, işlemsel, davranışsal	Orta-Yüksek	Düşük
HLS düzeyi	Bit düzeyinde, işlemsel, davranışsal	Orta-Yüksek	Düşük

Mantıksal kilitlemede kullanılan anahtar kapılarının türüne bağlı olarak mantıksal kilitleme teknikleri üç ana grupta sınıflandırılabilir: (1) XOR tabanlı, (2) MUX tabanlı ve (3) LUT tabanlı. Şekil 3.1, bu üç modelin her biri için kapı düzeyinde basit örnekler sunmaktadır [6].

RTL ve HLS seviyelerinde mantıksal kilitleme yapılarak operasyonel veya davranışsal yapı hedefleme yapılabilir. Bu iki seviyede sentezleme sonrası elde edilen



Şekil 3.1: Kullanılan kapı türüne göre mantıksal kilitleme teknikleri [6].

netlistlerdeki kilitleme işlemleri, genellikle bu üç anahtar kapı türünden biri (veya bunların kombinasyonları) kullanılarak gerçekleştirilebilir.

Mantıksal kilitlemenin gizli öğesi olan anahtar, kilitlenmiş devrenin doğru işlevselliğinin geri kazanılabilmesi için sağlanmalıdır. Mantıksal kilitlemenin anahtar yükleme işlemi, güvenilir bir tesiste gerçekleştirilmeli ve üretimden sonra kurcalamaya dayanıklı bellek içinde saklanmalıdır.

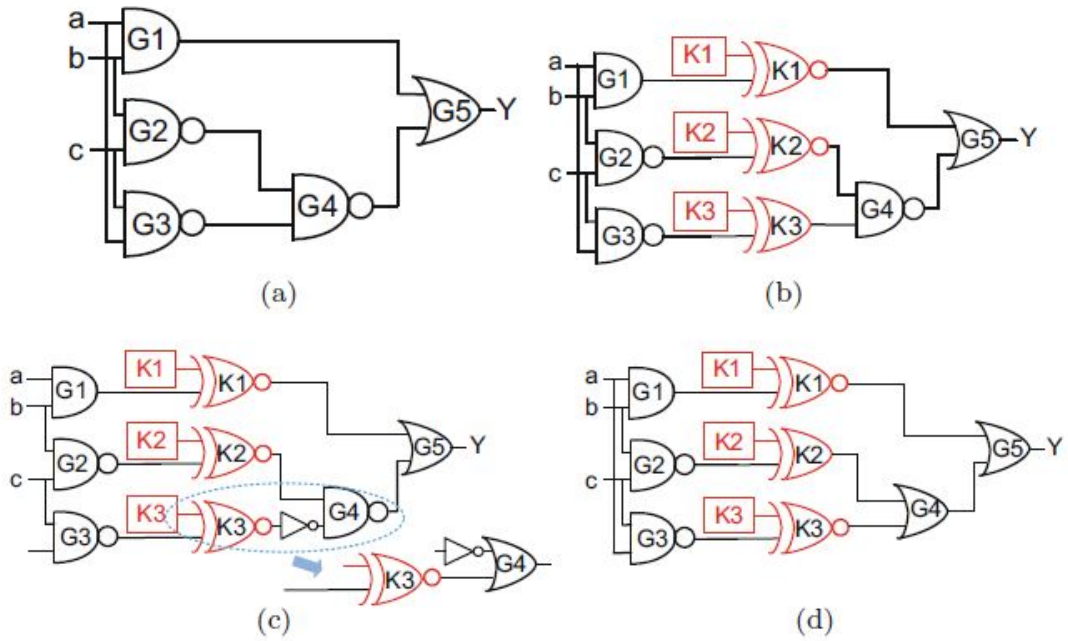
Mantıksal kilitleme tekniğinin XOR kapıları ile uygulanması Şekil 3.2'de gösterilmiştir. Şekil 3.2'de gösterilen kilitlenmiş devrenin farklı anahtar değerleri ile çalışma durumu Şekil 3.3'te tablo olarak verilmiştir. Tablodan görülebileceği gibi sadece doğru anahtar değerinde kilitli devre orijinal devre gibi çalışmaktadır.

Saldırıların hedefi mantıksal kilitleme anahtarını ele geçirmektir. Mantıksal kilitleme tekniği için çeşitli teknikler geliştirilirken [20] , öte yandan çok çeşitli saldırılar da gerçekleştirilmiştir [21]. Saldırılardan en etkili olanı SAT saldırısı olmuştur [22].

3.2 Mantıksal Kilitleme Tekniğine Yapılabilen Saldırı Yöntemleri

Mantıksal kilitleme tekniği için tehditler farklı sınıflandırmalarla ayrıştırılabilir. Örneğin doğrulama devresi gerektirip gerektirmediğine göre iki grupta değerlendirilebilir. Bu bağlamda saldırgan açısından saldırıların bazısı en az iki adet ASIC gerektirirken bunlardan biri aktifleştirilmiş (doğrulama devresi - oracle-guided)

olmalıdır [6]. [8]'da ise mantıksal kilitlemeye yapılan saldırılar algoritmik ataklar, yapısal ataklar ve yan kanal atakları olarak gruplandırılmıştır.



Şekil 3.2: Mantıksal Kilitleme uygulamasının örnek devre üzerinde gösterimi:
(a) Orijinal devre
(b) XOR/XNOR kapılar ile kilitli devre
(c) Kilitli devrenin farklılaştırılması
(d) b. deki devrede tersleyicilerle kapıların değiştirilmiş hali [8].

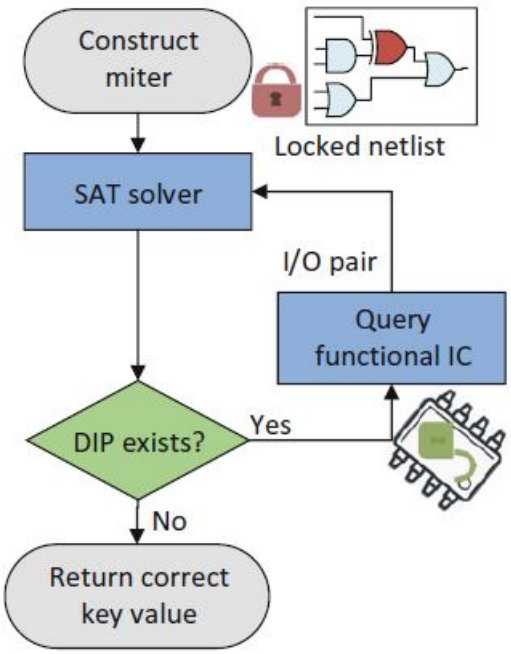
abc	Y	k0:000	k1:001	k2:010	k3:011	k4:100	k5:101	k6:110	k7:111
000	0	1	1	1	1	1	1	0	1
001	0	1	1	1	1	1	1	0	1
010	0	1	1	1	1	1	1	0	1
011	1	1	1	1	1	0	1	1	1
100	0	1	1	1	1	1	1	0	1
101	1	1	1	1	1	1	1	1	0
110	1	1	1	0	1	1	1	1	1
111	1	1	0	1	1	1	1	1	1

Şekil 3.3: Şekil 3.2'de verilen kilitli devrenin farklı anahtar değerleri ile işlevsellik durumu.

3.2.1 SAT saldırısı

SAT saldırısı mantıksal çözümleme tabanlı (Boolean satisfiability) bir saldırı yöntemidir. Saldırı için kilitli bir devre ve onun aktifleştirilmiş doğru çalışan bir örneğine ihtiyaç duyulur. Klasik mantıksal kilitleme tekniklerinin SAT saldırısı ile kolayca kırılabildikleri [22]'te gösterilmiştir. Saldırı aşağıda ki adımları

gerçekleştirerek doğru anahtar değerini bulmaya çalışır. Saldırının akış şeması Şekil 3.4'te gösterilmektedir.



Şekil 3.4: SAT saldırısının akış şeması [8].

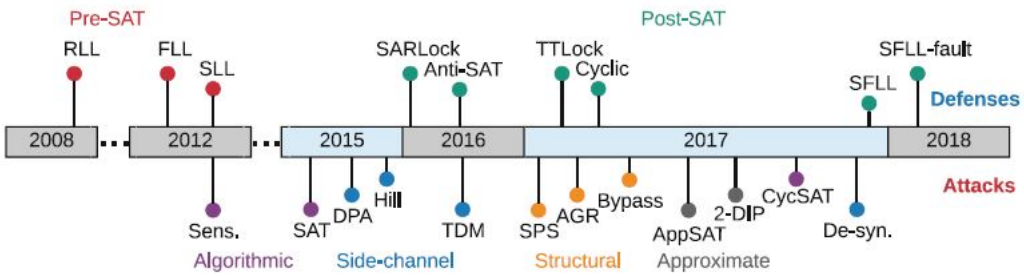
1. Oracle erişimi sağlanır:
   - Saldırgan, devrenin doğru çalışan bir örneğine (örneğin orijinal çip) sahiptir.
   - Bu oracle, verilen herhangi bir girişin doğru çıkışını üretir.
2. Kilitli devre analizi:
   - Saldırgan, elindeki kilitlenmiş devreyi (obfuscated circuit) analiz eder.
   - Devredeki anahtar girişlerinin (key inputs) bilinmediği varsayılır.
3. İki devreli model kurulur:
   - SAT çözücüsüne iki versiyon yüklenir:
     - Aynı girişleri alacak, farklı anahtarlarla çalıştırılacak iki kopya.
   - Amaç: Bu iki devre farklı sonuç üretirse, o giriş ayırt edici (discriminating input) olur.
4. Ayırt edici giriş bulunur:
   - SAT çözücüsü, hangi girişlerin farklı anahtar kombinasyonlarını ayırt edebildiğini belirler.
   - Bu giriş oracle'a verilerek doğru çıkış öğrenilir.

5. Yanlış anahtarlar elenir:
   - Doğru çıkışı vermeyen anahtar kombinasyonları sistematik olarak elenir.
   - Bu işlem tekrar tekrar yapılır.
6. Doğru anahtar elde edilir:
   - Tüm olasılıklar elendikten sonra geriye kalan tek kombinasyon doğru anahtardır.

3.2.2 SAT saldırısına dirençli mantıksal kilitleme tekniği çalışmaları

SAT saldırısı mantıksal kilitleme tekniği için bir dönüm noktasıdır. Geliştirilen yöntemler SAT atak öncesi SAT atak sonrası diye anılmaktadır. [8]'da mantıksal kilitleme tekniği çalışmaları kronolojik olarak sıralanmış ve incelenmiştir. SAT saldırısına dayanıklı mantıksal kilitleme teknikleri geliştirilmeye devam edilmektedir.

Şekil 3.5'te 2018'e kadar yapılan mantıksal kilitleme tekniği çalışmaları SAT öncesi ve SAT sonrası diye ayrılmış şekilde gösterilmiştir. Savunma amaçlı çalışmalar üst tarafta saldırı çalışmaları ise alt tarafta işaretlenmiştir. SAT saldırısı öncesinde RLL [20], FLL [23], SLL [24] gibi mantıksal kilitleme teknikleri önerilmiş ancak SAT saldırısı ile bunların kırıldığı görülmüştür. SAT sonrasında araştırmacılar SAR Lock [25], Anti-SAT [26], SFLL [27] gibi teknikler geliştirmişlerdir. 2018 sonrasında geliştirilen savunma amaçlı çalışmalara örnek olarak 2023'te yapılmış [28] ve 2024'te yapılmış [29] verilebilir.

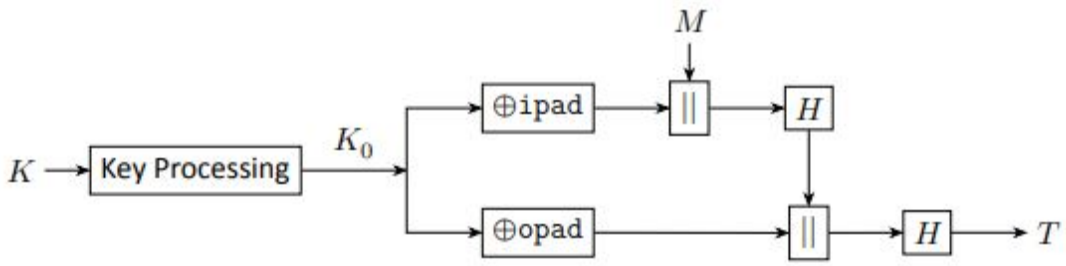


Şekil 3.5: SAT öncesi ve sonrası mantıksal kilitleme çalışmaları ve saldırılar [8].

4. KRİPTO MİMARİDE KULLANILAN ALGORİTMALAR

4.1 Anahtarlı HMAC Algoritması

Anahtarlı özet tabanlı mesaj kimlik doğrulama kodu (Keyed Hashed Message Authentication Code - Keyed HMAC) simetrik anahtarlı bir kriptografik mekanizmadır. Gizli bir anahtar kullanılarak bir kimlik doğrulama etiketi üretmek ve bu etiketle iletilen verinin kaynağını doğrulamak için kullanılır. Bu etiket, veride olabilecek yetkisiz değişikliklerin tespit edilmesini de sağlar. Şekil 4.1'de algoritmanın işlevsel diyagramı görülmektedir.



Şekil 4.1: HMAC Algoritmasının işlevsel diyagramı [30].

HMAC algoritması ile etiket 2 adımda üretilir.

1. Anahtar İşleme

Ara değer K_0 aşağıdaki şekilde belirlenir:

a. Eğer $(\text{len}(K) = b)$ ise $(K_0 = K)$ olarak ayarlanır.

   b: Kullanılacak özet fonksiyonunun blok büyüklüğüdür.

   K: Kullanılacak gizli anahtardır.

   K_0: K anahtarından elde edilen b bit uzunluğunda ara anahtardır.

b. Eğer $(\text{len}(K) > b)$ ise, $(K_0 = H(K)\,||\,0^{b-\ell})$

   ℓ: Kullanılacak özet fonksiyonunun çıkış blok büyüklüğüdür.

   Yani, $H(K)$'ye $(b-\ell)$ adet sıfır (bit) eklenerek b bitlik bir dize K_0 elde edilir.

c. Eğer (len$(K) < b$) ise, ($K_0 = K \,||\, 0^{b-\text{len}(K)}$)

Yani, (K)'ye ($b - \text{len}(K)$) adet sıfır (bit) eklenerek (b) bitlik bir dize (K_0) elde edilir.

2. Etiketin (T) Üretilmesi

Etiket değeri (T) 4.1'de gösterildiği gibi elde edilir:

$$T = \text{HMAC}(K,M) = H\left((K_0 \oplus \text{opad}) \,||\, H\left((K_0 \oplus \text{ipad}) \,||\, M\right)\right) \tag{4.1}$$

ipad: Bit dizisi "00110110" (0x36) 'b/8' defa tekrarlanmasıyla elde edilen iç dolgudur. opad: Bit dizisi "01011100" (0x5C) 'b/8' defa tekrarlanmasıyla elde edilen dış dolgudur.

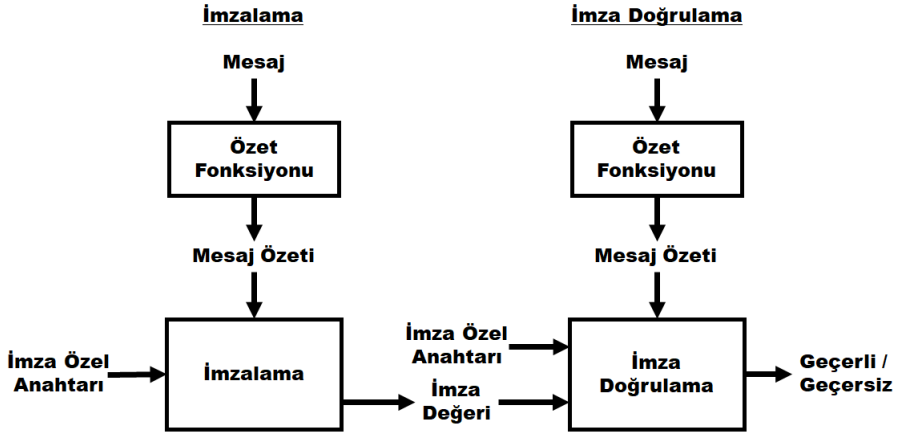
4.2 SHA256 Özet Algoritması

Bu çalışmada HMAC algoritması içinde özet algoritması olarak SHA-256 algoritması kullanılmıştır. SHA-256 algoritması FIPS 180-4 Secure Hash Standard [31] dokümanında tanımlanmaktadır.

Algoritma döngüsel sağa kütük kaydırma ve sağa kütük kaydırma fonksiyonları kullanarak 512 bitlik blokları işlemektedir. İşleme başlamadan önce standartta tanımlı sabitle ilklendirme yapılmaktadır. Mesaj bloğu 512 bitten kısa olduğu durumda dolgulama yapılmaktadır. Ardından fonksiyonları çalıştırarak elde ettiği 8 adet 32 bitlik özet parametresini birleştirerek 256 bit büyüklüğünde mesajın özeti elde edilmektedir.

4.3 Sayısal İmza Algoritması

Sayısal imza yazılı imzanın elektronik ortamda karşılığıdır. Sayısal imza algoritması (Digital Signature Algorithm - DSA) elektronik haberleşme kanalıyla iletilen ya da depolandığı elektronik ortamdan çıkarılan verinin/bilginin kaynağını belirleme ve bütünlüğünün bozulmadığını anlamak için kullanılan bir algoritmadır. FIPS-186-5 standartında [32] anlatılan imzalama ve imza doğrulama süreci Şekil 4.2'de gösterilmiştir. Standartta [32] 3 adet DSA algoritması tanımlanmıştır. Bunlar RSA, ECDSA ve EdDSA algoritmalarıdır. Bu çalışmada RSA algoritması kullanılacağı için RSA algoritması tanımı verilmiştir.



Şekil 4.2: İmzalama ve imza doğrulama süreci akış şeması.

4.3.1 RSA sayısal imza algoritması

RSA algoritması 1978'de Rivest R. , Shamir A ve Adleman L. tarafından tanımlanmıştır [33]. RSA algoritması şifreleme ve şifre çözme işlemlerinde de kullanılmaktadır. Ancak şifreleme ve şifre çözmede farklı anahtarlar (özel-açık anahtar çifti) kullanıldığı ve hız performansı nedeniyle pek tercih edilmemektedir. Bununla birlikte imzalama-imza doğrulama işlemleri için pratik ve kolay bir çözüm olmaktadır.

RSA algoritmasının dijital imza oluşturma ve doğrulama amacıyla kullanımı IETF RFC 8017 [34]'te rastsallık eklenerek (RSA Signature Scheme with Appendix – Probabilistic Signature Scheme) RSASSA-PSS şeması tanımlanmıştır. RSASSA-PSS, RSA imzalama prosedürü ve RSA imza doğrulama prosedürünü EMSA-PSS kodlama yöntemiyle birleştirmiştir.

RSA İmzalama Prosedürü :

$$S = RSASSA - PSS - SIGN(K, M) \tag{4.2}$$

4.2'de yer alan değişkenlerin tanımları aşağıda verilmiştir:

K: İmzalayanın RSA özel anahtarı

M: İmzalanacak mesaj (bir oktet dizisi)

S: İmza (uzunluğu k oktet olan bir oktet dizisi; burada k, RSA modülü n'nin oktet cinsinden uzunluğudur)

İşlem adımları şu şekildedir:

1. EMSA-PSS Kodlama: Mesaj M'ye, tanımlı kodlama işlemi uygulanarak uzunluğu $\lceil(\text{modBits}-1)/8\rceil$ oktet olan kodlanmış mesaj EM 4.3'teki gibi elde edilir:

$$EM = \text{EMSA-PSS-ENCODE}(M, modBits-1) \tag{4.3}$$

   modBits, RSA modülü n'nin bit cinsinden uzunluğudur.

2. RSA İmza:

   a. 4.4'te kodlanmış mesaj EM 'nin, temsili mesaj değeri m 'ye dönüşümü gösterilmiştir:

$$m = \text{OS2IP}(EM) \tag{4.4}$$

   m tamsayısının bit uzunluğu en fazla $modBits-1$ olmalıdır.

   b. Tamsayı biçimindeki temsili mesaj değeri m ile, RSA özel anahtarı K kullanılarak temsili imza değeri s, 4.5'teki gibi hesaplanır:

$$s = \text{RSASP1}(K, m) \tag{4.5}$$

   c. 4.6'da temsili imza değeri s 'nin, uzunluğu k oktet olan imza değeri S 'ye dönüşümü verilmiştir:

$$S = \text{I2OSP}(s, k) \tag{4.6}$$

   d. İmza S çıktı olarak verilir.

RSA İmza Doğrulama Prosedürü :

$$Sonuc = RSASSA-PSS-VERIFY((n,e), M, S) \tag{4.7}$$

4.7'de yer alan değişkenlerin tanımları aşağıda verilmiştir:

(n, e): İmzalayanın RSA açık anahtarıdır.

M: İmzası doğrulanacak mesajı ifade eder, bir oktet dizisidir.

S: Doğrulanacak imza, uzunluğu k oktet olan bir oktet dizisi; burada k, RSA modülü n'nin oktet cinsinden uzunluğudur.

Sonuç: Geçerli imza veya geçersiz imza olarak verilir.

İşlem adımları şu şekildedir:

1. Uzunluk kontrolü: İmza S'nin uzunluğu k oktet değilse, geçersiz imza çıktısı verilir ve işlem durdurulur.

2. RSA doğrulaması:

   a. İmza değeri S, imzanın tamsayı temsili değeri s'ye 4.8'deki gibi dönüştürülür:

   $$s = \text{OS2IP}(S) \tag{4.8}$$

   b. RSA açık anahtarı (n,e) ve temsili imza değeri s kullanılarak, temsili mesaj değeri m 4.9'daki gibi hesaplanır:

   $$m = \text{RSAVP1}((n,e),s) \tag{4.9}$$

   Eğer s değeri aralık dışında ise geçersiz imza çıktısı verilir ve işlem durdurulur.

   c. Temsili mesaj değeri m, uzunluğu $emLen = \lceil (modBits - 1)/8 \rceil$ oktet olan kodlanmış mesaj EM'ye 4.10'da gösterildiği gibi dönüştürülür:

   $$EM = \text{I2OSP}(m, emLen) \tag{4.10}$$

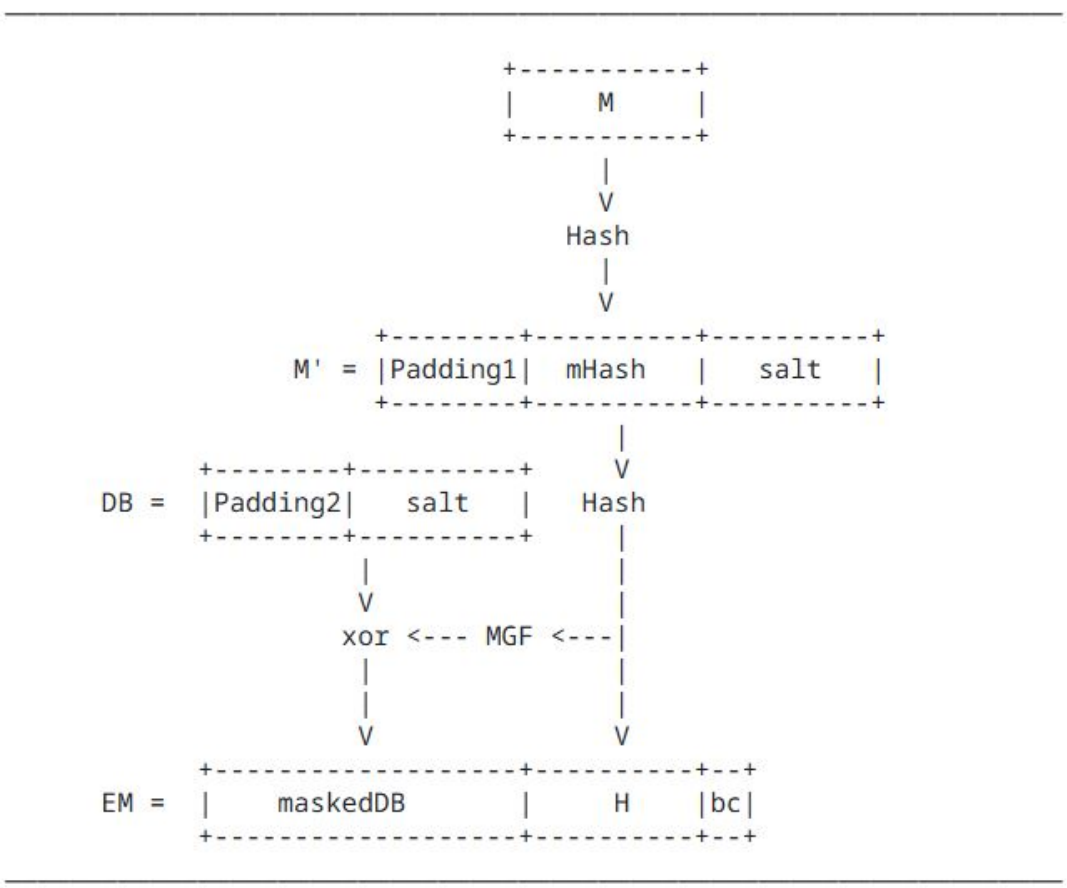
   Not: Burada $modBits$, RSA modülü n'nin bit cinsinden uzunluğudur. Eğer $modBits - 1$ 8'in tam katı ise $emLen$ değeri $k-1$ olur, aksi halde k'ye eşittir. Eğer I2OSP() sonucu aralık dışında ise, geçersiz imza çıktısı verilir ve işlem durdurulur.

3. EMSA-PSS doğrulaması: Mesaj M ve kodlanmış mesaj EM üzerinde EMSA-PSS doğrulama işlemi 4.11'de gösterildiği gibi uygulanır. *Sonuc* başarılı ise gerçerli imza çıktısı verilir, aksi halde geçersiz imza çıktısı verilir.

$$Sonuc = \text{EMSA-PSS-VERIFY}(M, EM, modBits - 1) \tag{4.11}$$

EMSA-PSS Mesaj Kodlama Prosedürü :

EMSA-PSS de tanımlanan kodlama işlemi Şekil 4.3'te gösterilmektedir.



Şekil 4.3: EMSA-PSS kodlama işlem akışı. Doğrulama işleminde önce tuz değeri elde edilir. Daha sonra M' özet değeri hesaplanıp karşılaştırılır [34].

RSA imzalama prosedürü (RSASP) şu şekildedir:

$$s = RSASP1(K, m) \tag{4.12}$$

4.12'de yer alan değişkenlerin tanımları aşağıda verilmiştir:

K: RSA özel anahtarı.

m: Mesaj tamsayı değeri, 0 ile n-1 arasında bir tamsayıdır. Mesajın izin verilmiş fonksiyonlar kullanılarak kodlanıp tamsayıya çevrilmesi ile elde edilir.

s: İmza değerini ifade etmektedir. 0 ile n-1 arasında bir tamsayı olması gerekir.

İşlem Adımları:

1. Eğer m değeri $0 \leq m < n$ aralığında değilse, hata oluşur işlem durdurulur.

2. İmza değeri s 4.13'te verilen formül kullanılarak hesaplanır:

$$s = m^d \bmod n \tag{4.13}$$

3. Hesaplanan s değeri imza değeri olarak verilir.

RSA imza doğrulama prosedürü (RSAVP) şu şekildedir:

$$m = RSAVP1((n,e),s) \tag{4.14}$$

4.14'te yer alan değişkenlerin tanımları aşağıda verilmiştir:

(n, e): RSA açık anahtarıdır ve geçerli olduğu varsayılmaktadır.

s: Doğrulanacak imza değeridir. $0 \leq s < n$ olacak şekilde bir tamsayı olması gerekir.

m: Mesaj tamsayı değeri, 0 ile n-1 arasında bir tamsayıdır.

İşlem Adımları:

1. Eğer s değeri $0 \leq s < n$ aralığında değilse, hata oluşur işlem durdurulur.

2. Mesaj tamsayı değeri 4.15'te gösterildiği şekilde hesaplanır:

$$m = s^e \bmod n \tag{4.15}$$

3. Hesaplanan m değeri çıktı olarak verilir.

FIPS sayısal imza standardı [32], güvenliği sağlamak için aşağıda sıralanan ilave kısıtlamalara dikkat edilmesini belirtmektedir.

- RSA Anahtar Çifti: RSA dijital imza işlemi için bir özel anahtar (n, d) ve bir açık anahtardan (n, e) oluşur. Özel anahtarın şu oluşumu da mevcuttur:
  (p, q, dP, dQ, qInv) beşlisinden ve (muhtemelen boş olan) (r_i, d_i, t_i) üçlülerinden oluşan bir dizi (i = 3, ..., u) içeren bir yapı.
  Bu anahtar çifti yalnızca dijital imzalar için kullanılmalı, başka amaçlarla (örneğin anahtar değişimi) kullanılmamalıdır.

- Modül (n): İki asal sayı (p ve q) çarpımıdır. Modül uzunluğu (nlen) bit cinsinden ifade edilir ve en az 2048 bit, çift sayı ve p ile q aynı uzunlukta olmalıdır.
- Gizlilik ve Güvenlik: p, q ve özel anahtar d gizli tutulmalıdır. n ve e herkese açık olabilir.
- Güvenlik Düzeyi: RSA'nın güvenliği, kullanılan modül uzunluğu ve özet (hash) fonksiyonunun güvenliği ile sınırlıdır. Hash fonksiyonu güvenliği, modül güvenliğine eşit veya daha yüksek olmalıdır.
- Sertifika Yetkilileri (CA): CA'nın kullandığı modül uzunluğu, abonelerinkinden küçük olmamalıdır.
- Anahtar Üretimi: RSA parametreleri (p, q, e) güvenli, onaylı rastgele bit üreteçleriyle oluşturulmalı, kullanılan tohumlar gizli tutulmalı veya imha edilmelidir.

5. ÖNERİLEN FPGA GÜVENLİK MODÜLÜ VE KRİPTO MİMARİ TASARIMI

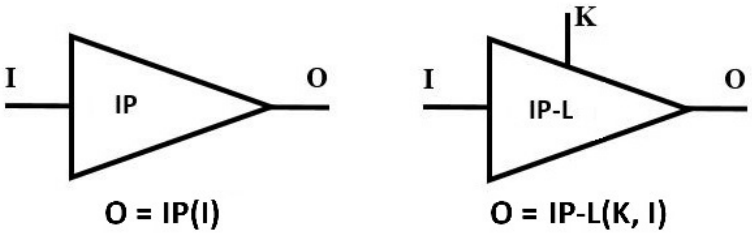
Bu bölümde 3PIP'lerin tasarım bilgisinin tasarımcısı dışında elde edilememesi için FPGA 'lerde sistem mimarisi kapsamında bulunacak bir güvenlik modülü ve geliştirme aracı sentezleme işlemlerinde bir değişiklik tanımlanmaktadır. Bu çözümün sahibi FPGA üreticileri, kullanıcılarıda 3PIP tasarımcılarıdır. Bu nedenle FPGA üreticileri güvenilir kabul edilmektedir. Aşağıda sistem bileşenleri detaylı olarak açıklanacaktır.

5.1 Önerilen Sistemin Genel Yapısı

Sistemin genel yapısı FPGA üreticilerinin FPGA sistem mimarisine yerleştirecekleri bir LUL modülü, 3PIP tasarımcılarının mantıksal kilitleme tekniği ile tasarımlarını kilitlemeleri, FPGA üreticisinin belirli bir kripto mimari adımlarını gerçekleştirmesi ve 3PIP kullanıcısının 3PIP yi uygun şekilde tasarımına dahil etmesini kapsamaktadır. Bu bölümde teorik olarak tasarım ve yöntem anlatılacaktır.

5.1.1 3PIP tasarımı ve mantıksal kilitleme uygulanması

3PIP tasarımcısı mantıksal kilitleme tekniği kullanarak 3PIP yi kilitleyecektir. Mantıksal kilitleme yaparken 3PIP kapı sayısına uygun büyüklükte bir kilitleme anahtarı 'K' kullanılacaktır. Şekil 5.1'de bir 3PIP devresi ve mantıksal kilitleme uygulanmış 3PIP devresinin (3PIP-L) şematik gösterimi yapılmıştır. 3PIP devresi sadece bir giriş ve bir çıkışla gösterilirken, mantıksal kilitleme yapılmış 3PIP devresine 'K' girişi eklenmektedir. 3PIP kilitleme işlemi netlist dosyası üzerinde gerçekleştirilmelidir. Mantıksal kilitlenmiş 3PIP sentezlendiğinde oluşan bit dosyası içinde de hala mantıksal kilitleme uygulanmış şekilde kapılar içerecektir. Mantıksal kilitleme yapılmış 3PIP devresinin 'K' girişine doğru K anahtarı verildiği sürece orijinal 3PIP devresi gibi çalışacaktır.



Şekil 5.1: 3PIP ve mantıksal kilitleme yapılmış 3PIP şematik gösterimi.

5.1.2 Sistemde uygulanacak kripto mimarinin tasarımı

Sistemde uygulanacak kripto mimari FPGA üreticisinin gerekli parametreleri elde ederken gerçekleştirmesi gereken işlemleri tanımlamaktadır. Aynı işlemler LUL modülünde mantıksal kilitleme yapılmış 3PIP devresinin doğru çalışabilmesi için 'K' anahtarı elde edilirken de kullanılmaktadır. Aşağıdaki işlemleri gerçekleştirirken FPGA üreticisi mantıksal kilitleme yapılmış 3PIP dosyasına ve 'K anahtarına sahip olmalıdır.

1. FPGA üreticisi üretici $3PIP-L$ devresine K_0 anahtarı ve I_0 giriş verisini n cycle boyunca vererek devrenin O_{0n} çıkış veri dizisini elde eder. Bu işlem 5.1'de tanımlanmıştır. $3PIP-L$ devresi $3PIP_L$ şeklinde gösterilmiştir.

$$O_{0n} = 3PIP_L(K_0, I_0) \ \ (n\ clock\ boyunca) \tag{5.1}$$

   5.1'de K_0, FPGA üreticisinin K girişi için tanımladığı sabit anahtar değerini, I_0 ise I girişinden sağladığı, $0x00...0$ ile başlayıp her saat (clock) döngüsünde bir artarak n'e kadar sayan sayaç değerini temsil etmektedir. ($n \gg 0$).

2. FPGA üreticisi $3PIP-L$ devresinin çıkış verisi olan O_{0n} verisinin anahtarlı HMAC algoritması ile özet değerini 5.2'de gösterildiği şekilde hesaplar. $HMAC(3PIP_L)$ özet değerini ifade etmektedir.

$$HMAC(3PIP_L) = HMAC(K_{FPGA}, O_{0n}) \tag{5.2}$$

   5.2'de K_{FPGA}, FPGA üreticisinin mantıksal kilitleme için belirlediği gizli anahtarı ifade etmektedir. K_{FPGA} anahtarı FPGA'lere üretilirken yazılır ve FPGA

üzerinde gömülü olarak bulunur. K_{FPGA} anahtarının FPGA'de bulunduğu belleğin kurcalamaya karşı dayanıklı bellek olduğu varsayılmaktadır.

3. FPGA üreticisi anahtar maske parametresi P_{km}'yi 5.3'te belirtilen şekilde hesaplar. Ancak bu adım LUL modülünde K anahtarını elde ederken 5.4'te gösterildiği gibi yapılır.

$$P_{km} = HMAC(3PIP_L) \oplus K \tag{5.3}$$

$$K = HMAC(3PIP_L) \oplus P_{km} \tag{5.4}$$

5.3 ve 5.4'te K, $3PIP-L$ devresinin orijinal $3PIP$ devresi şeklinde çalışmasını sağlayan doğru "K" anahtarını belirtmektedir.

4. FPGA üreticisi K anahtarını ve 1. adımda tanımlanan I_0 giriş verisini kullanarak O_{1n} çıkış veri dizisini 5.5'te gösterildiği şekilde elde eder.

$$O_{1n} = 3PIP_L(K, I_0) \ \ (n\ clock\ boyunca) \tag{5.5}$$

5.5'te K anahtarı doğru olduğu durumda O_{1n} çıkış verisi orijinal $3PIP$ devresinin çıkış verisidir.

5. FPGA üreticisi, K_{FPGA} anahtarını kullanarak 5.5'te elde edilen O_{1n} çıkış verisinin anahtarlı HMAC özet değerini 5.6'da gösterildiği gibi hesaplar.

$$HMAC(3PIP) = HMAC(K_{FPGA}, O_{1n}) \tag{5.6}$$

O_{1n} çıkış verisi orijinal $3PIP$ devresinin çıkış verisi olduğundan özet değeri $HMAC(3PIP)$ şeklinde gösterilmiştir.

6. 5.6'da hesaplanan değer aynı zamanda sistemde tanımlanan kimlik doğrulama parametresi P_{ka} parametresidir. Bu durum 5.7'de ifade edilmektedir.

$$P_{ka} = HMAC(3PIP) \tag{5.7}$$

7. FPGA üreticisi 5.8'de gösterildiği gibi 3PIP-L dosyasını imza algoritması ile kendi özel anahtarını kullanarak imzalar ve S_{3PIP-L} imza değerini elde eder. İmzalama ve imza doğrulama işlemlerinde Bölüm 4'te anlatılan RSA imzalama (RSASSA-PSS-SIGN) ve imza doğrulama (RSASSA-PSS-VERIFY) algoritması kullanılır.

$$S_{3PIP-L} = Imzala(K_{ozel}, (3PIP-L)) \tag{5.8}$$

$$Sonuc = ImzaDogrula(K_{acik}, (3PIP-L), S_{3PIP-L}) \tag{5.9}$$

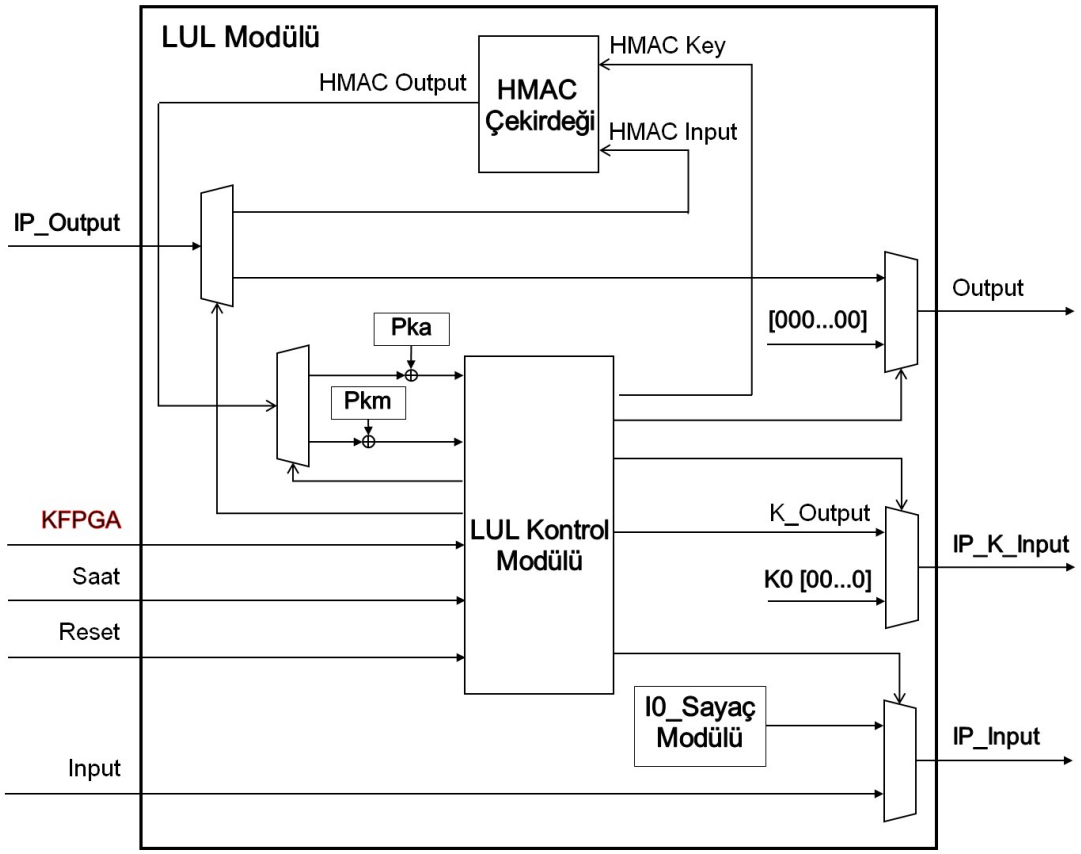
5.9'da gösterilen imza doğrulama işlemi 3PIP-L'nin kullanıldığı FPGA IP'si sentezlenirken FPGA geliştirme aracı (IDE) tarafından gerçekleştirilecektir.

5.1.3 FPGA güvenlik modülü tasarımı

Şekil 5.2'de FPGA sistemi için tasarlanan güvenlik modülünün (Mantıksal Kilitleme Çözme Modülü : Logic Unlocker (LUL) Modülü) yapısı şematik olarak gösterilmektedir. Şekil 5.2'den de anlaşılacağı gibi LUL modülü aslında 3PIP-L modülünü kapsayan bir modül olacak şekilde tasarlanmıştır. $3PIP-L$ devresinin girişleri $3PIP\_Input$ ve $3PIP\_K\_Input$ ile çıkışı $3PIP\_Output$ bu modülün içinde kalmalıdır. K_{FPGA} anahtarı FPGA güvenli bellekten LUL modülü içine alınacaktır. Şekilde n değeri gösterilmemiştir. n değeri FPGA modeli için yeterli büyüklükte sabit bir değer olacak ve LUL modülünde kazılı olarak bulunacaktır. LUL modülü 3 ade MUX, 2 adet DEMUX, 1 adet sayaç modülü, 1 adet anahtalı HMAC algoritma çekirdeği, 1 adet LUL Kontrol Modülünden oluşmaktadır.

K_{FPGA} anahtarı FPGA *Reset* işareti ile güvenli bellekten LUL modülü içine alındıktan sonra LUL modülü çalışmaya başlayacaktır. LUL Kontrol Modülü kripto mimaride belirtilen adımları gerçekleştirecek şekilde giriş ve çıkış işaretlerini yönlendirmesini ve zamanlamayı gerçekleştirecektir.

LUL modülü çalışmaya başladıktan sonra HMAC çekirdeğine K_{FPGA} anahtarını gönderecektir. Daha sonra 1. adımda belirtildiği gibi n saat işareti boyunca $3PIP-L$ devresine K_0 anhtarını ve I_0 sayaç çıkışını giriş olarak verecek ve $3PIP\_Output$ çıkış verisini (O_{0n}) uygun boyuta çevirerek HMAC çekirdeğine yönlendirecektir. Tüm



Şekil 5.2: LUL Modülü yapısının şematik gösterimi.

çıkış değerleri HMAC çekirdeğine gönderilip özet alma işlemi tamamlandıktan sonra, oluşan özet değerini P_{km} ile XOR layıp K anahtarını elde edecektir.

Ardından 4. adımda belirtildiği gibi n saat işareti boyunca $3PIP-L$ devresine K anhtarını ve I_0 sayaç çıkışını giriş olarak verecek ve $3PIP\_Output$ çıkış verisini (O_{1n}) uygun boyuta çevirerek HMAC çekirdeğine yönlendirecektir. Tüm çıkış değerleri HMAC çekirdeğine gönderilip özet alma işlemi tamamlandıktan sonra, oluşan özet değerini P_{ka} ile XOR layıp $0x$"00....0" dizisi elde edecektir. Bu durum her şeyin doğru olduğunu gösterecek ve LUL Kontrol Modülü 3PIP-L modülünü normal çalışma modunda çalıştırmaya başlayacaktır.

Normal çalışma modunda 3PIP-L devresinin $3PIP\_K\_Input$ girişi sürekli K anahtarı ile sürülecektir. LUL modülüne giren $Input$ işareti 3PIP-L devresinin $3PIP\_Input$ girişine yönlendirilecektir. 3PIP-L devresinin çıkışı $3PIP\_Output$ işareti de LUL modülünün çıkışı $Output$ işaretine yönlendirilecektir.

Herhangi bir olumsuzluk durumunda LUL modülü 3PIP-L devresine I_0 sayaç işareti verecek ve LUL modülünün çıkışı *Output* işaretine $0x00....0$ dizisi yönlendirilecektir.

5.2 Önerilen Sistemin Uygulama Adımları

5.2.1 3PIP tasarımcısının yapacağı işlemler

3PIP tasarımcısının yapması gereken işlem 3PIP devresinin içerdiği kapı sayısına uygun büyüklükte bir K anahtarı belirlemek ve bu anahtarla uygun bir mantıksal kilitleme aracı kullanarak 3PIP yi kilitlemektir. Daha sonra elde ettiği 3PIP-L dosyasını ve K anahtarını FPGA üreticisine gönderecektir.

5.2.2 FPGA üreticisinin yapacağı işlemler

FPGA üreticisi LUL modül tasarımını FPGA modellerine yerleştirerek ürettiği FPGA lerde LUL modülü ve K_{FPGA} anahtarın bulunmasını sağlamalıdır. 3PIP koruma için açık ve özel anahtar çifti oluşturmalıdır. IDE programına imza doğrulama işlemi ve açık anahtarı eklemelidir. Bunların yapıldığını varsayıyoruz.

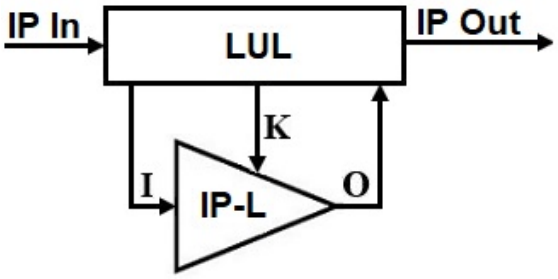
Yukarıda ki varsayım üzerine FPGA üreticisi 3PIP tasarımcısı tarafından kendisine gönderilen dosyayı ve K anahtarını kullanarak kripto mimari tanımda belirtilen adımları gerçekleştirecektir. Elde edeceği P_{km} ve P_{ka} parametreleri ile 3PIP-L dosyasının imza değeri S_{3PIP-L}'yi 3PIP tasarımcısına iletecektir.

5.2.3 3PIP kullanıcısının yapacağı işlemler

3PIP tasarımıcısı müşterisine mantıksal kilitleme yapılmış 3PIP-L dosyasını iletirken P_{km} ve P_{ka} parametreleri ile 3PIP-L dosyasının imza değeri S_{3PIP-L}'yi de iletir. 3PIP kullanıcısı kendi IP tasarımına 3PIP-L 'yi aşağıda tanımlandığı gibi dahil eder.

- 3PIP ve LUL Modülünün Kullanım Tanımlaması:
  Şekil 5.3'te 3PIP-L dosyası ve LUL modülünün nasıl kullanıldığı gösterilmektedir. Geliştirme aracı içinde sunulan hazır LUL modülü FPGA projesine eklenir. 3PIP-L dosyasının örneklemi ile LUL modülü bağlantıları Şekil 5.3'te gösterildiği gibi

yapılır. Gösterilen dışında bağlantı yapılamaz. P_{km} ve P_{ka} parametreleri ile 3PIP-L dosyasının imza değeri S_{3PIP-L} LUL modülü içine yazılır.



Şekil 5.3: 3PIP-L 'nin LUL modülü ile birlikte kullanımı.

- FPGA IP Tasarım Sürecinde 3PIP'nin Test Edilebilirliği:
  3PIP kullanıcısı 3PIP dahil tasarımını test etmek isterse bu aşamada 3PIP-L dosyası henüz doğru çalışmadığı için 3PIP test edilemeyecektir. Ancak 3PIP tasarımcısı 3PIP-L devresi için test vektörleri hazırlayabilir. Bu şekilde 3PIP-L dosyası ile test gerçekleştirilebilir.

- 3PIP İçeren FPGA IP Tasarımının Sentezlenmesi:
  3PIP kullanıcısı kendi FPGA IP sini tamamladıktan sonra sentezleme işlemini gerçekleştirmek istediğinde FPGA geliştirme aracı bir LUL modül kullanımı ve 3PIP-L dosyası varsa önce LUL modüle yazılan S_{3PIP-L} imza değerini denklem (5.9)' da gösterildiği gibi doğrulayacaktır. İmza doğrulama başarılı ise sentezleme işlemine devam edilir ve bit akış dosyası oluşturulur. İmza doğrulama başarısız olursa sentezleme işlemi durdurulur.

5.3 Önerilen Sistemin Doğru Çalıştığının İspatlanması

Bu bölümde önerilen kripto mimarinin ve güvenlik modülünün doğru çalışacağını göstermek için VHDL dili kullanılarak yapılan tasarım ve benzetim çalışmaları anlatılmıştır.

5.3.1 Örnek 3PIP devresine mantıksal kilitleme uygulanması

Çalışmada 3PIP devresi olarak ITC'99 benchmark devrelerinden faydalanılmıştır [35]. [36]'da benchmark devreleri detaylı olarak tanımlanmaktadır. Bu çalışmada kavramsal

kanıt gösteriminde kolay ve basit gerçekleme için B01 benchmark devresi (b01.vhd) kullanılmıştır. Mantıksal kilitleme için "NEOS" [37] kullanılmıştır.

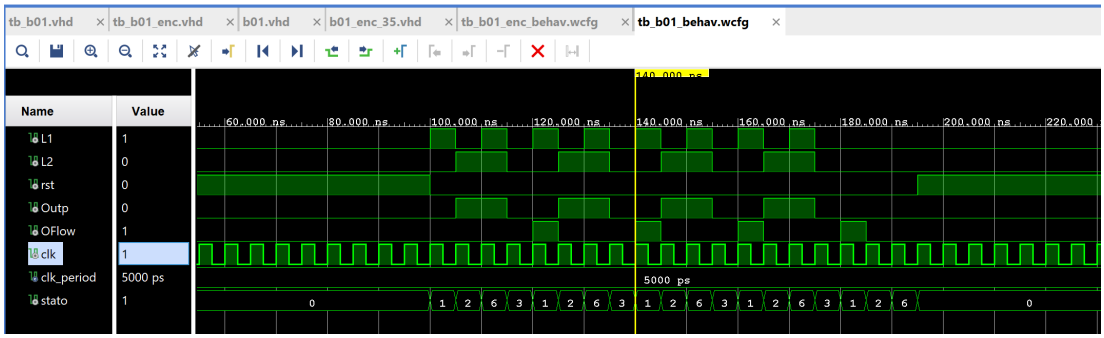
B01 benchmark devresine kapı sayısı dikkate alınarak 35 bit uzunluğunda bir anahtar ile mantıksal kilitleme yapılmıştır. Mantıksal kilitleme yapılmış 3PIP devresi "b01_enc.vhd" olarak isimlendirilmiştir. Mantıksal kilitleme "b01.bench" netlist dosyasına yapılmış daha sonra "b01_enc.bench" netlist dosyasından "b01_enc.vhd" dosyası elde edilmiştir.

Mantıksal kilitlemenin doğru uygulandığı benzetim ortamında test edilmiştir. Geliştirme ve benzetim için AMD firmasının Vivado programı 2024.2 versiyonu kullanılmıştır. Bu amaçla orijinal b01 devresi belirli süre farklı giriş değerleri uygulayacak test bench devresi ile çalıştırılmış, daha sonra aynı giriş değerleri b01_enc devresine doğru anahtar ve yanlış anahtar değeri ile birlikte uygulanmıştır. Doğru anahtar değeri ile çalıştırıldığında orijinal devre ile aynı çıkış değerini verdiği görülmüştür. Benzetim sonuçlarına ait ekran görüntüleri Şekil 5.4'te verilmiştir.

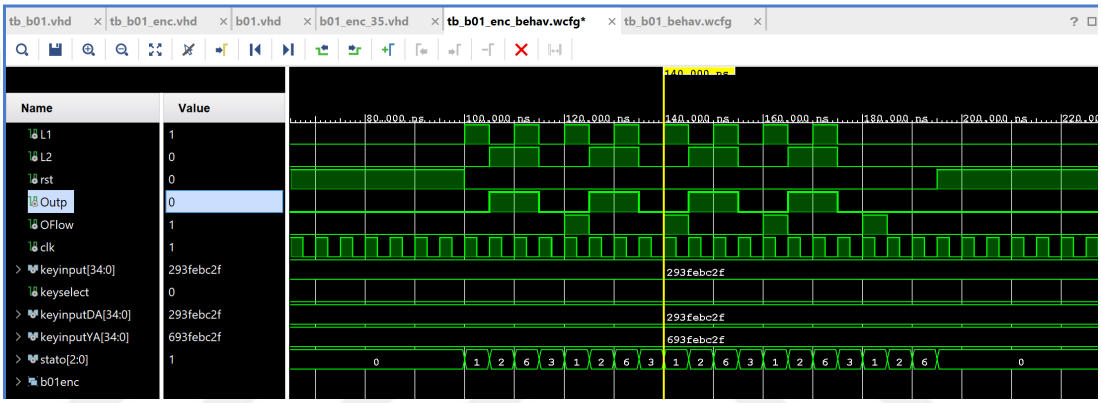
5.3.2 P_{km} ve P_{ka} parametrelerinin elde edilmesi

Bu aşamada P_{km} ve P_{ka} parametreleri elde edilmiştir. Öncelikle Vivado programı kullanılarak b01_enc devresinden K0 anahtarı ile 4 bitlik I0 sayaç değeri verilerek çıkış değeri elde edilmiştir. PyCharm 2025 programında Python hash kütüphanesi kullanılarak elde edilen çıkış değerinin HMACSHA256 özet değeri hesaplanmıştır. Hesaplanan özet değerinin düşük anlamlı 35 biti kullanılarak P_{km} parametresi elde edilmiştir. Daha sonra Vivado programında b01_enc devresinden K anahtarı ile 4 bitlik I0 sayaç değeri verilerek tekrar çıkış değeri elde edilmiştir. PyCharm 2025 programında Python hash kütüphanesi kullanılarak elde edilen çıkış değerinin HMACSHA256 özet değeri P_{ka} hesaplanmıştır.

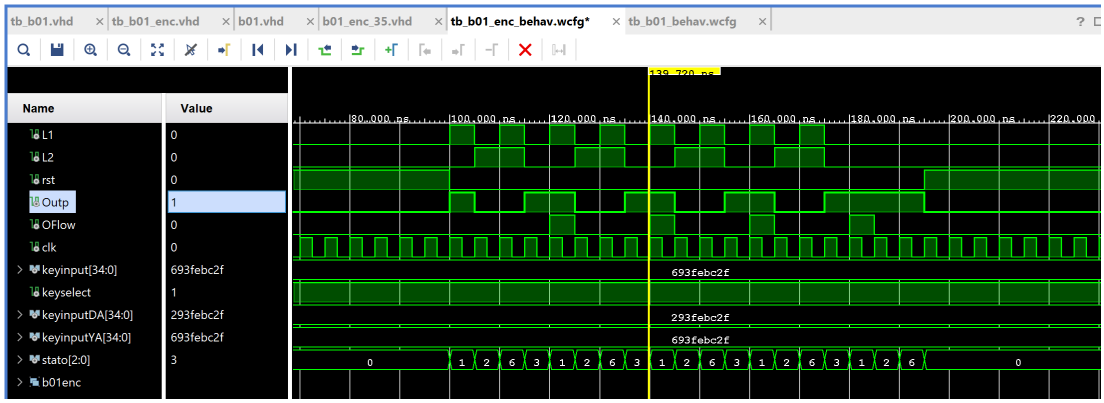
Hesaplanan değerler:
P_{km} : b'01001000010011010111000011011100100'
P_{ka} : X"2a274d9d6779820dc9e0ac2d2011f0f578022f9710271a86b614fb6e4bf1a6f9"



(a) b01.vhd devresinin çalışması.



(b) b01_enc.vhd devresinin doğru anahtar değeri ile çalışması key="01010010011111111101011110000101111".



(c) b01_enc.vhd devresinin yanlış anahtar değeri ile çalışması key="11010010011111111101011110000101111".

Şekil 5.4: b01.vhd ve b01_enc.vhd devrelerinin test edilmesi.

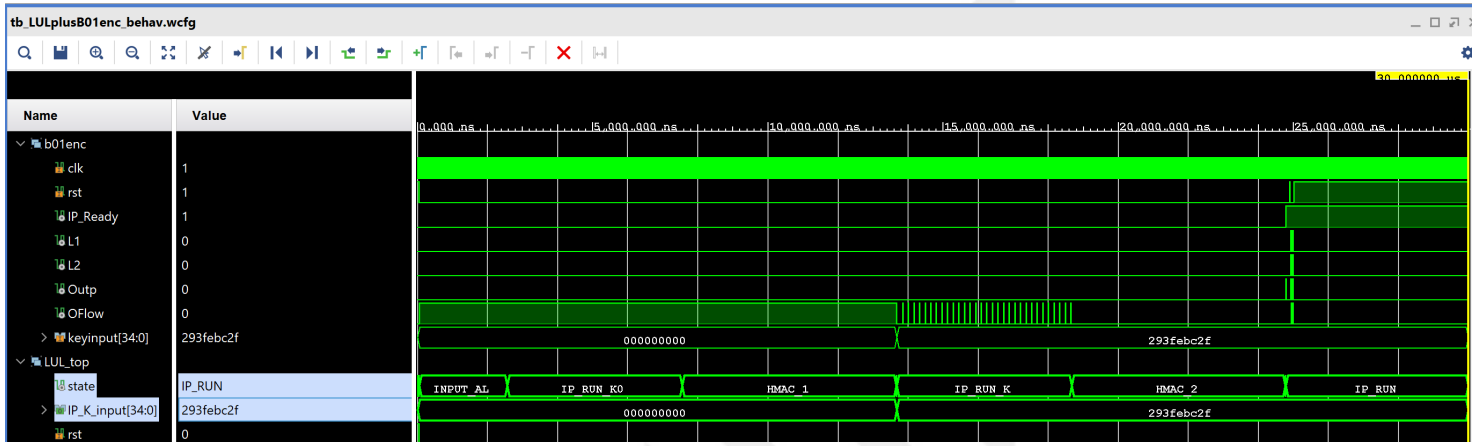
5.3.3 Mantıksal kilitlenmiş devrenin çalıştırılması

Çalışmada önerilen güvenlik modülü (LUL modülü) tasarımı VHDL dilinde yapılmıştır. Hazırlanan LUL modülü ve b01_enc devresi bir top modülde birlikte kullanılarak LUL modül tarafından b01_enc devresi giriş ve çıkışlarının kontrol edilmesi sağlanmıştır. Nihayet bir test bench devresi ile sistem test edilerek çalışması benzetim ortamında görüntülenmiştir. LUL modülü durum makinesi (RESET, IDLE, INPUT_AL, IP_RUN_K0, HMAC_1, IP_K_HESAPLA, IP_RUN_K, HMAC_2, PKA_CHECK, IP_RUN, IP_STOP) durumlarını içermektedir. "n" değeri 1000 olarak belirlenmiştir. Önceki adımda hesaplanan P_{km} ve P_{ka} parametreleri sabit olarak LUL modülüne yazılmıştır.

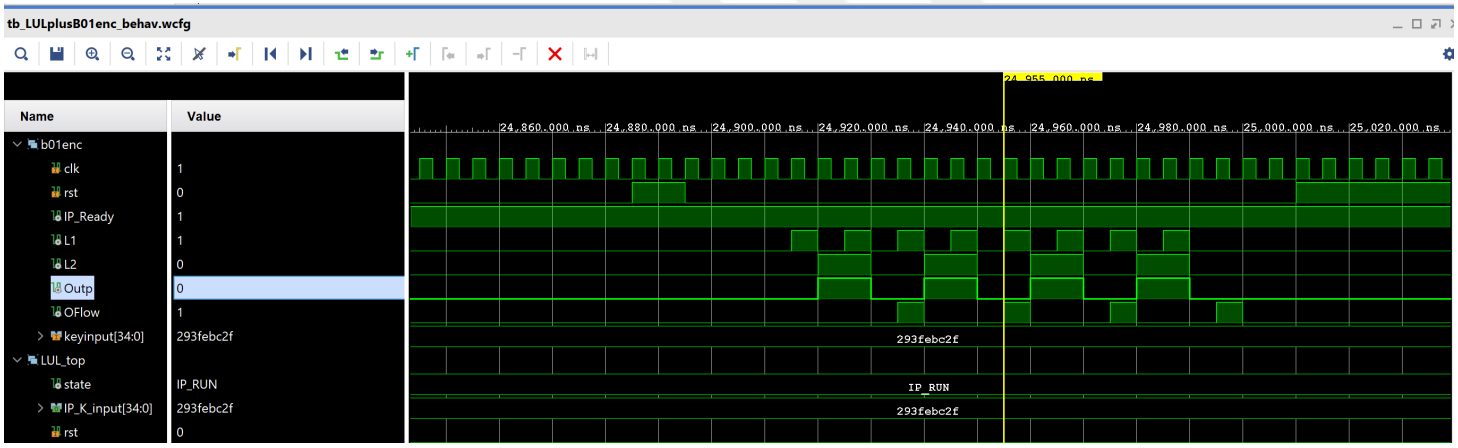
RESET sonrasında INPUT_AL durumunda 4 bitlik I0 sayaç değeri 2000 bit uzunluğunda kaydedilmektedir. Daha sonra IP_RUN_K0 durumunda K0 ile sürülen b01_enc devresine 1000 saat işareti süresinde kaydedilen input değeri verilmekte ve 1000 bit uzunluğunda çıkış değeri kaydedilmektedir. Ardından HMAC_1 durumunda kaydedilen çıkış değerinin K_{FPGA} anahtarı kullanılarak HMAC özet değeri hesaplanmaktadır. IP_K_HESAPLA durumunda özet değerinin düşük anlamlı 35 biti ve P_{km} parametresi kullanılarak K anahtarı elde edilmektedir.

IP_RUN_K modunda b01_enc devresi K anahtarı ile sürülürken başta toplanan 2000 bitlik giriş verisi uygulanarak yeni 1000 bitlik çıkış verisi elde edilmektedir. HMAC_2 durumunda elde edilen çıkış değerinin K_{FPGA} anahtarı kullanılarak HMAC özet değeri hesaplanmaktadır. PKA_CHECK durumunda özet değeri ve P_{ka} parametresi XOR edilir ve sonuç 0 ise IP_RUN durumuna geçer. Sonuç 0 dan farklı ise IP_STOP durumuna geçer ve b01_enc devresi girişi ve çıkışı 0 olarak sürülür.

Test çalışmasına ait benzetim ekran görüntüleri verilmiştir. Şekil 5.5'te LUL modülünde K anahtarının doğru elde edilmesine ait benzetim ekranı görülmektedir. IP_RUN durumunda b01enc devresine giriş verilerek beklendiği gibi çalıştığı görülmektedir. Şekil 5.6'da LUL modülünde hatalı K elde edilip b01enc devresine uygulanıp P_{ka} parametresinin doğrulanamadığı ve IP_STOP durumuna geçtiği görülmektedir. IP_STOP durumunda b01enc devresinin çalışmadığı görülmektedir.

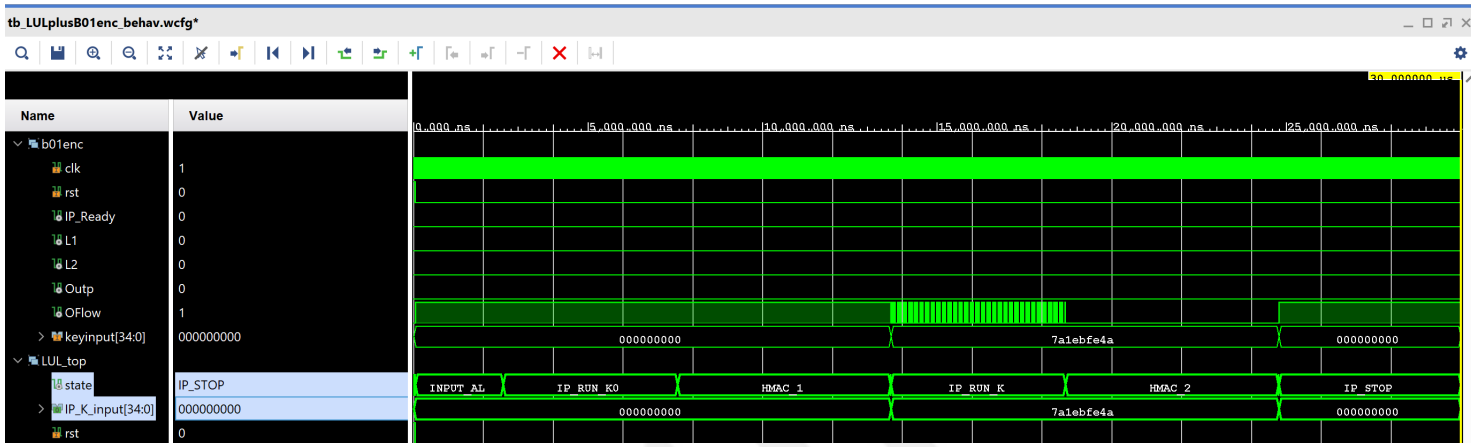


(a) LUL Modülü durum makinesi IP_RUN durumu.

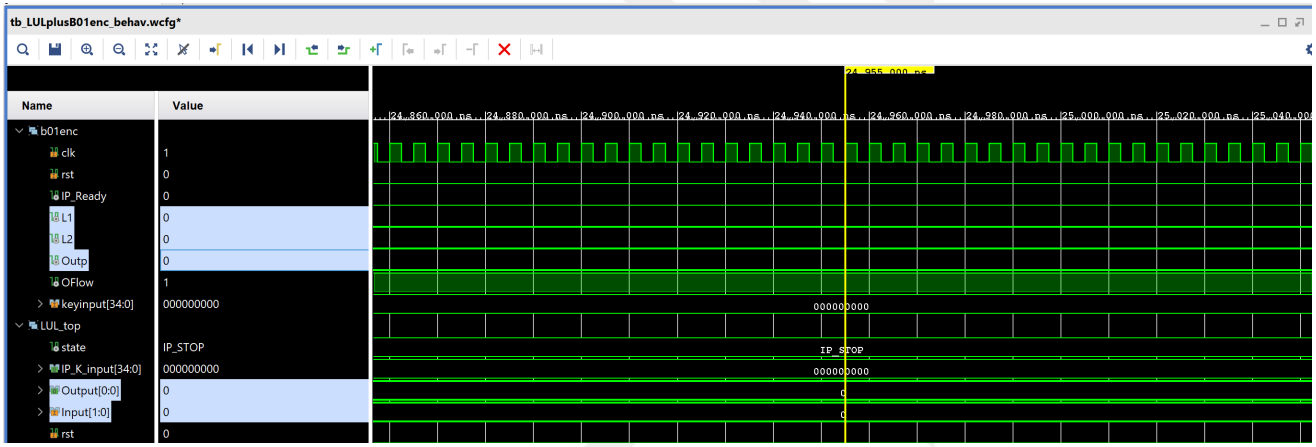


(b) IP_RUN durumunda b01enc devresinin doğru çalıştığının kontrolü.

Şekil 5.5: LUL Modülünde b01enc devresinin doğru anahtar elde edilerek çalışması.



(a) LUL Modülü durum makinesi IP_STOP durumu.



(b)IP_STOP durumunda b01enc devresinin çalışmadığının kontrolü.

Şekil 5.6: LUL Modülünde b01enc devresinin yanlış anahtar elde edilerek durdurulması.

5.4 Mantıksal Kilitleme Tekniğinin 3PIP Devrelere Etkisi ve LUL Modülünün Performans Özellikleri

Bu bölümde önce mantıksal kilitleme tekniğinin 3PIP devrelerin alan, güç ve zaman performansını nasıl etkilediği incelenecektir. Daha sonra LUL modülü ile mantıksal kilitlenmiş devrenin birlikte kullanım durumlarında tasarımı yapılan LUL modülünün performans değerleri farklı devreler için gösterilmiştir.

Performans analizi için ITC'99 [36]'da verilen benchmark devrelerinden farklı büyüklükte olan 4 tanesi (B14, B22, B18, B19) kullanılmıştır. B14 devresi Viper işlemcisinin alt bir kümesinin gerçeklemesidir. B15 devresi ise 80386 işlemcisi alt küme gerçeklemesidir. B22, B18 ve B19 devreleri B14 ve B15 devrelerinin farklı sayıda ve değiştirilmiş gerçeklemelerinden oluşturulmuştur. Bu devrelere ait devre büyüklükleri Çizelge 5.1'de gösterilmektedir. Bu devrelere mantıksal kilitleme uygulamak için [37]'de verilen NEOS programı kullanılmıştır. Dört devrede de 128 bit uzunluğunda anahtar olacak şekilde mantıksal kilitleme uygulanmıştır. Kilitleme işlemi devrenin ".bench" dosyası üzerinde uygulanmış, daha sonra ABC programı [38] yardımıyla Verilog dosyasına çevrilmiştir. Benchmark devrelerde clock ve reset sinyali bulunmadığından verilog dosyalarına bu sinyaller eklenmiştir.

Çizelge 5.1: Performans Karşılaştırılmasında Kullanılan Benchmark Devrelerin İçerdiği Yapısal Eleman Sayıları.

Yapısal Eleman	B14	B22	B18	B19
Giriş	32	32	37	24
Çıkış	54	22	23	27
D-FlipFlops	245	735	3320	6642
Evirici	1531	4491	20372	41107
Kapı	8567	25460	94249	190213

5.4.1 Mantıksal kilitleme tekniğinin 3PIP devrelerin FPGA kaynak gereksinimlerine etkisinin incelenmesi

Mantıksal kilitlemenin benchmark devresine olan etkisini anlayabilmek için benchmark devresinin orijinal hali ve kilitlenmiş hali herhangi bir ilave olmaksızın ayrı ayrı Vivado 2024.2 programında proje olarak oluşturulmuştur. Sentez ve

implementasyon adımları gerçekleştirilmiştir. Proje oluşturulurken board olarak Kria KV260 Vision AI Starter Kit SOM kullanılmıştır. Board üzerinde yer alan device modeli "xck26-sfvc784" olarak belirtilmektedir. Toplanan veriler post-implementation raporlarından elde edilmiştir.

Çizelge 5.2'de oluşturulan projelerin zaman analizine ilişkin veriler yer almaktadır. Projelerde ilk olarak saat işareti periyodu 10 ns (100 MHz) olarak belirlenmiş ve tüm projelerin bu frekansta çalıştıkları gözlenmiştir. Daha sonra periyot 1 ns çözünürlükle düşürülerek çalışabildiği en yüksek frekans belirlenmiş ve bu şekilde sentezleme ve implementasyon yapılarak sonuçlar alınmıştır.

Çizelge 5.2: Test Edilen Benchmark Devrelerin Zamanlama Analizi Sonuçları.

Devre Adı	Çalışma Frekansı (MHz)	Worst Negative Slack (ns)	Worst Hold Slack (ns)	Worst Pulse Width Slack (ns)
B14	200	0,464	0,101	2,225
B14-Kilitli	200	0,085	0,051	2,225
LUL ‖ B14-Kilitli	200	0,376	0,010	2,225
B22	200	0,164	0,068	2,225
B22-Kilitli	200	0,030	0,063	2,225
LUL ‖ B22-Kilitli	200	0,066	0,010	2,225
B18	142,86	0,009	0,037	3,225
B18-Kilitli	125	0,111	0,008	3,725
LUL ‖ B18-Kilitli	125	0,148	0,010	3,725
B19	111,11	0,199	0,010	4,225
B19-Kilitli	125	0,128	0,021	3,725
LUL ‖ B19-Kilitli	125	0,093	0,010	3,725

Zaman analizinde çalışma frekansı, WNS (worst negative slack: en kritik ve en kötü yolun zamanlama hatası), WHS (worst hold slack: en erken gelen verinin zamanlama hatası), WPWS (worst pulse width slack: en kötü (en negatif) darbe genişliği farkı) değerleri çizelgede gösterilmiştir. Her üç değer için toplam değerleri ifade eden TNS, THS ve TPWS değerleri raporlarda sıfır olarak hesaplanmış, bu nedenle çizelgeye konulmamıştır. Orijinal devrelerde devre büyüklüğü arttıkça çalışma frekansının düştüğü görülmektedir. Mantıksal kilitleme işleminin frekans üzerinde sistematik bir etkisinin olmadığı, B18 devresinin kilitli halinde çalışma

frekansı düşerken B19 devresinde kilitleme yapılmış devrenin daha yüksek frekansta çalışabildiği görülmektedir. Ayrıca büyük ve karmaşık devrelerde WPWS değerinin arttığı görülmektedir.

Benchmark devrelerin orijinal hali ve mantıksal kilitleme yapılmış haline ait FPGA kaynak kullanımları Çizelge 5.3'te gösterilmektedir. Çizelgede kaynak olarak sadece kullanılan CLB LUT ve CLB Register adetleri gösterilmiştir.

B14 devresine mantıksal kilitleme uygulandığında CLB LUT ve CLB Register sayısında değişim oranları sırasıyla %18 ve %0 olmuştur. Aynı değerler en büyük devre olan B19 devresinde %1 ve %0 şeklinde gerçekleşmiştir. Benchmark devrelere mantıksal kilitleme uygulanmasının CLB Register sayısında bir artışa neden olmadığı görülmüştür. CLB LUT bakımından küçük devrelerde görece yüksek oranda artış görünürken, devre büyüklüğü arttıkça kilitleme işleminden kaynaklanan maliyet artışının yüzde olarak sıfır seviyesine düştüğü gözlenmiştir. Büyük devrelerde kilitleme için eklenecek yapıların oranı mevcut devreye göre halihazırda azalırken, sentez ve uygulama sırasında optimizasyon sonucu bu oranın sıfıra indiği söylenebilir.

Benchmark devrelerin orijinal hali ve kilitli haline ait tahmini güç tüketim maliyet değerleri Çizelge 5.4'te verilmiştir. Güç maliyeti olarak orijinal ve kilitli devreler karşılaştırıldığında mantıksal kilitlemenin önemli bir etki oluşturmadığı gözlenmiştir. Devre büyüdükçe güç maliyetinde bir miktar azalma olduğu görülmektedir. Bunun sentez ve uygulama sırasında gerçekleşen optimizasyon sonucunda kaynak kullanımının azalmasından kaynaklandığı söylenebilir.

5.4.2 LUL modülünün FPGA kaynak gereksinimi ve 3PIP devrelere etkisinin incelenmesi

LUL Modülünün kullanıldığı projelerin (LUL+) performans değerleri Çizelge 5.2, 5.3 ve 5.4'te verilmiştir. LUL modülü ile mantıksal kilitleme yapılmış benchmark devrelerin birlikte kullanılarak oluşturulan projelerinden bu veriler elde edilmiştir.

Projeler oluşturulurken benchmark devrelerinin farklı giriş-çıkış sayısına sahip olmaları nedeniyle basit bir düzenleme yapılmıştır. LUL modülü çalışırken benchmark devresi 1000 saat işareti boyunca çalıştırılmış, benchmark devre çıkışından alınan çıkış

Çizelge 5.3: Test Edilen Benchmark Devrelerin Orijinal Durumu, Kilitli Durumu ve LUL Modülü ile Birlikte Kullanım Durumlarına Ait FPGA Donanım Kaynağı Gereksinimleri.

Device: xck26-sfvc784 — CLB LUT Adedi: 117120 — CLB Register Adedi: 234240

Devre Adı	CLB LUT		CLB Register	
	Adet	(%)	Adet	(%)
B14	1059	100	215	100
B14-Kilitli	1245	118	214	100
LUL ‖ B14-Kilitli	9398	887	17134	7987
B14-Kilitli	1487	14	216	1
LUL Modülü	7915	86	16956	99
LUL → Hashcore	7753	84	14777	86
LUL ‖ B14-Kilitli*	1645	155,3	2395	1114
B22	3342	100	613	100
B22-Kilitli	4187	125	613	100
LUL ‖ B22-Kilitli	12077	361	17570	2866
B22-Kilitli	4228	35	613	3
LUL Modülü	7850	65	16957	97
LUL → Hashcore	7742	64	14777	84
LUL ‖ B22-Kilitli*	4335	129,7	2793	455,6
B18	14564	100	3033	100
B18-Kilitli	14445	99	3032	100
LUL ‖ B18-Kilitli	22740	156	19985	659
B18-Kilitli	12496	55	3017	15
LUL Modülü	10245	45	16968	85
LUL → Hashcore	7762	34	14779	74
LUL ‖ B18-Kilitli*	14978	102,8	5206	171,6
B19	28553	100	6061	100
B19-Kilitli	28801	101	6055	100
LUL ‖ B19-Kilitli	36704	129	23013	380
B19-Kilitli	27399	74	6055	26
LUL Modülü	9312	26	16958	74
LUL → Hashcore	7762	21	14779	64
LUL ‖ B19-Kilitli*	28942	101,4	8234	135,9

*: Hashcore modülü maliyetleri çıkarılmıştır.

verilerinin en düşük anlamlı ikinci bitleri toplanarak 1000 bit uzunluğunda HMAC algoritması giriş verisi oluşturulmuştur.

Çizelge 5.4: Test Edilen Benchmark Devrelerin Hesaplanan Güç Tüketim Değerleri.

Devre Adı	Toplam Güç		Statik Güç		Dinamik Güç	
	(W)	(%)	(W)	(%)	(W)	(%)
B14	0,440	100	0,289	66	0,151	34
B14-Kilitli	0,462	100	0,289	63	0,173	37
LUL ‖ B14-Kilitli	0,688	100	0,290	42	0,398	58
B14-Kilitli					0,034	5
LUL Modülü					0,346	50
LUL → Hashcore					0,339	49
B22	0,495	100	0,289	58	0,206	42
B22-Kilitli	0,519	100	0,289	56	0,230	44
LUL ‖ B22-Kilitli	0,828	100	0,291	35	0,537	65
B22-Kilitli					0,169	20
LUL Modülü					0,326	39
LUL → Hashcore					0,318	38
B18	0,497	100	0,289	58	0,208	42
B18-Kilitli	0,464	100	0,289	62	0,175	38
LUL ‖ B18-Kilitli	0,664	100	0,290	44	0,374	56
B18-Kilitli					0,118	18
LUL Modülü					0,229	34
LUL → Hashcore					0,206	31
B19	0,609	100	0,290	48	0,319	52
B19-Kilitli	0,599	100	0,290	48	0,309	52
LUL ‖ B19-Kilitli	0,792	100	0,291	37	0,501	63
B19-Kilitli					0,232	29
LUL Modülü					0,234	30
LUL → Hashcore					0,215	27

Projeler ilk olarak 10 ns saat işareti ile çalıştırılmış ve tüm devrelerin bu frekansta çalıştığı görülmüştür. Daha sonra yukarıda ifade edildiği gibi 1'er ns azaltılarak denemeler yapılmış ve en yüksek çalışma frekansı belirlenmiştir.

Çizelge 5.2'de LUL+ projelerine ait zaman analizi verileri gösterilmiştir. Çizelgeden görüldüğü üzere WNS ve WHS değerlerinin azaldığı, ancak çalışma frekansını etkilemediği görülmektedir. FPGA tasarımında WNS ve WHS değerlerinin yüksek olması daha tercih edilen bir sonuçtur.

Çizelge 5.3'te görüldüğü gibi LUL+ olarak gerçekleştirilen projelerde kullanılan CLB LUT miktarında artış olmaktadır. Artış miktarı devrelerde adet olarak 8150-8750 aralığında değişmektedir. Artış oranı devre büyüklüğü arttıkça belirgin şekilde azalmaktadır (%887 -> %129). LUL modülünde CLB LUT maliyetini yaklaşık 7750 adet ile Hashcore modülü (HMAC algoritması) oluşturmaktadır. Çoğu FPGA'de HMAC algoritması hazır olarak bulunmaktadır. Hashcore modülü hariç tutulduğunda LUL modülünün getirdiği CLB LUT artışı B14 devresinde %55 iken B19 devresinde %1'e düşmektedir. LUL modülü kullanımı ile kullanılan CLB Register miktarında büyük artış oluşmaktadır. Ancak maliyeti artışı büyük oranda Hashcore modülünden kaynaklanmaktadır. CLB Register sayısında ortalama 16960 adet artış gerçekleşirken bunun 14777 adedi Hashcore modülü için kullanılmaktadır. Hashcore modülü hariç tutulduğunda CLB Register maliyet artışı %1114'ten %35'e inmektedir. LUL modülü kaynaklı CLB Register maliyeti Hashcore modülü hariç ortalama 2180 adet olup devreye göre değişmemektedir. Çoğu FPGA modeli gömülü HMAC algoritması içermektedir. LUL modülü kolaylıkla hazır HMAC algoritmasını kullanacak şekilde düzenlenebilir.

Çizelge 5.4'te LUL+ projelerde toplam güç tüketiminde bir artış olduğu görülmektedir. Ancak bu projelerde alt modül olarak yer alan benchmark devrelerinin güç tüketimlerinin, orijinal benchmark devrelerine kıyasla daha düşük olduğu gözlemlenmektedir. LUL modülünde kaynak kullanımının büyük bir kısmının Hashcore modülünden kaynaklandığı dikkate alındığında, Hashcore modülü hariç tutulduğunda LUL+ tasarımının dinamik güç tüketiminin, orijinal benchmark devresine göre daha düşük olduğu anlaşılmaktadır. Bu durum LUL modülünün kendi başına güç tüketimine belirgin bir olumsuz etkisinin bulunmadığını göstermektedir.

Ayrıca, kriptografik mimarinin doğası gereği LUL modülünün yalnızca reset sonrasında kısa bir süre aktif olarak çalışması, güç tüketimi üzerindeki etkisinin ihmal edilebilir düzeyde olmasına neden olacaktır. LUL modülüne ait durum makinesinin IP_RUN durumuna ulaşması, 100 MHz çalışma frekansında yaklaşık 45 µs sürmektedir. Bu durum LUL modülünün toplam çalışma süresi içindeki etkinliğinin oldukça sınırlı olduğunu ortaya koymaktadır.

6. GÜVENLİK ANALİZİ VE KARŞILAŞTIRMA

6.1 Sistemde Bulunan Kritik Varlıklar ve Tehdit Senaryoları

Tanımlanan sistemde bulunan korunması gerekli kritik varlıklar aşağıda listelenmiş ve açıklanmıştır.

- K_{FPGA} : FPGA üreticisi tarafından üretilmiş özel anahtar. Her FPGA üzerinde üretimde güvenli bellek alanına yazılmaktadır. P_{km} ve P_{ka} parametrelerinin hesaplanmasında anahtarlı HMAC algoritmasında kullanılır.
- K : 3PIP tasarımcısı tarafından belirlenir. 3PIP netlistinin mantıksal kilitleme yöntemiyle kilitlenmesi için kullanılır. Tasarımcı ve FPGA üreticisi tarafından bilinmekte olup korunmalıdır.
- $K_{özel}$: FPGA üreticisine ait özel anahtardır. 3PIP-L dosyalarının imzalanması amacıyla kullanılmaktadır. Sadece FPGA üreticisi tarafından bilinmekte olup korunmalıdır.

6.1.1 K_{FPGA} anahtarının elde edilme durumu

K_{FPGA} anahtarı elde edilirse K anahtarı da elde edilebilir. Bu durumda sistemin sağlayacağı güvenlik sağlanamaz. Bu nedenle K_{FPGA} anahtarı gizli tutulmalıdır. Anahtar sadece FPGA üreticisi tarafından bilinmekte ve her FPGA üzerinde güvenli bellekte tutulmaktadır. Güvenli bellekten okunamayacağı varsayılmaktadır. Çalışma sırasında LUL modülünden dışarı çıkarılması mümkün değildir. Ancak anahtarlı HMAC algoritması çalıştırılırken yan kanal analizi yapılarak elde edilmesi söz konusu olabilir. Bu nedenle kullanılacak HMAC algoritması yan kanal analizine dayanıklı olacak şekilde gerçeklenmeli ve kullanılmalıdır. Bu çalışma kapsamında yan kanal analizi yapılmamıştır.

6.1.2 K anahtarının elde edilme durumu

K anahtarı 3PIP-L nin 3PIP haline dönmesini sağlamaktadır. Sistemden K anahtarı elde edilirse sistem güvenlik sağlayamaz. K anahtarı FPGA çalışırken LUL modülü içinde oluşmakta ve 3PIP-L devresine giriş olarak verilmektedir. K anahtarının elde edilmesi için aşağıdaki senaryoların olabileceği belirlenmiş ve her senaryoda güvenlik sağlanıp sağlanmadığı değerlendirilmiştir.

6.1.2.1 Sahte 3PIP-L kullanım senaryosu

Saldırgan K anahtarını elde etmek için sahte 3PIP-L dosyası oluşturabilir. Öyle ki bu sahte 3PIP-L devresi K anahtarı oluşturulup 3PIP-L devresine uygulandıktan sonra anahtarı çıkış olarak dışarı verecektir. Saldırı analiz edildiğinde önce sahte 3PIP-L devresi içeren FPGA IP 'si sentezlenebilmesi gerekir. Ancak sentezleme öncesinde 3PIP-L dosyasının imzası S_{3PIP-L} doğrulanmaktadır. Sahte 3PIP-L dosyasının imzası doğrulanamayacağı için sentezleme gerçekleşmeyecektir. Saldırganın imza doğrulama adımını bir şekilde aştığını ve sentezlemeyi başardığını varsayılırsa bu durumda sahte 3PIP-L devresinden elde edilecek O_{0n} çıkış verisi farklı olacaktır. Farklı O_{0n} verisinin özet değeri de farklı hesaplanacak ve P_{km} parametresi ile xor edildiğinde farklı bir K' anahtarı elde edilecektir. Dolayısıyla bu saldırı başarısız olacaktır.

6.1.2.2 Öykünücü 3PIP-L kullanım senaryosu

Saldırgan K anahtarını elde etmek için öykünücü 3PIP-L dosyası oluşturabilir. Öykünücü-3PIP-L devresi içinde bulunan bellekte orijinal 3PIP-L devresinin vereceği O_{0n} ve O_{1n} çıkış verisinin bulunması gerekir. Öyle ki bu öykünücü-3PIP-L devresi çıkış olarak içinde bellekte bulunan O_{0n} ve O_{1n} dizilerini dışarı verecek, K anahtarı oluşup çalışma moduna geçtiğinde çıkış verisi olarak K anahtarını çıkış olarak verecektir. O_{0n} çıkış verisi bilinmektedir ancak O_{1n} verisi bilinmemektedir. O_{1n} verisini sağlamak için taklitçi IP içine orijinal 3PIP-L devresi yerleştirilir. Bu durumda saldırı sonraki senaryoda belirtilen saldırıya evrilmektedir. Dolayısıyla saldırı bu

şekilde gerçekleştirilemez. Ayrıca öyükünücü 3PIP-L dosyası imza doğrulamayı da geçemeyecek ve başarısız olacaktır.

6.1.2.3 Truva atı ve 3PIP-L içeren T-3PIP-L kullanım senaryosu

Saldırgan K anahtarını elde etmek için 3PIP-L dosyasını ve truva atı içeren bir T-3PIP-L oluşturabilir. Öyle ki; bu T-3PIP-L devresi K anahtarı oluşturulup P_{ka} doğrulandıktan sonra truva atı devreye girecek ve K anahtarını dışarıya verecektir. Saldırı analiz edildiğinde T-3PIP-L devresinin sentezleme öncesinde imza doğrulama işleminde başarısız olacağı görülmektedir. Dolayısıyla bu saldırı gerçekleştirilemeyecektir.

6.1.2.4 SAT saldırısı senaryosu

3PIP-L modülü mantıksal kilitleme uygulanarak kilitlenmiş durumdadır. SAT saldırısı mantıksal kilitlemeyi kırabilmektedir. Saldırgan K anahtarını elde etmek için SAT saldırısı yapmayı deneyebilir. Ancak SAT saldırısının gerçekleştirilebilmesi için ASIC devrelerde tarama zincirinden faydalanılmaktadır. Üretim ve test aşamalarını geçmiş ASIC devrelere mantıksal kilitlemeyi çözecek K anahtarı yazılmaktadır. Saldırgan K anahtarı yüklenmiş bir ASIC devreyi elde edip SAT saldırısı için gerekli giriş-çıkış verilerini tarama zincirini çalıştırarak elde ederek saldırıyı gerçekleştirebilmektedir. FPGA IP sinin test yöntemi farklı olduğundan tarama zinciri gerekli değildir. Bu nedenle 3PIP-L ye SAT saldırısı gerçekleştirilemez.

6.2 Önerilen Yöntemin Alternatif Yöntemle Karşılaştırılması

Hutchings tarafından yapılan çalışmada [7], 3PIP korunmasına yönelik bir yöntem önerilmektedir. Bu nedenle söz konusu yöntem, bu çalışmada alternatif bir yaklaşım olarak ele alınmış ve önerilen sistem ile özellikler, uygulama durumu ve maliyet yükleri açısından karşılaştırılmıştır. Yapılan değerlendirmeler aşağıda sunulmaktadır.

Öncelikle, bu çalışmada önerilen sistem, güncel FPGA'lere uygulanabilir bir yapı değildir. Buna karşın alternatif yöntem, mevcut FPGA'lerde bulunan kısmi yeniden programlanabilir (partial reconfiguration) modeller üzerinde uygulanabilir durumdadır.

Alternatif yöntemin önemli bir avantajı, birden fazla 3PIP'nin eş zamanlı olarak korunabilmesine olanak sağlamasıdır. Buna karşılık, önerilen sistemde her bir 3PIP için ayrı bir LUL güvenlik modülü kullanılacak şekilde bir tasarım yapılmıştır. Ancak bu durum, sistem açısından temel bir kısıt oluşturmamaktadır. FPGA mimarilerinde birden fazla LUL modülü paralel kullanılarak daha fazla 3PIP'nin eş zamanlı korunması mümkündür. Ya da LUL modülü farklı 3PIP'leri sürecek şekilde düzenlenebilir.

Bu çalışmada önerilen sistemde, çözümün uygulanmasından sorumlu taraf FPGA üreticisidir ve bu taraf güvenilir kabul edilmektedir. 3PIP geliştiricisi ve kullanıcı dışında herhangi bir üçüncü tarafa ihtiyaç duyulmamaktadır. Bu yönüyle, sistem mimarisi daha sade ve doğrudan bir yapı sunmaktadır.

Buna karşılık, alternatif yöntemde geliştirici ve kullanıcıya ek olarak "Güvenilir Anahtar Tutucu" olarak tanımlanan üçüncü bir tarafın varlığı zorunludur. Bu gereklilik, uygulama zorluğunu artıran önemli bir faktör olarak öne çıkmaktadır. Alternatif yöntemin uygulanabilmesi için, 3PIP kullanıcısının FPGA'leri ya önceden anahtar yüklenmiş şekilde temin etmesi ya da FPGA'lerin satın alım sonrasında anahtar yükleme işlemi için güvenilir anahtar tutucuya gönderilmesi gerekmektedir.

Bu süreç, FPGA üzerinde dahili bir konfigürasyon flash belleği bulunması durumunda mümkün olabilmektedir. Ancak harici bellek kullanımı söz konusu olduğunda, dizgi (bitstream) aşamasından sonra harici belleğin programlanması için FPGA'in güvenilir anahtar tutucu tarafa gönderilmesi zorunlu hale gelmektedir. Bu durum, hem zaman hem de lojistik açıdan ek maliyetler doğurmaktadır.

Önerilen yöntemde ise, 3PIP geliştiricisinin FPGA üreticisi ile yalnızca bir kez tanımlı bir prosedürü yürütmesi yeterlidir. Bu özellik, yöntemin uygulama kolaylığını ve pratikliğini artırmaktadır.

Getirdiği kısıtlamalar açısından değerlendirildiğinde, bu çalışmada önerilen sistem FPGA özelliklerinin kısıtlama olmadan kullanılabilmesine imkan vermektedir. Buna karşın, alternatif yöntem FPGA'in kısmi yeniden programlanabilme (partial reconfiguration) özelliğini kullanmakta ve bu durum 3PIP kullanıcısının söz konusu özelliği kullanmasına imkan vermemektedir.

Alternatif yöntemde güvenlik, önceden yüklenecek Yükleyici uygulamasını içeren statik bitstream dosyasının FPGA bitstream şifreleme özelliği kullanılarak şifrelenmesi ile sağlanmaktadır. 3PIP'lere ait şifreleme anahtarları bu bitstream dosyası içerisinde yer almaktadır. Yükleyici uygulaması içeren bu şifreli bitstream'e ait anahtar, FPGA'nin konfigürasyon anahtar belleğine statik olarak kazınmakta ve söz konusu anahtar yalnızca güvenilir anahtar tutucu tarafından bilinmektedir.

Bu yaklaşım, 3PIP kullanıcısının kendi IP güvenliğini artırmak amacıyla kullanabileceği alternatif bir çözüm yolu bırakmamaktadır. Ayrıca, yan kanal analizi gibi yöntemlerle bu anahtarın elde edilmesi durumunda, tersine mühendislik teknikleri kullanılarak hem kullanıcıya ait IP'nin hem de 3PIP'lerin güvenliğinin ihlal edilmesi mümkün hale gelmektedir.

Önerilen yöntemde ise, klasik FPGA bitstream şifreleme işlemi kullanıcı tarafından gerçekleştirilebilmekte ve gerektiğinde ilave koruma adımları da kullanabilmektedir. 3PIP güvenliği, FPGA içerisine gömülü K_{FPGA} anahtarı kullanılarak sağlanmakta olup, 3PIP'lere ait mantıksal kilitleme anahtarları çalışma anında FPGA içinde dinamik olarak oluşturulmaktadır.

Çizelge 6.1'de, önerilen yöntem ile alternatif yöntem, getirdikleri kaynak maliyet artışı açısından karşılaştırılmıştır. Alternatif yöntemde testler için OpenCores'tan seçilmiş 3PIP'ler kullanılmış; bunların orijinal ve şifreli halleri maliyetleri açısından karşılaştırılmış ve statik kod (Yükleyici uygulamasını içeren) kısmına ait maliyet ayrı olarak verilmiştir. Maliyette önemli yer tuttuğu için burada yalnızca kullanılan LUT ve FF adedi alınarak karşılaştırma yapılmıştır. Bu çalışmada önerilen yöntem için FF yerine CLB Register sayısı alınmış olup, FF'ler bunun bir alt kümesini oluşturmaktadır.

Çizelge 6.1: Alternatif Yöntem ve Önerilen Yöntemin FPGA Donanım Kaynağı Gereksinimi Artış Oranları.

Devre Adı	Önerilen Yöntem		Alternatif Yöntem	
	LUT Artışı %	Reg. Artışı %	LUT Artışı %	FF Artışı %
Kilitli/Şifreli 3PIP	-1 / 25	–	-1 / 13,7	0 / 3,3
LUL+ / Loader+	29 - 787	280 - 7887	35 - 1251	46 - 1205

Çizelgeden görüldüğü gibi, test devreleri için LUT artış oranı önerilen yöntemde %-1–25 aralığında iken, alternatif yöntemde %-1–13,7 aralığındadır. Ancak bu artış alternatif yöntemde devre büyüklüğü ile doğru orantılı değildir. Önerilen yöntemde, devre büyüklüğü arttıkça artış oranı düşmektedir. Alternatif yöntem en büyük devrede %10 artışa sebep olurken önerilen yöntemde büyük devrelerde artış yerine azalma görülmektedir. Register olarak önerilen yöntem herhangi bir artış getirmezken, alternatif yöntemde FF sayısı %0–3,3 aralığında artışa sebep olmaktadır.

Yöntemlerin toplam kaynak maliyet karşılaştırması için alternatif yöntemde ayrı verilen statik kod maliyeti, şifreli 3PIP maliyeti ile toplanmıştır. Önerilen yöntem için Bölüm 5.4'te verilen LUL+ proje maliyetleri kullanılmıştır.

Çizelgede LUT artış oranı aralığı önerilen yöntemde %29–787, alternatif yöntemde %35–1251 olarak verilmiştir. Register/FF artış oranı ise önerilen yöntemde %280–7887, alternatif yöntemde %46–1205 şeklindedir. Önerilen yöntemde artışların büyük bir kısmı HMAC algoritmasından kaynaklanmaktadır. HMAC hariç tutulduğunda, artış oranları LUT için %1–55, Register için %35–1014 olarak hesaplanmıştır.

Çizelge 6.2'de, iki yöntem FPGA üzerinde çalışmaya hazır duruma gelme süresi açısından karşılaştırılmıştır. Alternatif yöntemde, sistemin hazır hale gelme süresi 166 ile 326 sn arasında değişmektedir. İlk olarak statik kod FPGA üzerine yazılmakta, ardından Loader tarafından asıl FPGA IP'si parça parça okunup çözülmekte ve FPGA programlanmaktadır. Bu süreler sistemler için çok uzun olarak değerlendirilebilir. Buna karşılık, önerilen yöntemde FPGA IP'si doğrudan programlanmakta ve reset sinyali sonrasında mikro saniyeler içinde çalışmaya hazır duruma gelmektedir.

Çizelge 6.2: Alternatif Yöntem ve Önerilen Yöntemin FPGA Üzerinde Çalışır Duruma Gelme Süreleri.

	Önerilen Yöntem	Alternatif Yöntem
Konfigürasyon Süresi	45 us*	166 sn / 326 sn

*: 100 MHz çalışma frekansında işlem süresi.

7. SONUÇLAR VE ÖNERİLER

Bu tezde FPGA projelerinde kullanılan 3PIP'lerin telif haklarını tasarımcısı dışında kalan; IP kullanıcısı veya başka taraflara karşı korumak için mantıksal kilitleme tabanlı kriptografik bir FPGA güvenlik mimarisi tasarımı yapılmıştır. Bu amaçla bir kripto mimari tasarımı ve bu kripto mimarinin yürütülmesini gerçekleştirecek güvenlik modülü tasarımı yapılmıştır.

Önerilen sistemin uygulayıcısının FPGA üreticileri olacağı varsayılmıştır. IP ve 3PIP tasarımcıları bu sistemde kullanıcı olacak ve sistem kullanıldığında 3PIP tasarımları IP hırsızlığına karşı korunmuş olacaktır. FPGA üzerinde fazla sayıda güvenlik modülü yerleştirildiğinde birden fazla 3PIP eş zamanlı olarak güvenlik sistemi ile korunarak aynı projede kullanılabilecektir.

Mevcut durumda 3PIP'lerin fazladan kullanımı ile ilgili bir korunma sağlamamaktadır. Gelecek çalışmalarda bu konuda iyileştirme yapılarak koruma kapsamı genişletilebilir. Ayrıca yöntem kapsamında FPGA geliştirme aracı tarafından yapılması gereken imza doğrulama mekanizmasının sağladığı güvenliği sağlayacak alternatif yöntemler üzerinde çalışılabilir. İmza algoritması ile hedeflenen güvenlik FPGA içinde kontrol edilerek uygulanabilirse önerilen sistemin sağladığı güvenlik artacaktır. İlave olarak HMAC algoritması için FPGA üzerinde yer alan HMAC blokları kullanılarak LUL modülü kaynak kullanımı iyileştirilebilir.

Öte yandan sistemde kullanılan RSA imza algoritması asimetrik algoritma olup kuantum hesaplamaya karşı dayanıklı değildir. Kuantum tehdidine karşı geliştirilmiş algoritmalar kullanılarak bu zayıflık ortadan kaldırılabilir.

KAYNAKLAR

[1] **Shamsi, K.**, **Li, M.**, **Plaks, K.**, **Fazzari, S.**, **Pan, D.Z. ve Jin, Y.** (2019). IP protection and supply chain security through logic obfuscation: A systematic overview, *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, *24*(6), 1–36.

[2] **Rostami, M.**, **Koushanfar, F. ve Karri, R.** (2014). A primer on hardware security: Models, methods, and metrics, *Proceedings of the IEEE*, *102*(8), 1283–1295.

[3] **Group, I.E.W.** *ve diğerleri* (2015). *IEEE 1735-2014: IEEE Recommended Practice for Encryption and Management of Electronic Design Intellectual Property (IP)*.

[4] **Zhang, T.**, **Wang, J.**, **Guo, S. ve Chen, Z.** (2019). A comprehensive FPGA reverse engineering tool-chain: From bitstream to RTL code, *IEEE Access*, *7*, 38379–38389.

[5] **Duncan, A.**, **Rahman, F.**, **Lukefahr, A.**, **Farahmandi, F. ve Tehranipoor, M.** (2019). FPGA bitstream security: a day in the life, *2019 IEEE International test conference (ITC)*, IEEE, s.1–10.

[6] **Kamali, H.M.**, **Azar, K.Z.**, **Farahmandi, F. ve Tehranipoor, M.** (2022). Advances in logic locking: Past, present, and prospects, *Cryptology ePrint Archive*.

[7] **Hutchings, D.**, **Taylor, A. ve Goeders, J.** (2024). Toward FPGA Intellectual Property (IP) Encryption from Netlist to Bitstream, *ACM Transactions on Reconfigurable Technology and Systems*, *18*.

[8] **Yasin, M.**, **Rajendran, J.J. ve Sinanoglu, O.** (2020). *Trustworthy hardware design: Combinational logic locking techniques*, Springer.

[9] **Bhandari, J.**, **Moosa, A.K.T.**, **Tan, B.**, **Pilato, C.**, **Gore, G.**, **Tang, X.**, **Temple, S.**, **Gaillardon, P.E. ve Karri, R.** (2023). Not all fabrics are created equal: Exploring eFPGA parameters for IP redaction, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *31*(10), 1459–1471.

[10] **Torrance, R. ve James, D.** (2011). The state-of-the-art in semiconductor reverse engineering, *Proceedings of the 48th Design Automation Conference*, s.333–338.

[11] **Kahng, A.B.**, **Lach, J.**, **Mangione-Smith, W.H.**, **Mantik, S.**, **Markov, I.L.**, **Potkonjak, M.**, **Tucker, P.**, **Wang, H. ve Wolfe, G.** (1998). Watermarking techniques for intellectual property protection, *Proceedings of the 35th annual Design Automation Conference*, s.776–781.

[12] **Caldwell, A.E.**, **Choi, H.J.**, **Kahng, A.B.**, **Mantik, S.**, **Potkonjak, M.**, **Qu, G. ve Wong, J.L.** (1999). Effective iterative techniques for fingerprinting design IP, *Proceedings of the 36th annual ACM/IEEE Design Automation Conference*, s.843–848.

[13] **Li, M.**, **Shamsi, K.**, **Meade, T.**, **Zhao, Z.**, **Yu, B.**, **Jin, Y. ve Pan, D.Z.** (2017). Provably secure camouflaging strategy for IC protection, *IEEE transactions on computer-aided design of integrated circuits and systems*, *38*(8), 1399–1412.

[14] **Rajendran, J.**, **Sinanoglu, O. ve Karri, R.** (2013). Is split manufacturing secure?, *2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, s.1259–1264.

[15] **Mal-Sarkar, S.**, **Krishna, A.**, **Ghosh, A. ve Bhunia, S.** (2014). Hardware trojan attacks in fpga devices: threat analysis and effective counter measures, *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI*, s.287–292.

[16] **Karn, R.R.**, **Knechtel, J. ve Sinanoglu, O.** (2025). Logic Locking for Random Forests: Securing HDL Design and FPGA Accelerator Implementation, *International Conference on Information Systems Security and Privacy*, cilt 2, Science and Technology Publications, Lda, s.463–473.

[17] **U.S. Department of Defense** (2022). *DoD Microelectronics Third-Party IP Review Process for LOA 1*, `https://media.defense.gov/2022/Dec/08/2003127959/-1/-1/0/CTR_DOD_MICROELECTRONICS-THIRD_PARTY_IP_REVIEW_PROCESS_FOR_LOA1.PDF`, date retrieved: 28.05.2025.

[18] **U.S. Department of Defense** (2023). *DoD Microelectronics Third-Party IP Review Process for LOA 2*, `https://media.defense.gov/2023/Feb/27/2003168052/-1/-1/0/CTR_DOD_MICROELECTRONICS_THIRD_PARTY_IP_REVIEW_PROCESS_FOR_LOA2.PDF`, date retrieved: 28.05.2025.

[19] **U.S. Department of Defense** (2023). *DoD Microelectronics Third-Party IP Review Process for LOA 3*, `https://media.defense.gov/2023/Jun/29/2003250482/-1/-1/0/CTR_DOD_MICROELECTRONICS-THIRD_PARTY_IP_REVIEW_PROCESS_FOR_LOA3.PDF`, date retrieved: 28.05.2025.

[20] **Roy, J.A.**, **Koushanfar, F. ve Markov, I.L.** (2008). EPIC: Ending piracy of integrated circuits, *Proceedings of the conference on Design, automation and test in Europe*, s.1069–1074.

[21] **Rajendran, J.**, **Pino, Y.**, **Sinanoglu, O. ve Karri, R.** (2012). Security analysis of logic obfuscation, *Proceedings of the 49th annual design automation conference*, s.83–89.

[22] **Subramanyan, P.**, **Ray, S. ve Malik, S.** (2015). Evaluating the security of logic encryption algorithms, *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, IEEE, s.137–143.

[23] **Rajendran, J.**, **Zhang, H.**, **Zhang, C.**, **Rose, G.S.**, **Pino, Y.**, **Sinanoglu, O. ve Karri, R.** (2013). Fault analysis-based logic encryption, *IEEE Transactions on computers*, *64*(2), 410–424.

[24] **Yasin, M.**, **Rajendran, J.J.**, **Sinanoglu, O. ve Karri, R.** (2015). On improving the security of logic locking, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *35*(9), 1411–1424.

[25] **Yasin, M.**, **Mazumdar, B.**, **Rajendran, J.J. ve Sinanoglu, O.** (2016). SARLock: SAT attack resistant logic locking, *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, IEEE, s.236–241.

[26] **Xie, Y. ve Srivastava, A.** (2016). Mitigating SAT attack on logic locking, *Cryptographic Hardware and Embedded Systems–CHES 2016: 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings 18*, Springer, s.127–146.

[27] **Yasin, M.**, **Sengupta, A.**, **Nabeel, M.T.**, **Ashraf, M.**, **Rajendran, J. ve Sinanoglu, O.** (2017). Provably-secure logic locking: From theory to practice, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, s.1601–1618.

[28] **Rathor, V.S.**, **Singh, M.**, **Sahoo, K.S. ve Mohanty, S.P.** (2023). GateLock: Input-Dependent Key-based locked Gates for SAT resistant logic locking, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *32*(2), 361–371.

[29] **Yoshimura, M.**, **Tsujikawa, A. ve Hosokawa, T.** (2024). CRLock: A SAT and FALL Attacks Resistant Logic Locking Method for Controller at Register Transfer Level, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, *107*(3), 583–591.

[30] **National Institute of Standards and Technology** (2024). *NIST SP 800-224 (Initial Public Draft)*, `https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-224.ipd.pdf`, date retrieved: 28.05.2025.

[31] **National Institute of Standards and Technology** (2015). *FIPS PUB 180-4: Secure Hash Standard*, `https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf`, date retrieved: 28.05.2025.

[32] **National Institute of Standards and Technology** (2023). *FIPS PUB 186-5: Digital Signature Standard (DSS)*, https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf, date retrieved: 28.05.2025.

[33] **Rivest, D.R.**, **Shamir, A. ve Adleman, L.** (1978). RSA (cryptosystem), *Arithmetic Algorithms And Applications*, 19.

[34] **Moriarty, K.**, **Kaliski, B.**, **Jonsson, J. ve Rusch, A.** (2016). *PKCS #1: RSA Cryptography Specifications Version 2.2*, RFC 8017, https://www.rfc-editor.org/info/rfc8017.

[35] **Politecnico di Torino CAD Group** (n.d.). *I99T: ITC'99 benchmarks*, https://github.com/cad-polito-it/I99T, https://github.com/cad-polito-it/I99T, gitHub repository.

[36] **Corno, F.**, **Sonza Reorda, M. ve Squillero, G.** (2000). RT-level ITC'99 benchmarks and first ATPG results, *IEEE Design Test of Computers*, *17*(3), 44–53.

[37] **Er, J. ve contributors** (n.d.). *NEOS: Optimization Toolbox*, https://github.com/jinyier/NEOS, https://github.com/jinyier/NEOS, gitHub repository.

[38] **UC Berkeley ABC Group** (n.d.). *ABC: A System for Sequential Logic Synthesis and Verification*, https://github.com/berkeley-abc/abc, https://github.com/berkeley-abc/abc, gitHub repository.

ÖZGEÇMİŞ

Adı SOYADI: Teyyar AÇAR

ÖĞRENİM DURUMU:

- **Lisans:** 1998, İstanbul Teknik Üniversitesi, Elektrik - Elektronik Fakültesi, Elektronik ve Haberleşme Mühendisliği

MESLEKİ DENEYİMLER VE ÖDÜLLER:

- 2015 yılından bu yana TÜBİTAK BİLGEM'de donanım ve yazılım güvenliği üzerine çalışıyor.

DİĞER YAYINLAR, SUNUMLAR VE PATENTLER:

- **Açar, T.**, Tiryakioğlu, F., Örs Yalçın, B. "Üçüncü Şahıs IP Koruması için Mantıksal Kilitleme Tabanlı FPGA Sistem Tasarımı", *33. IEEE Sinyal İşleme ve İletişim Uygulamaları Kurultayı* 25-28 Haziran, 2025 İstanbul, Türkiye.