Preventing Piracy and Reverse Engineering of SRAM FPGAs Bitstream

Lilian Bossuet¹, Guy Gogniat¹, Wayne Burleson²

¹ LESTER Lab
Université de Bretagne Sud
Lorient, France
lilian.bossuet@univ-ubs.fr

² Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, USA
burleson@ecs.umass.edu

Outline

- Design Security
- Actual Solutions
- Propositions
- Conclusion
Approche de conception d’interface de communication pour les systèmes sur puce

FPGA and Security ...

- System Security using FPGA
- Protecting FPGA data and resources
- FPGA Design Security

Use the FPGA to protect
- Network isolation (firewalls)
- Smart cards
- Sensor networks

Protect the FPGA
- Data encryption scheme
- Protection against hardware damage

Protect the FPGA configuration
- Bitstream encryption
- Watermarking

- Network isolation (firewalls)
- Smart cards
- Sensor networks

Design Security

http://lester.univ-ubs.fr/
Need for Design Security

Protection against

- **Cloning**
  - A competitor makes copy of the boot ROM or intercepts the bitstream and copies the code.

- **Reverse Engineering**
  - A competitor copies a design by reconstructing a "schematic" or netlist level representation; in the process, he understands how the design works and how to improve it, or modify it with malicious intent.

Attack Types

- **Noninvasive**
  - Monitored by external means such as brute force key generation, changing voltages to discover hidden test modes, etc.

- **Invasive**
  - Decapped and then microprobed using focused ion beams, or other sophisticated techniques to determine the contents of the device.

Levels of Semiconductor Security

- **#3** Can be broken into in a government sponsored lab (e.g. USA NSA or France DGA)

- **#2** Can theoretically be broken into with time and expensive equipment

- **#1** Not secure, easily compromised with low costs tools

Source: IBM systems journal Vol. 30 No 2 - 1991
**Actual Semiconductor Level of Security**

*Source: ACTEL*

<table>
<thead>
<tr>
<th>DEVICES</th>
<th>SECURITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM FPGA</td>
<td>LEVEL 1</td>
</tr>
<tr>
<td>ASIC Gate Array</td>
<td>LEVEL 2 -</td>
</tr>
<tr>
<td>Cell-Based ASIC</td>
<td>LEVEL 2 -</td>
</tr>
<tr>
<td>SRAM FPGA with Bitstream encryption</td>
<td>LEVEL 2</td>
</tr>
<tr>
<td>Flash FPGA</td>
<td>LEVEL 2 +</td>
</tr>
<tr>
<td>Antifuse FPGA</td>
<td>LEVEL 2 +</td>
</tr>
</tbody>
</table>

---

**Actual Solutions**
Approche de conception d’interface de communication pour les systèmes sur puce

Flash and Antifuses FPGA

- Flash and Antifuse FPGA = 10%
- SRAM FPGA = 90%
- BUT not really “reconfigurable” just configurable...

Xilinx Solution

- Secret keys (Triple DES - 3 x 56 bits)
- Need of an external battery to save the keys
- The decryption circuit takes FPGA resources (silicon)
- No flexibility for the decryption algorithm
- Partial reconfiguration is no more available

Protection against cloning and reverse engineering
Algotronix (U.K.) Proposition

- A secret cryptographic key can be stored on each FPGA
- No need for the CAD tools or any person to have knowledge of the key
- The Encryption/Decryption circuit takes FPGA resources (silicon)...
- No flexibility for the decryption algorithm

```
http://lester.univ-ubs.fr/
```


```
http://lester.univ-ubs.fr/
```
Propositions...

Desirable Characteristics

- Strong protection against cloning and reverse engineering for SRAM FPGA
- No additional battery => the key is embedded (laser)
- Choice of the suitable encryption algorithm and architecture (security policy)
- No use of FPGA application-dedicated resources
Approche de conception d’interface de communication pour les systèmes sur puce

Security Policy

Actual solutions: the whole design (all IPs) has the same security policy

Proposition: Each IP can be encrypted with its own algorithm

Security Policy

- Application partitioning with necessary security levels
- All of the application doesn’t need to be encrypted (free available IP)

Security-Critical Part: your Intellectual property which needs higher security due to:
  - development cost
  - potential security breach

No critical Parts: Generic functions and cores such as communication protocols, etc.
Approche de conception d’interface de communication pour les systèmes sur puce

Encryption - Decryption management

- Once the FPGA is configured encryption and decryption circuits do not use FPGA resources
- Use of partial dynamic reconfiguration
- Use of self-reconfiguration (to configure each IP with the corresponding decryption circuit)
- Embedded secret key

Encryption Management (in the lab.)

IP1 (SCP1) algorithm 1

IP2 (SCP2) algorithm 2

IP3 (NCP)

FPGA : secret KEY

Encryption circuit1

Decryption circuit1

EPROM

SCP1 Encrypted

http://lester.univ-ubs.fr/
Approche de conception d’interface de communication pour les systèmes sur puce

Encryption Management (in the lab.)

IP1 (SCP1) algorithm

IP3 (NCP)

IP2 (SCP2) algorithm 2

Encryption circuit2

FPGA : secret KEY

EPROM

SCP1 Encrypted

Decryption circuit1

SCP2 Encrypted

Decryption circuit2

FPGA : secret KEY

Encryption Management (in the lab.)

IP1 (SCP1) algorithm 1

IP3 (NCP)

IP2 (SCP2) algorithm 2

Decryption circuit1

SCP2 Encrypted

Decryption circuit2

NCP

http://vsp2.ecs.umass.edu/vspg/vspg.html
Approche de conception d’interface de communication pour les systèmes sur puce
Approche de conception d’interface de communication pour les systèmes sur puce

Decryption Management (at power up)

- Partial and self reconfiguration
- The decryption circuit1 is replaced by the decryption circuit2

http://lester.univ-ubs.fr/
Approche de conception d’interface de communication pour les systèmes sur puce

Decryption Management (at power up)

- Partial and self reconfiguration
- The decryption circuit1 is replaced by the decryption circuit2
- The decryption circuit2 is replaced by the non critical part that is not encrypted

Main issues

- Partial and self-reconfiguration
- Area related to the decryption algorithms
- Keys management for each encryption/decryption circuit
- Configuration controller to perform the configuration management
Approche de conception d’interface de communication pour les systèmes sur puce

Area related to the decryption algorithms

- Once the last encrypted bitstream is configured the last decryption algorithm can be replace by non critical parts

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>#slices of the cryptographic core</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rijndael</td>
<td>4312 5302 2902</td>
</tr>
<tr>
<td>Serpent</td>
<td>1250 7964 4438</td>
</tr>
<tr>
<td>RC6</td>
<td>1749 3189 1139</td>
</tr>
<tr>
<td>Twofish</td>
<td>2809 3053 1076</td>
</tr>
</tbody>
</table>

~ 100 Mbits/s < Throughput < ~400 Mbiys/s

- ICAP performance: 66 Mbits/s, throughput is not the major concern but area

Keys management

- Secret key is define during the fabrication process (laser)
- Define a large key e.g. 1024 bits that is used to generate each secret key (56 bits, 128 bits)
- Hardwired key generator
- Hardwired mechanism to authenticate each decryption circuit and encrypted bitstream (signature)
Approche de conception d’interface de communication pour les systèmes sur puce

Configuration Controller

EPROM
SCP1 Encrypted
Decryption circuit1
SCP2 Encrypted
Decryption circuit2
NCP

Configuration Controller

IP1
IP3
FPGA : secret KEY

Advantages / Inconveniences

+ The encryption/decryption circuit doesn’t take FPGA resources
+ Choice of a suitable encryption algorithm and architecture (to obtain a required security level)
+ Only designer knows the chosen algorithm and architecture
+ The system can be upgraded with new encryption algorithms
+ No additional battery with an embedded secret key

- Complex system, keys management
- Management of partial, self and dynamic reconfiguration
- Take time during the system set up

L.E.S.T.E.R
Laboratoire d'Etudes en Systèmes Temporisés

http://lester.univ-ubs.fr/
Conclusion

- In consequence of the rise of the FPGA, it is necessary to prevent piracy and reverse engineering of FPGA Bistream
- Existing SRAM FPGA are insecure
- We propose original solutions that take advantage of the partial and dynamic FPGA configuration, with a no-fixed encryption algorithm and architecture
- It is a first step toward security policy for FPGA and the solutions still to be improved...
Approche de conception d’interface de communication pour les systèmes sur puce

Comments?
Questions?
Thank you

http://vsp2.ecs.umass.edu/vspg/vspg.html
http://lester.univ-ubs.fr/
Implementing Secure, Upgradeable FPGA Designs

Dr. Yankin Tanurhan
Senior Director of IP and Applications
Actel Corporation

---

Agenda

- Overview
- Need for Security
- Methods of Attack, Theft, and Device Security
- Example
- Summary
Market Situation

- Traditionally, ASICs held the majority of System IP …
  - High integration
  - Relatively inexpensive in high volume
  - Thought to be relatively secure
- … and FPGAs were primarily used as glue logic
  - Relatively small devices with little IP content
  - Interface between ASSPs or custom ASICs
- Today, FPGAs becoming attractive ASIC alternatives and, therefore, the repository of most, if not all, system IP
  - High integration of system-level functions
  - Shorter lead times
  - No expensive NREs
  - Increased density and ability to handle ever faster clock speeds
  - Flexibility and variety of upgrade methods

Increasing System Risk

- Many systems now have most, if not all of the sensitive IP contained in an FPGA
  - A typical system might incorporate a processor or DSP, memory, ASSPs and one or more FPGAs
- If you can read the contents of the FPGA, you can duplicate (or enhance) the function of the entire system since all other components are off-the-shelf
- The vulnerability of FPGAs to copying puts the proprietary IP of the system at risk
FPGA: System IP Repository


FPGA = Glue Logic  FPGA = ASIC Replacement

- Today, the FPGA is the heart of the system

System IP: Protection Needed

- With most system IP in FPGAs, protection is paramount
  - Development time of valuable IP is costly
    - Months to years of effort
  - Trade secrets need to be protected
    - Competitors love any hints you can drop to them
  - Unauthorized copying hurts the bottom line
    - No replacement for lost potential revenue
Programming Flexibility

- FPGA flexibility comes at a cost
- Ease of programming methods inversely proportional to security

Easy to Program ⇔ Less Secure

- Time spent on enabling security features up front saves time and money in the long run
- Novel update approaches need to be scrutinized for security risks
  - http vs. https
  - Programming at trusted site vs. programming by non-qualified third party

Basic Security Principles

- Virtually any protection method can be cracked given enough time and resources
- At some point, the IP value within a circuit is less than the effort that must be expended to copy it
- Two methods of attack are primarily used:
  - Noninvasive: Monitored by external means, such as:
    - brute force key generation
    - changing voltages to discover hidden test modes
  - Invasive: To determine contents of device, it is decapped, then microprobed using sophisticated techniques
Noninvasive Attacks

- Attacker copies an external NVM containing the bitstream or attacks the device to get the code
- Simple security schemes are easily bypassed using noninvasive techniques
  - On some devices, it is possible to erase a secured device prior to reprogramming
  - Starting the erase and then immediately powering down the device can erase the security bit and not the program contents

Invasive Attacks

- Invasive attacks require decapsulation of the device
- Several invasive techniques can be used to break on-chip security schemes
  - Microprobing
  - Focused ion beams
  - Localized laser removal of metalization and deposition of conducting traces
- Some FPGAs are difficult to invasively attack
  - Distributed nature of security keys and physical characteristics of the silicon
  - Lack of obvious visible difference between programmed and non-programmed elements
Methods of Theft

- **Overbuilding**: Unscrupulous contract manufacturer buys standard parts on the open market and overbuilds, selling extra production for profit
- **Cloning**: Competitor makes a copy of the boot prom or intercepts the bitstream from the on-board processor and copies the code
- **Reverse Engineering**: Competitor copies a design by reconstructing a “schematic” or netlist level representation; as a result, he understands how the design works and how to improve it

Overbuilding

- Most common IP theft
- Also called “run-on counterfeiting”
- Built with inventory of standard parts bought on the open market
- Unauthorized overbuilt inventory sold for profit
  - Identical to the genuine product
  - Has no support or design overhead cost
  - Can drive the design company out of business

- Increasing reliance on standard products in embedded systems design makes it easier for unscrupulous manufacturers to overbuild
How to Prevent Overbuilding

- Best protection is to control “critical component” supply
  - Any part that has to be programmed or uniquely made by the designer
- Deliver one or more FPGAs already programmed
- Utilize available encryption keys on FPGAs
  - Deliver encrypted bit stream to manufacturer
  - Only parts with programmed encryption key can be used with the encrypted bit stream
  - Reduces programming time required by the designing company

Cloning

- Less prevalent than overbuilding, but serious problem
- Competitor makes a copy of a design -- stealing part or all of the system IP
- Accomplished by copying the bit stream from an unprotected FPGA
- Unless strong security schemes are used, even devices with ‘security’ bits can be copied
- Companies specialize in copying programmable and similar logic devices
How to Prevent Cloning

- Best protection against cloning is to use a secure FPGA and to enable all of the security features

Reverse Engineering

- More difficult than cloning, but more pervasive
- Accepted practice used by many companies to evaluate the competition
- Explicit details on how the design works determined by testing and analysis
- Reverse engineering techniques can give a competitor all of the IP used in a design
  - Allows them to recreate it and improve on it
  - Can then disguise it to thwart possible legal action
How to Prevent Reverse Engineering

- Equal time spent to protect against reverse engineering as is spent protecting against cloning
- Most common form is simple I/O attacks
  - Cycle through large number of inputs to determine the output response
- Difficult when processing elements are used in the design
  - Add variability with the processing elements in the design
- Split up circuit between multiple parts and use token scheme when passing data
  - Tokens are not easily deduced
  - Without the correct tokens the circuit will not work
- Reverse engineering protection won’t help if FPGA isn’t secure

Patent Protection?

- Costly and time consuming to obtain
  - To specify an enforceable patent requires both engineering and legal resources
  - Patent application process is slow
  - Limits effectiveness in dynamic marketplace
- Enforcement is costly and uncertain
  - Litigation can run into millions of dollars
  - If an invention’s value is less than the cost of enforcing the patent protection, it’s essentially unprotected
  - Legal system is notoriously unpredictable
  - Lack of international standards
    - Differing rules, requirements and enforcement policies make international patent protection uncertain
Levels of Device IP Security*

- **Level 1:** Devices are insecure
  - Easily copied by technically knowledgeable individuals with low cost, easily accessible tools

- **Level 2:** Devices are moderately secure
  - Can theoretically be copied by skilled individual or team with access to sophisticated and expensive equipment

- **Level 3:** Devices are highly secure
  - Copied only with extraordinary difficulty by a government sponsored lab (e.g. NSA) with “unlimited” resources

* Source: IBM

---

Methods of Device Security

- The methods available to protect the device from security attacks differ by type of device
  - FPGA – SRAM, Antifuse and Flash
  - ASIC
- Essentially, two categories of FPGA security:
  - **Internal:** internal structure of the FPGA lends itself to the inherent security of the design after the user has programmed the device
  - **External:** a bit stream is encrypted to allow for transport across a non-secure medium and is decrypted (3DES, AES, etc.) by dedicated hardware within each FPGA device

* Source: IBM
Popular Encryption Standards

- DES – Data Encryption Standard
  - Algorithm developed in early 1970’s by IBM
  - Adopted as Data Encryption Standard by National Institute of Standards and Technology (NIST) in 1976
  - 64 bits of data encrypted/decrypted using 56-bit key
  - Symmetric algorithm – using same private key, encrypted data can be decrypted
  - Electronic Frontier Foundation (EFF) has cracked DES using:
    - EFF hardware, 100,000 computers, and 22 hours to kill
      (http://www.eff.org for details)
- 3DES – Triple DES
  - As name implies, uses three iterations of DES algorithm
  - In 1999, NIST specified preferred use of Triple DES instead of DES
  - 64 bits of data encrypted/decrypted using 168-bit key (3 56-bit keys)
  - Symmetric algorithm

Encryption Standards (cont.)

- AES – Advanced Encryption Standard
  - Based on Rijndael algorithm
  - Chosen by NIST to replace DES/3DES
  - 128 bits of data encrypted/decrypted using 128/192/256-bit key
  - Also a symmetric algorithm
  - Mathematical analysis junkies figure that, if you can build a DES-cracker capable of finding a DES key in 1 second, it would take that same hardware **149 trillion years to crack a 128-bit AES key**

- Key combinations
  - DES: \(2^{56} \approx 7.2 \times 10^{16}\)
  - 3DES: \(2^{168} \approx 3.7 \times 10^{56}\)
  - AES: \(2^{256} \approx 1.2 \times 10^{77}\)
SRAM FPGA Security - Level 1

- SRAM devices are volatile
- Configuration data must be loaded each time power is cycled
  - Data stored in a PROM or microprocessor and fed to FPGA at power up
  - Bit stream data easily copied at boot-up
- Newer SRAM FPGAs support encrypted bitstreams (3DES/AES) to increase security

ASIC Security - Level 2

- Often considered “secure,” ASICs are relatively easy to reverse engineer
- Packages are decapped; layers stripped and photographed one by one; composite overlay of layers yields transistor layout and all interconnects
- Even complex designs can be easily decoded in a relatively short time
Flash FPGA Security - Level 2+

- Nonvolatile devices do not require external bit stream
  - No bit stream download during power-up
  - Programming information can be read out of the device, so locks are required to protect IP
- Highly resistant to invasive attacks
  - Decapping and stripping only reveals structure of the device, not actual contents of cell
  - Large number of switch elements; attempting to determine the state of millions of switches is prohibitive

Flash FPGA Security - Level 2+ (cont.)

- Multiple Locking Features Available
  - User Lock Feature – “Lock and keep the key”
    - Security is based on 128-bit AES encryption with a User Supplied Key
    - Key is required to program the parts
    - Information read out of the devices is encrypted with the key
  - Permanent Lock Feature – “Lock and throw away the key”
    - Permanently locks parts
    - Further programming or reading of the devices is not possible
Antifuse FPGA Security - Level 2+

- All data internal to device
- Nonvolatile
- No bit stream download during power-up
- Programming info can’t be read out of device
- Immune to invasive attacks
  - State of antifuse only detected in cross-section
  - Million of antifuses and < 2% are programmed - “needle in haystack” problem
  - Difficult to distinguish programmed from unprogrammed antifuse
- FuseLock
  - Locking antifuses prevent non-invasive attack
  - Device remains locked even with no power

Technology Summary

<table>
<thead>
<tr>
<th>Technology</th>
<th>Security Level</th>
<th>Security Overhead</th>
<th>Security Shelf-life</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional SRAM FPGA</td>
<td>1</td>
<td>No Security</td>
<td>No Security</td>
</tr>
<tr>
<td>Hybrid NVM/ SRAM FPGA</td>
<td>1</td>
<td>None</td>
<td>10 Yrs+</td>
</tr>
<tr>
<td>Gate Array</td>
<td>2-</td>
<td>None</td>
<td>Indefinite</td>
</tr>
<tr>
<td>Cell-Based ASIC</td>
<td>2-</td>
<td>None</td>
<td>Indefinite</td>
</tr>
<tr>
<td>SRAM w/ 3DES encryption</td>
<td>2</td>
<td>Significant</td>
<td>&lt;5 Yrs*</td>
</tr>
<tr>
<td>Flash-based FPGAs</td>
<td>2+</td>
<td>None</td>
<td>20 Yrs+</td>
</tr>
<tr>
<td>Antifuse-based FPGAs</td>
<td>2+</td>
<td>None</td>
<td>Indefinite</td>
</tr>
</tbody>
</table>

*Battery backup required, typical service life < 5 years
Application: Subscription IP

- FPGAs can now be used for creation of efficient subscription systems
- FPGAs with internal nonvolatile memory can be used to store:
  - Unique Board/Device Serial Number
  - Secure Keys
  - Set-top Box Configuration
- Storing valuable information absolutely requires that contents are secure

Secure IP Delivery Example

- IP provider can send evaluation board to customer
- Provider’s customer can program any number of IP blocks in a secure manner
Secure IP Delivery Example (cont.)

- IP provider security software:
  - Licensed using serial number from evaluation board
  - Software encryption algorithm uses internal secret key (known only to IP provider) together with serial number to generate encrypted IP keys written to Flash memory

![Diagram of IP delivery process]

Secure IP Delivery - Step 1

- Step 1:
  - Provider gives each board a unique serial number
  - Serial number encrypted with secret key known only to IP provider
  - Provider write-locks page to prevent overwriting serial number
  - Read-locking must not be done on area of Flash memory where encrypted serial number is stored (part of authentication mechanism)
  - Keys are encrypted, stored in Flash memory, and read and write-locked
Secure IP Delivery – Step 2

- Step 2:
  - Provider’s customer buys eval board and chooses IP cores to purchase
  - Customer downloads encrypted programming files for each IP core from secure website
  - Customer authorized to unlock specific cores purchased from provider
  - The IP core programming files are encrypted using security keys known only to provider

Secure IP Delivery – Step 3

- Step 3:
  - Provider gives the customer the passcodes for purchased IP cores
Secure IP Delivery - Step 4

- Step 4:
  - Customer enters passcodes into provider’s supplied software
  - The software reads the eval board’s serial number via JTAG to perform hardware authentication
  - Each authorized page of Flash memory is unlocked to provide access to encrypted IP security keys

Secure IP Delivery – Step 5

- Step 5:
  - Provider’s programming software is used to decrypt an IP security key, using eval board’s serial number, then program the encrypted IP core programming file previously downloaded from the Web (step 2)
Subscription Use Model

OEM programs AES Key and FROM with unique TAG

Part is deployed in ‘box’

‘Customer’ requests additional feature by supplying TAG value of ‘box’ over Internet

New feature is enabled and will only work with the correctly encrypted design supplied

OEM looks up AES key based on TAG supplied

New feature-enabled design is encrypted with AES Key and sent to customer via Internet, satellite, or direct to ‘box’ for secure ISP

New feature is enabled and will only work with the correctly encrypted design supplied

Recap: Steps to Protect Intellectual Property

- Use secure FPGA technology to minimize attacks at the physical level.
- Guard against simple I/O scan attacks.
- Employ procedures to implement and track IP and programming changes to limit design exposure in manufacturing channel.
- Add digital “watermarks”/“fingerprints” to design to later help prove that a competitive design is a copy.
- If outsourcing production, ensure that additional units are not produced without your knowledge.
- Use trusted silicon vendors for design implementation.
Summary

- FPGAs are displacing ASICs with increasing frequency as repository of system IP
- Design piracy is on the rise
- Security needed to protect your investment in IP
- Various methods of attack, theft and device security can be mitigated by enabling the security features provided by some FPGAs
- Flash- and antifuse-based FPGAs typically more secure than SRAM FPGAs and ASICs
- Applications, such as subscription systems, can benefit from today’s FPGA security