Prüfung von Manipulationsmöglichkeiten von Hardware in verteilten Fertigungsprozessen (PANDA)

Analysis of Hardware Manipulations in Distributed Manufacturing Processes
## Document history

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Editor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content 1.0</td>
<td>28.11.23</td>
<td></td>
<td>Final version</td>
</tr>
</tbody>
</table>

*Table 1: List of document versions*
Acknowledgement

This study was authored by IHP – Leibniz-Institut für innovative Mikroelektronik on behalf of the German Federal Office for Information Security (BSI).
Table of Contents

1 Introduction ................................................................................................................................................. 6

2 WP2: Design and Manufacturing steps from the initial idea to the final product ........................................ 8
   2.1 Selecting IP blocks ............................................................................................................................... 9
       2.1.1 Selection criteria .......................................................................................................................... 10
       2.1.2 Actors and responsibilities ........................................................................................................... 10
       2.1.3 Tools ........................................................................................................................................... 11
   2.2 ASIC Design ....................................................................................................................................... 11
       2.2.1 Actors and responsibilities ........................................................................................................... 12
       2.2.2 Tools ........................................................................................................................................... 12
   2.3 Mask Production ................................................................................................................................. 13
       2.3.1 Actors and responsibilities ........................................................................................................... 14
       2.3.2 Tools ........................................................................................................................................... 15
   2.4 Manufacturing .................................................................................................................................... 15
       2.4.1 Actors and responsibilities ........................................................................................................... 16
       2.4.2 Tools ........................................................................................................................................... 16
       2.4.3 Available Information .................................................................................................................. 16
   2.5 Test ..................................................................................................................................................... 16
       2.5.1 Actors and responsibilities ........................................................................................................... 16
       2.5.2 Tools ........................................................................................................................................... 17
       2.5.3 Available Information .................................................................................................................. 17
   2.6 PCB Design ........................................................................................................................................ 17
       2.6.1 Actors and responsibilities ........................................................................................................... 17
       2.6.2 Tools ........................................................................................................................................... 17
       2.6.3 Available Information .................................................................................................................. 18
   2.7 Production and assembly of PCBs ...................................................................................................... 18
       2.7.1 Actors and responsibilities ........................................................................................................... 18
       2.7.2 Tools ........................................................................................................................................... 19
       2.7.3 Available Information .................................................................................................................. 19
   2.8 Software/firmware development or use of existing ones .................................................................... 19
       2.8.1 Actors and responsibilities ........................................................................................................... 19
       2.8.2 Tools ........................................................................................................................................... 19
       2.8.3 Available Information .................................................................................................................. 20
   2.9 Summary .......................................................................................................................................... 20

3 WP3: Identification of potential vulnerabilities and attack scenarios including a detailed elaboration of
       selected attack scenarios ....................................................................................................................... 22
   3.1 Introduction ...................................................................................................................................... 22
Appendix 1: Examples of selecting IP Cores (IHP experience)
Appendix 2: Examples of pins and pads
Appendix 3: Overview of MITRE hardware attack scenarios
Appendix 4: Interview with IHP designers
Appendix 5: Interview with IHP PCB experts
Appendix 6: Attack scenario evaluations
Appendix 7: Implemented Trojan code
Appendix 8: PCB experiment Additional details
1 Introduction

Today, the development and implementation of complex IT systems is very often no longer carried out by a single manufacturer, who performs and controls all development and production steps completely himself. This division of labour has clear advantages such as shorter “time-to-market” and reduced costs. It also has the advantage that the necessary competences and tools/machines are focused on core areas. However, the clear disadvantage is that there are risks associated with the division of labour in terms of product quality (reliability/safety). This also holds true for all aspects of security, leading to the loss of confidentiality, privacy etc. Changes in the product can be made at almost all stages of development and production. The aim of this study is to analyse the potential risk in all the development steps from the initial design down to the final product.

In 2018 Bloomberg published an article that discussed potential manipulations of Supermicro motherboards that were “extended” by an additional Integrated Circuit (IC) during manufacturing [1]. Supermicro customers are amongst others companies like Amazon and Apple who use the mainboards e.g. in their cloud servers. According to Bloomberg the alleged manipulations were discovered during routine reviews during an acquisition of Elemental by Amazon (AWS) as well as by Apple in their data centres. The involved companies denied the reports. Bloomberg on the other hand insisted on the factual correctness of their reporting. They state that “investigators determined that the chips allowed the attackers to create a stealth doorway into any network that included the altered machines” and cite several independent anonymous sources from within the allegedly involved companies as well as from the U.S. government [1]. In addition, Bloomberg published another article in 2021, where a manipulation of software i.e. of the Basic Input/Output System (BIOS) of Supermicro motherboards was mentioned [2].

Whether the reported attacks really happened or not is still an open discussion. Supermicro claims that such attacks did not happen and that they were not informed by any governmental agency. On the other hand, Bloomberg insists on the reports.

Even though it is not clear whether this attack really happened as described by Bloomberg, their reports clearly indicates that Information technology (IT) products compiled by many companies scattered all over the world are to a certain extent vulnerable to malicious manipulations run by one of these different suppliers.

Since there is little to no information available about the functionality of the aforementioned additional ICs and the changes in the BIOS, there is also no information about the goals of the attack and the attacker. In principle an attacker can pursue the following goals with hardware-related attacks:

- Damage to the reputation of a supplier by deliberately manufacturing low-quality Application-Specific Integrated Circuits (ASICs), this attack then targets a specific victim.
- Espionage to obtain know-how, company secrets and/or secret information such as cryptographic keys. This can be achieved, among other things, through the use of backdoors. Such attacks can target both the manufacturer and its customers, and can have a “broad” effect.
- “Kill switch”: Here, the attacker tries to manipulate the system in such a way that he can switch off its functionality. This attack also targets many rather than one specific victim, at least in preparation. If the “kill switch” is activated, this may be done very selectively.

We focus on these attack scenarios, since they are widely discussed in the literature1,2. Of course each scenario comes with certain risks for the attacker as well, e.g. damaging the reputation of a supplier by deliberately manufacturing low-quality ASICs3 must be carried out respectively concealed in such a way that the

---

1 Cf. e.g. [https://www.tandfonline.com/doi/full/10.1080/15228053.2020.1824878](https://www.tandfonline.com/doi/full/10.1080/15228053.2020.1824878)
2 Cf. e.g. [https://www.amida.com/enabling-hardware-trojan-detection-and-prevention-through-emulation/](https://www.amida.com/enabling-hardware-trojan-detection-and-prevention-through-emulation/)
3 Cf. e.g. [https://doi.org/10.1007/978-3-319-68511-3_3](https://doi.org/10.1007/978-3-319-68511-3_3)
reputation damage does not spread to the attacker, or that such reputation damage is willingly taken into account.

The type of attack and its complexity depend greatly on the target of the attacker. The possible points of attack in the value chain are discussed in this report. The analyses in this report are based on the assumption that the “system developer” i.e. the company that aims at implementing a certain functionality and that starts the development process has no malicious insiders. This is due to the fact that in the core of the design process all essential information are available. We are aware of the fact that also a team member of the “system development entity” might be malicious. But since the focus of this study is on the threats that arise from a global supply chain, we assume in this document that the “system development entity” is benign. A reflection of malicious insiders in the “system development entity” will be given in the conclusion to ensure that this threat is not overlooked.

The focus of the work packages is on development steps not executed by “system development entity”. So, mainly the “interfaces” between the manufacturing steps are considered potentially dangerous as information exchanged is the basis for a potential attacker to mount his attack. The external subcontractors are considered no trustworthy i.e. there may be malicious insiders or the whole company may be malicious.

This report is structured corresponding to the work packages (WP) of the project PANDA, i.e. each chapter of this report represents the results of one of the following work packages:

- WP2: Design and Manufacturing steps from the initial idea to the final product
- WP3: Identification of potential vulnerabilities and attack scenarios including a detailed elaboration of selected attack scenarios
- WP4: Prevention and detection options, risk assessment and recommendations

We present conclusions and recommendations in Chapter 5.

This document includes in addition 8 appendixes which present details e.g. on the experiments and expert interviews etc.
2 WP2: Design and Manufacturing steps from the initial idea to the final product

The assessment of the risks in the different development steps is done based on the assumptions concerning trustworthiness as discussed in Chapter 1. The following figure presents a bird’s eye view on the development steps of a complex IT system consisting of hard- and software. Green parts are considered benign while grey and black are potentially dangerous, reflecting the assumptions about trustworthiness.

![Figure 2.1: Schematic presentation of the development steps of a complex IT system from the initial idea (step 1) via selecting IP blocks and designing own hardware (step 2), manufacturing it (step 3), designing and manufacturing Printed Circuit Boards (PCBs) (step 4) to installing software (step 5) and testing the developed system (step 6); green refers to internal "system developer" process steps; grey and black refer to external subcontractors considered not trustworthy or to external process steps.](image-url)
It starts with the specification of the required functionality via designing, testing and realising the layout of the ASIC/System on Chip (SoC) under development. In Figure 2.1 we identified steps that are executed by the company that is developing the ASIC/SoC in green assuming that the design team is a part of the system developer team. For the green marked phases, we assume that the risk is rather low, i.e. we do not really focus on insider threats. The grey and black coloured blocks represent steps and components that are (or can be) executed by or bought from third parties.

The development steps will be discussed in detail in the rest of this chapter.

### 2.1 Selecting IP blocks

When creating complex designs, existing third-party designs, e.g. accelerators for mathematical or cryptographic operations, interfaces such as SPI or the like, may not be developed in-house because time and/or skills are lacking and purchasing of externally provided components is more efficient. These Intellectual Property (IP) cores may already have been modified by an attacker. This is particularly problematic because such changes are extremely difficult to detect in simulations, tests and measurements because there are no non-manipulated comparison possibilities. A detailed analysis of the IP cores is usually also extremely time-consuming or even completely impossible due to the time and skills required for this and the lack of information. The use of open designs can potentially provide a remedy here, as malicious manipulations may be detected by the “swarm intelligence” of the community.

In this section we discuss the selection of IP blocks. Some examples such as selecting processor cores i.e. RISC-V and communication cores such as Controller Area Network (CAN) bus controllers are given in Appendix 1. More information about functionality, selection criteria as well as integration of IP blocks into a design can be found in [3]-[10].

The following types of IP blocks are considered when selecting the IP block(s) for the system under development:

- **Soft cores** are implemented in a Hardware Description Language (HDL) with no or minimal optimisation for a specific target technology (vendor, family and device) to allow adaptation to the technology selected by the customer.

- **Firm cores** are implemented (generally) in an HDL and are optimised for a specific target technology (vendor, family and device) e.g. to ensure improved performance, power or area characteristics.

- **Hard cores**: Here the functionality is implemented in fixed-logic at the gate and signal route level. Thus, the design team has no influence on the functionality.

Besides the different types of IP blocks concerning their implementation, IP blocks can also be classified according to their functionality. There are four large categories of IP blocks, i.e. blocks providing processing (processor cores, Digital Signal Processor (DSP) cores etc.), interfaces e.g. Serial Peripheral Interface (SPI), Universal Asynchronous Receiver-Transmitter (UART), etc., communication such as Peripheral Component Interconnect (PCI), CAN, and specialty e.g. cryptographic functions like AES cores or True Random Number Generators.

In addition to the type of the IP cores their origin is an important aspect that is taken into account when selecting the IP cores:

- Field-Programmable Gate Arrays (FPGA) vendors
- IP libraries that are part of the FPGA tools
- Third party IP suppliers
- Open Access groups
- Universities, Research organisations
- Internally implemented
2.1.1 Selection criteria

The criteria for selecting an IP core cover a broad variety. It is difficult to define weights for the criteria in the general case. Design teams will need to define what their priorities are. The selection criteria range from compatibility via maturity over legal to cost issues and are detailed in the following bullet point list, in which also specific aspects for each of the coarse grained criteria are listed.

1. IP compatibility
   - Function compatibility
   - Input/Output (I/O) compatibility
   - Software compatibility
   - Cycle-to-cycle compatibility

2. IP verification/maturity/vendor
   - Available Information (completeness, technical support)
   - Means to evaluate functionality/ technical data etc.
   - Maturity of the design
   - Assessment of the vendor(s) (e.g. based on size of the company, time in business, reputation, customer service, etc.)

3. Legal issues
   - License
   - Portability and reusability
   - Type of access to the code/ adaptation of the functionality
   - Any type of restrictions

4. Costs
   - Total cost of ownership ≥ licensing fees + support and maintenance fees + unforeseen expenses (especially with immature IP)

In order to provide a deeper insight into the selection process we collected the experience made by colleagues when selecting IP cores. This experience covers commercial and open cores as well as cores from tool vendors. In addition, we provide the same information for an IP core IHP has commercialised. These details are given in Table A1.1 – Table A1.4 in Appendix 1.

2.1.2 Actors and responsibilities

The project and design teams\(^4\) are responsible to select an IP block and to thoroughly validate the IP core that is considered to be bought in order to reduce the time-to-market, design complexity etc. Important aspects are for example the interfaces which will be used/needed to integrate the IP cores with other parts of the design(s). Here it might be a plus if several IP cores are acquired from the same vendor, as they might come with better interoperability than IP core from different vendors. Depending on the kind of the selected IP blocks – soft or hard core – either the design team or the layout-design team will be responsible for the integration of the IP cores. If the back-end phase of the design flow will be delegated to subcontractors, the integration of IP blocks has to be considered as non-trustworthy i.e. then it is an external process step. Also evaluation of the functionality of the IP core is essential, as testability is. Here the design team needs to

\(^4\) As described at the beginning of this chapter the project/design team is a part of the system developer team, i.e. the tasks performed by the project/design team are internal "system developer" process steps (these steps are denoted in green in Figure 2.1 and Figure 2.2).
prepare an objective assessment of all the IP cores. Who takes the final decision depends on the structures of the company. The actors and their tasks in the design flow phases are shown in Table 2.1, columns “actors” and “details”, respectively. The tasks of design team are described more detailed in Subsection 2.2.2.

2.1.3 Tools

The tools that are used to integrate an IP block mainly depend on whether it is a soft core i.e. delivered in an HDL or e.g. as a hard core. This also determines in which design step the IP core is integrated. This may happen when describing the functionality or as a part of a physical design (see Figure 2.2 and Table 2.1, Section 2.2). Thus, the tools applied are the same as for the development of the rest of the whole design (see Subsection 2.2.2, Table 2.1, column “Tools, results”).

2.2 ASIC Design

In this section we discuss the process of the engineering of an ASIC, step-wise. Figure 2.2 represents the steps of the processes and their connections schematically. We give a short overview of the steps as well as the applied tools and actors in Table 2.1. More information about the steps can be found in [11]-[23].

The core of the design process i.e. the digital design flow is normally split into two phases:

• Front-end design: covers all steps from the specification to generating the Gate level netlist, normally done by the system developer

  More detailed, the front-end part of the ASIC design flow includes coding the data flow of each functional block as well as the interactions between the functional blocks in a hardware description language (VHDL or Verilog). Front-end design ignores logic delays and speed for the Register Transfer Level (RTL) coding and verification. The result is the functionally correct RTL code that can be synthesised into a netlist list for the target technology.

• Back-end design (or a physical design): is the process of converting the gate-level netlist into functional hardware. It covers the generation of the Graphic Database System (GDS) file in GDS-II format from the Gate level netlist. This can be done by the system developer as well as by a third party.

Due to the assumption that the "system development entity" is benign (see Chapter 1, p. 6), the availability of all information is not considered an issue during the front-end designing phase of the ASIC engineering. This phase of the designing is marked green in Figure 2.2 as well as in in Table 2.1.

There exist several companies that are focusing on the physical design i.e. these companies are offering the back-end design steps as a service to other companies. So, if these design steps are subcontracted they are no longer considered to be done by a trustworthy entity and are therefore marked grey in Figure 2.2 and in Table 2.1.

Depending on the kind of IP block applied, the integration can be done either in the front-end or in the back-end of the design flow, for example a soft IP-block described in VHDL can be integrated as a single functional block to the whole design for the synthesis. As the IP cores are provided by a third party they are considered extremely dangerous. This is the reason why they are coloured black in Figure 2.2.

The steps of the front-end and back-end design flow are listed and shortly described in Table 2.1. We applied the same colours for indicating internal and external parts/processes and by that also the level of risk as in Figure 2.2. Additionally, Table 2.1 gives an overview of the applied tools and responsibility of the actors.
2.2.1 Actors and responsibilities

In the design team different roles are defined: some designers are specialised on the implementation of the functionality (front-end design team). Here a further specialisation may be applied, for example writing of testbench files, i.e. there are employees developing the functional logic, while others are responsible for testing this functionality. The back-end design team is more specialised on the place and routing of the developed and synthesizable designs. This also covers aspects such as timing, clock trees, power consumption, chip area, etc. Not always the functionality described and tested in the front-end design flow phase is synthesizable into a technology dependent net list. Even if it is synthesizable, the place and route process can require to re-design some functional blocks. In such a case the both teams – the front-end and back-end design teams – have to do (many) iterations to fulfil all tests in the back-end design flow phase. Even though the back-end design team is part of the trustworthy "system developer team", unintentional mistakes can occur here due to a lack of in-depth specialised knowledge of the application: e.g. the functionality remains correct, but the placement of elements can cause successful Side-Channel Analysis (SCA) and/or Fault Injection (FI) attacks. However, such unintentional mistakes can occur also in the contribution of the front-end team i.e. by the entire "system developer team". The actors and their tasks in the design flow phases are shown in Table 2.1, column "actors" and "details", respectively.

2.2.2 Tools

There are a few tool vendors that are acting worldwide. These vendors such as Cadence, Synopsis and Mentor are providing tool suites that reach from editors for implementing behavioural models, via simulation of these models to generating layouts, see Table 2.1.

In order to adapt these models to the technology of the selected manufacturer in addition the Process Design Kit (PDK) of the manufacturer is required. These PDKs provide the information about the technology to be used in the manufacturing such as timing, power consumption, area and geometric shape of the gates.
### Table 2.1: The steps of the front-end and back-end design flow with applied tools and responsibility of the actors.

<table>
<thead>
<tr>
<th>Design flow steps</th>
<th>Details</th>
<th>Tools, results</th>
<th>Actors</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Functionality description:</strong> Implementa-</td>
<td>HDL (hardware definition language) code writing: Verilog or VHDL</td>
<td><strong>Applied Tools:</strong> Synopsis VCS</td>
<td>Design team</td>
</tr>
<tr>
<td><strong>Integration of soft IP-block(s)</strong></td>
<td>RTL (Register-Transfer Level) code: describes the desired hardware by logic gates, but the description is still technology-independent. RTL code is synthesizable with EDA (Electronic Design Automation) tools to produce gate level implementations.</td>
<td>Result: correctly working RTL model (.vcd), technology independent</td>
<td></td>
</tr>
<tr>
<td><strong>Front-end design flow part</strong></td>
<td>Conversion of the RTL description to a gate-level net list, i.e. the functionality is described using gates of the target technology and connections between them</td>
<td>Typical synthesis tools: - Cadence RTL Compiler/Build Gates/Physically Knowledgeable Synthesis (PKS) - Synopsys Design Compiler</td>
<td></td>
</tr>
<tr>
<td><strong>Required:</strong> target technology gate parameters (timing, power, area)</td>
<td><strong>Result:</strong> technology dependent net list</td>
<td><strong>Design team, usually specialised on layouts</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Floor and power planning</strong></td>
<td>Integration of hard IP-block(s)</td>
<td>Tool examples for the Automatic Place and Route: - Cadence (SOC Encounter, VoltageStorm, NanoRoute) - Synopsys (Design Compiler, IC Compiler)</td>
<td></td>
</tr>
<tr>
<td><strong>Placement and routing</strong></td>
<td>placing functional blocks in the chip area, allocating routing areas between them, planning critical power and ground connections, determining pad locations, constraints for internal core(s), special power pads, etc.</td>
<td><strong>Placement and routing:</strong> pre-placement, in-placement, post-placement before clock tree synthesis (i.e. with an ideal clock) and post-placement after clock tree synthesis. The gate level netlist is converted to a physical representation, i.e. in a layout.</td>
<td></td>
</tr>
<tr>
<td><strong>Physical and timing verification</strong></td>
<td>Different tests to verify/check the correctness of the layout: Design Rule Checking (DRC); Logical equivalence checks (Layout vs. Schematic, LVS); Electrical Rule Checking (ERC), etc.</td>
<td><strong>Preparation to tape out:</strong> final – the verified - layout file is streamed out in GDS-II format. It is de-facto standard for IC layouts. It is a binary file format representing planar geometric shapes hierarchically. The data is used for creating photo-masks. As well as should be given to a fab for the ASIC fabrication.</td>
<td></td>
</tr>
<tr>
<td><strong>Back-end design flow part</strong></td>
<td></td>
<td><strong>Result:</strong> the Graphic Database System file, i.e. the GDS-II (GDS2) file.</td>
<td></td>
</tr>
</tbody>
</table>

### 2.3 Mask Production

A photomask or reticle is a plate with holes or transparent films that determines which parts of a wafer are illuminated during the photolithography process. Thereby a specific pattern is formed using photosensitive polymers on a substrate. For each layer of an ASIC at least one mask is needed. Multiple masks used during manufacturing are called a set of masks. Figure 2.3 shows an example of a mask from the set of masks manufactured for an IHP ASIC.
The set of the masks of all ASIs on the wafer to be produced has to be manufactured at first. The first step when producing masks is to compose layout data provided for each single ASIC design into a common "wafer-layout". This step is known as “tapeout”. This is needed to ensure an optimal usage of the area of the wafer and is done for multi project wafers, i.e. when different designs are manufactured together to save money, but also when an engineering run is planned i.e. when a run with a single design shall be done. The input data are layout data in GDS-II or OASIS and sometimes CAD formats like DWG (from ‘drawing’) and DXF (drawing exchange format) or other.

2.3.1 Actors and responsibilities

Compiling the layout data of single ASIC designs into a common wafer-layout usually will be done by the chip manufacturer, as it is done by IHP. Sometimes it is offered as a service by the mask shops. As there are only a few masks shops working worldwide and even a few are German we provide a list here. There are photomask manufacturers in Germany:

- Heidelberg Instruments [24]
- Advanced Mask Technology Center GmbH [25]-[26]
- Compugraphics Jena GmbH [27]

Non-German photomask manufacturers are:

- Advanced Reproductions Corporation (USA)
- Compugraphics (UK)
- Dai Nippon Printing (DNP) Co., Ltd (Japan)
- Hoya Corporation (Japan)
- Nippon Filcon (Japan)
- Photronics, Inc. (USA)
- Taiwan Mask Corporation (Taiwan)
- Toppan Photomasks, Inc. (Japan)

The world leading ASIC manufacturers Intel, Globalfoundries, IBM, NEC, TSMC, UMC, Samsung, and Micron Technology either can manufacture the masks themselves or have joint ventures with the photomask manufacturers.

Figure 2.3: A single mask from a set of masks manufactured for IHP: the mask is manufactured in glass, the dimensions are 15cm x 15cm x 6mm. For the manufacturing of the transistors around 15 – 20 masks are required, for the metal layers there are 3 masks per layer needed.
Nowadays, depending on the light wavelength used in the photolithography process for the mask manufacturing, mainly two kinds of photolithography processes are applied:

- Deep Ultraviolet (DUV) photolithography process [28];
- Extreme Ultraviolet (EUV) process.

There are three main companies supplying the equipment required for the DUV/EUV photolithography process:

- ASML
- Nikon
- Canon

ASML is the dominating company in this field. This is partly due to the fact that EUV lithography is unique to ASML. The market data available showed that ASML also has a kind of monopoly in the immersion lithography market selling 68 systems in 2021 compared to Nikon that sold only 6. Thus, most mask shops are relying on the machines from ASML.

Mask order form examples can be found in [29]-[31]. Further useful links are [32]-[35].

### 2.3.2 Tools

Different tools for optimising the layout of the common layout are available e.g. k-layout, beneath widespread tools there are niche market tools such as the one from TexEDA.

### 2.4 Manufacturing

ASIC fabrication covers all steps needed to manufacture an ASIC starting from the GDS-II as an input, normally done by the foundry. Figure 2.4 shows an example of wafers from which some ASICS have been isolated as well as isolated ASICS manufactured in IHP.

![Figure 2.4: Wafers from which some ASICS have been isolated as well as isolated ASICS.](image)

The means to manipulate ASICS during manufacturing are limited. Complex changes to the functionality can be done by reverse engineering the mask sets. An example of the reverse engineering of a chip using its masks is described in [36]. To change/add the functionality of an ASIC design the mask set has to be manipulated after the complex reverse engineering but before chip manufacturing. Or the attackers try only to integrate
additional functionality into the attacked design, without a deep understanding of its original functionality. Also for this step the attackers need to manipulate the set of the masks in a short time period.

The situation looks different if there is a part of the design that is often used and that can be easily identified by the manufacturer and for which an already manipulated design is available. This may hold true especially for IP cores of interfaces such as SPI.

But, attackers – the malicious manufacturing fab – can still manipulate ASICs during the actual manufacturing process, e.g. by changing the doping in individual gates [37]. Even though complex additional functions cannot be integrated in this way, the desired functionality can be severely restricted or changed. Furthermore, it is possible to change the thickness of the wiring within the ASIC, so that goals such as “sabotage” and “damage to the reputation” of the ASIC suppliers could be realised through significantly more frequent and earlier failures of the ASICs.

### 2.4.1 Actors and responsibilities

The manufacturer performs wafer processing but may in addition offer services such as testing, isolating and thinning of ASICs, or even packaging. There are also companies specialised on these steps so the assignment of tasks to individual entities depends on the supply chain decisions of the “system development entity”. If ASICs are isolated by a third party, may it be the manufacturer or an additional service provider, some ASICs might be stolen and marketed under a different name.

### 2.4.2 Tools

Here, we consider the manufacturing process as a single step, as we are focussing on the interfaces in the supply chain, and consider manipulations of the machinery implausible. Therefore manufacturing machines are not discussed in this report.

### 2.4.3 Available Information

The manufacturer gets mask sets to run the production. So, it has the same information at least in principle as the mask shop and can in principle reengineer the functionality of the design, and then apply sophisticated changes. Approaches such as split manufacturing allow to limit the available information to a certain subset, which at least makes reengineering more demanding.

In some cases, the manufacturer gets GDSII data for composing the reticle. In these cases, reengineering the logic functionality is easier than from the masks. If complex manipulations are feasible at all mainly depends on the time interval between retrieving the masks or GDSII data and the expected delivery data of the ASICs. The shorter this time is the less probably manipulations are.

### 2.5 Test

For testing of ASICs two different types of tests need to be distinguished: production tests and functional tests. For the latter normally scan chains are included in the designs.

#### 2.5.1 Actors and responsibilities

Production tests is normally done by the manufacturer to ensure that the ASIC production was successful, to determine the yield of the production etc. Here the functionality of the ASIC in the sense of the realised logic is not tested, it is merely the electrical correctness i.e. detection of short circuits or similar issues.

Functional test can/should be done by the “system development entity”. To do so the design team needs to develop test vectors that are applied to evaluate if the ASIC provides correct answers/reactions.

During test no manipulations can be applied to the ASIC under test. The major threat is that scan chains, needed to run in depth functional test, and providing access to internal data are not properly disabled after the test and by that allow an attacker later to access sensitive data.
2.5.2 Tools

There are different vendors such as Agilent which provide test equipment. But analogous to the machinery for manufacturing we do not consider these machines relevant in the attacks as the main issue in testing are scan chains.

2.5.3 Available Information

If the test is run by the manufacturer all information that is provided to this entity is still available plus know how on their own process. If the functional test is run by the “system development entity” all information on the system under design is available. If the functional test is done by a subcontractor all information about the functionality of the system under development needs to be revealed to this subcontractor.

2.6 PCB Design

Designing the PCB is an essential step in the realisation of a certain product and according to the Bloomberg report, it is a critical step. During the design of PCBs, additional chips can be integrated into the overall circuit. These can fulfil different functions: espionage, implementation of backdoors or sabotage. For this, adaptations of the schematics by the attacker are necessary. Figure 2.5 shows the steps from designing the schematic of the PCB via layout and creating Gerber files down to the PCB fabrication and assembly which are discussed in Section 2.7.

Figure 2.5: PCB designing steps, from schematic of the PCB via layout down to the PCB fabrication and assembly. The result of the PCB design and PCB layout steps is a Gerber file using different editors. The Gerber data format is the standard file format for designing PCBs. It is a set of 2-D ASCII files containing information about the PCB extracted from a computer-aided design (CAD) file. Alternatively, a CAD file can be used. PCB NC drill files consist of PCB drilling and routing information. The NC formats were originally designed by CNC (Computer Numerical Control) drill and route machine vendors as proprietary input formats for their equipment; there are many NC formats.

2.6.1 Actors and responsibilities

Designing the PCB can be done by the “system development entity” or may be subcontracted to a third party specialised on PCB design. Here unintentional mistakes or simply bad design or designs realised without sophisticated know how on some issues such as side channel attacks may make a design extremely vulnerable. Design elements that have an impact on the SCA resistance are for example capacitors.

2.6.2 Tools

There are many CAD tools for designing PCBs available, examples are:

- Altium Designer
- Mentor Graphics
WP2: Design and Manufacturing steps from the initial idea to the final product

• Cadence software (Allegro PCB Editor, OrCAD PCB Editor)
• Zuken (CR-8000, CADSTAR)
• Solidworks PCB
• KiCad EDA
• Autodesk (Eagle PCB)

Which of the tools is used depends on the preferences of the different companies, in some cases a single company may even use two or more. We do not consider the tools as an issue in the design process it is merely the use of the tools, which is a threat.

2.6.3 Available Information

In case the “system development entity” is designing the PCB all information is available, this includes not only the functionality of the PCB but also the information on the ASICs designed and to be placed on the board. If a third party is realising the design of the PCB they need to get information of the functionality of the board, and pads of the ASICs that need to be connected on the PCB. In other words such a third party will get all information needed to manipulate the design and can adapt the design to its own purposes.

2.7 Production and assembly of PCBs

Production and assembly of PCBs can be carried out by a single or by different companies. In both cases additional chips can be integrated into the overall circuit, with all the aforementioned consequences to product reliability and security. Figure 2.6 represents a layer model of a PCBs showing the connections between different layers with the intended ASICs on top and bottom and some potentially malicious ASICs shown as dark grey rectangles embedded between the PCB layers. This last case might require more effort from the attacker, but is technically feasible.

Figure 2.6: Layer model of a PCBs showing the connections between different layers with the intended ASICs on top and bottom and the potentially malicious ASIC shown as dark grey rectangles embedded between the PCB layers.

2.7.1 Actors and responsibilities

Here we distinguish between the PCB manufacturer and the assembly company. The former produces the PCBs while the latter is equipping those with packaged ASICs, passive components etc. Beneath these two entities the “system development entity” is responsible for evaluating the quality and functionality of the assembled PCBs.

In June 2023 Eagle was announced to be discontinued, but it will be supported until June 2026.
2.7.2 Tools

We do not consider the production machines an issue when it comes to malicious manipulation which is the reason not to discuss them here.

2.7.3 Available Information

For complex manipulations of the PCB the attacker needs the schematics. If we consider the PCB manufacturer as the attacker, schematics might already be at hand or can be reverse engineered using the Gerber files. The level of information they provide to a PCB manufacturer depends on the design of the PCB. In principle, only PCB-pads and the connections between the pads are provided\(^6\). So, assessing the intended functionality might become very challenging. But for some chips e.g. PCI interfaces the footprints are quite unique providing additional information that was not directly given. In principle, it is even possible to hide ASICs in the PCB (see Figure 2.6).

If we consider the assembly company as attacker, manipulations of the PCB become more challenging. On the one hand there is only limited information on the PCB available on the other hand the PCB is already manufactured i.e. a piece of hardware that cannot be altered. But depending on the attacker’s goal, it is conceivable that such changes are not necessary. The assembly company knows which ASICs to put in which place and has the chance to integrate additional malicious ASICs by integrating those with the originally planned ones into a “system in package”. This means that multiple ICs are covered with a common plate or filling. These then fulfil their function in the package and are therefore optically undetectable or at least very difficult to detect. An example of two ICs inside a single package is shown in Appendix 2. An example of multiple chips inside a package can be found in [38].

2.8 Software/firmware development or use of existing ones

Given the complexity of today’s systems, not all software components are developed in-house. Firmware for embedded systems in particular often comes with the corresponding ASICs. Thus, the firmware is largely beyond the control of the system manufacturer and can intentionally or unintentionally represent a weak point of the system [39]. These changes are difficult to detect or can only be detected with great effort, which might be even more challenging than for hardware manipulations as the hardware design is under full control of the design team. With such changes, both espionage and backdoors can be realised. In the realisation of these attacks, there is usually no specific attack target in the sense of an explicit victim, i.e. there can be many widely spread infected systems. However, when activating the espionage functionality or using the backdoor, a specific victim is then selected.

2.8.1 Actors and responsibilities

Depending on the commercial model the responsibilities are divided between the “system development entity” and one or several potential customers. In the one case the “system development entity” selects the software that is installed on the hardware. This may cover especially drivers for communication interfaces, but possibly also protocols. Finally, the hardware is sold without any software installed and the customer will select firmware, drivers etc.

2.8.2 Tools

In order to manipulate software any software development tool can be used. So, listing tools would be misleading as even a text editor and the appropriate compiler are sufficient for such manipulation.

\(^6\) Pads are contact areas on a PCB or on a chip, see examples in Appendix 2.
2.8.3 Available Information

Independent of the responsibilities the information available is the same. All entities need to know which processors, interfaces etc. are put on the PCB. The type of available hardware components defines which software components can be deployed.

The software may be developed by the “system development entity” or the customer, or any of these entities are using software provided by a third party or open source software. Analysing software to ensure that it is free of any malicious functions is a tedious task and might be done with limited effort. Here open source software may look like an alternative as many people are cooperating and might detect such malfunctions. The heartbleed bug [40] clearly shows that this assumption does not hold true. In that case a developer was trying to improve an openSSL implementation but instead introduced a security leak which was discovered only long time after the change.

2.9 Summary

In Chapter 2 we discussed the potential risks so far identified for each of the different development steps, starting from the specification of the functionality via ASIC development and production via PCB design and manufacturing to software deployment.

• Specification of the functionality

In this step the major risk stems from malicious insiders who have access to all essential information and can manipulate the specification. In this document the design team is considered to be benign.

• Front-end design phase

In order to reduce cost and time externally realised Soft IP-cores are eventually used. Their core functionality i.e. the one for which they are included as well as additional (potentially malicious) functionality can be tested. This costs additional time which kind of contradicts the major motivation of their use. As the consequence, this testing/control of the functionality could be not done at all or not as thoroughly as needed. This is be especially critical for highly sensitive functions such as cryptographic operations. Manipulations of IP cores implementing different interfaces are eventually less critical but are in the end highly efficient from the perspective of the attacker as these cores will be used in many designs.

Beneath intentional manipulations accidental mistakes may seriously compromise the integrity of the ASIC. For example, developers of cryptographic designs have to be experts in SCA and FI attacks. A straight-forward implementation of a cryptographic algorithm will work correctly but will be vulnerable to physical attacks. The complexity of designing tamper resistant cryptographic functions is due to the fact that manufacturing technology, target frequency, applied gate library, placement of the gates in the design, length of the wires, etc. significantly influence the resistance of the design against attacks.

• Back-end design phase:

This step can be outsourced to specialised firms which then get full access to the VHDL code meaning the functionality can be as easily manipulated as a malicious insider could do. Also, in this step there is a certain chance for unintentional “errors”, that will have impact on the design’s quality. This can be due to hassle or missing expertise i.e. if a layout is done for a cryptographic design but without special know how on SCA and FI attacks. Similar issues hold true if he “system developer team” itself is doing the back-end design.

Even if this step will be done by the "system developer team" the integration of the hard IP-cores can be done without testing their functionality, so a malicious one might go undetected. In-depth testing requires debugging of the IP functionality at the IP block interface level. Ideally, the functionality of the IP will not be in question; however, without access to internal codes, debugging IP functionality can be a complex challenge.
• Masks development

The mask shops are provided with the layout data i.e. they have all information that is needed to determine the logical function of the design, given there is sufficient time and resources to do so. Changes in an ASIC design can also be achieved by manipulating the mask layout. To do this, the attackers must first determine the desired functionality of the design in order to then be able to integrate changes that make sense to them into the mask sets. But the following information is missing: functional description of the design, potential fields of application, thus:
- The effort to understand the functionality (reverse engineering) is quite high;
- The target of an attack is uncertain.

The situation is different if the design under attack is well known by the manufacturer/mask shop as it is often used and by that can be easily identified e.g. IP cores for interfaces such as SPI. In such a case reverse engineering is not needed. And the original design may be exchanged by an already manipulated one.

• Chip die fabrication

The potential risks and steps for potential manipulations are very similar to those described for mask sets, as the fab receives the masks as input.

• Bonding, testing, capsulation

Intentional or unintentional errors e.g. during bonding will lead to low quality and thereby negatively affects the reputation of ASIC manufacturer or the “system development entity” or both.

• PCB design, fabrication and assembly

Designing the PCB can be done by the “system development entity” or may be subcontracted to a third party specialised on PCB design. Here unintentional mistakes or simply bad design or designs realised without sophisticated know how on issues such as side channel attacks may make a design extremely vulnerable. Design elements that have an impact on the SCA resistance are for example capacitors. Also intentional manipulations are feasible i.e. additional chips can be integrated into the overall circuit. These additional ICs can fulfil different functions: espionage, backdoor, sabotage.

• Selection and integration of Software IP

Normally not all software components are developed in-house. Firmware for embedded systems in particular often comes with the corresponding ASICs. The firmware is largely beyond the control of the system manufacturer and can intentionally or unintentionally represent a weak point of the system. These changes are difficult to detect or can only be detected with great effort. With such changes, both espionage and backdoors can be realised.

• System testing

During test no manipulations can be applied to the ASIC under test. The major threat is that scan chains, needed to run in-depth functional tests and providing access to internal data, are not properly disabled after the test and thereby allow an attacker later to access sensitive data.
3 WP3: Identification of potential vulnerabilities and attack scenarios including a detailed elaboration of selected attack scenarios

The assessment of potential attack scenarios is presented in this chapter. The discussion of potential attacks, information relevant to successfully conduct an attack in a certain development phase, discussed in Chapter 2, are the basis on which this chapter builds. While analysing the literature we realised that some relevant publication in this area are using a different “clustering” of design activities, which we are adapting. This also includes phases that we did not consider earlier such as “maintenance and repair”.

In the following, we will discuss attack scenarios. Here, we will not only focus on discussing technical details, but consider relevant roles, responsibilities and organisational aspects relevant for the threat actors as well. This is in order to sketch out realistic attack scenarios and rule out others, e.g. those that might be technically possible but make unreasonable assumptions with respect to human factors. We start by giving a motivating example, which depicts the point of view of an attacker, in this case a malicious PCB design company. In the following we present the results of our literature research and the metric we used to assess possible attack scenarios.

3.1 Introduction

3.1.1 Motivating Example

Let us consider that the PCB design company that was subcontracted is malicious. Most probably not all PCB designs are manipulated and not all employees are collaborating in the manipulations. This means an attack needs to be planned well in advance to select the potential victim and to decide which persons in the company need to be involved. So, when a call for tender is published, our malicious company will analyse available information of the potential victim. This may be the field of business of the client, its potential customers, business partners etc. After this analysis is done, the attacker will decide whether an attack is beneficial/feasible. The next step is ensure getting the contract. One way is to tune the offer e.g. by requesting a cheap price that ensures that the contract is won, but is not so cheap that the offer raises any kind of suspicions. This means that the management of our malicious company or the finance department are at least partially malicious. Please note that a design engineer cannot mount such an attack on his own, as this person cannot influence the offer submitted or the final design review, which could reveal malicious changes.

Once the contract is signed, the attack team will be built. In this team there will be different roles: designer, tester and quality control for example. Depending on the size of the company these roles relate to different employees or eventually to just one. In any case the criminal master mind of the company will ensure that all team members that need to cooperate in ensuring to produce a malicious design are informed about the goal and are willing to contribute. The smaller the team can be kept the easier this can be ensured and the more probable it is that benign employees are not discovering the conspiracy.

The next steps are:

1. Analysing the data provided by the client
   These cover the schematic of the board, see Figure 3.1 for an example including the footprint of non-standard components e.g. specific ASICs, and the electrical connections of all elements, material of the PCB, number of layers and dimensions etc. Based on the result of the analysis the attack can be planned and conducted.
Figure 3.1: Schematic of a printed circuit board: it includes the footprint of non-standard components e.g. specific ASICs, and the electrical connections of all elements, material of the PCB, number of layers and dimensions etc. Based on the result of the analysis the attack can be planned and conducted.
There might be specialists in the company for different types of attacks such as impacting the reputation of the client, data theft and sabotage. For data theft as an example the following needs to be done:

- Finding the location to be tapped
  - One of the first steps is determining the positions of communication ICs such as USB, UART etc. That is not a difficult task, as from the schematic these components are known and will be positioned by the designer.
  - Analysing which IC can exchange data with which other IC(s) may provide hints on which of the data connections are of interest, i.e. shall be tapped. Here additional information e.g. about the application for which the PCB shall be used might provide an even better understanding. If such information is missing it might be an option to focus on data lines that are used for internal communication only, as they might carry sensitive information that is normally not revealed to the outer world.

- Identify triggers for data to be intercepted and connect if necessary. This can be done based on the assumptions about the potential data exchanged via the data lines to be tapped, or can simply be based on a time schedule. The latter might eventually be designed to be adaptable, so that once the attacker has learned more about the communication he can eavesdrop more often.

2. **Modifying the design of the PCB**
   
   After the analysis this step is slightly more straightforward:

   - Determining the location for an additional communication chip’s potential positions:
     - in the vicinity of the signal to be intercepted which is an easy task since the attacker routes the signal lines, and helps keeping extra wires short, i.e. helps to conceal the attack
     - in the best case at a VIA, as this most probably is the least noticeable
     - Preferably between the inner layers, as this is going to prevent quality control to identify the additional unplanned IC by optical inspection. Ideally, the attacker selects a position at which there is metal atop and underneath the additional IC. Depending on the metal used for the bond wires of the implanted chip it can or it cannot be detected when X-raying the PCB e.g. as part of the quality control.

The intention of this example is to show that and how a real-world attack can be realised. Please note that there exist many more potential attack scenarios, as we have shown in the previous chapter.

3.1.2 **Structure of this chapter**

The rest of this chapter is structured as follows. The following subsection strives to cluster attacks reported in the literature according to different development phases. The intention is to provide some insight in the probability and plausibility of potential attacks based on the number of references and type of references. This subsection slightly enhances the attack surface compared to the one described in Chapter 2, based on the new aspects found in the literature. As there exist of course many more leverage points for different attack scenarios than those discussed in the motivating example, Subsection 3.3 aims at providing a generally applicable means to assess the probability of a certain attack. While this is of course coarse grained, it allows to assess attack scenarios in all design phases.

### 3.2 Supply Chain Attack Literature Review

In Chapter 2 potential attack points in the development cycle have been introduced, providing also information required to run the respective attack. In order to provide a thorough assessment of the probability that a certain attack will really be conducted, a kind of plausibility check is required. While in the literature fancy attacks such as modifying mask sets are discussed by scientists working in the field, these attacks have not been reported outside the scientific community, and according to expert interviews these attacks are demanding and time consuming. Even though the attacks discussed in scientific articles are most
probably unrealistic at the time being they might become relevant in the future. So, we analysed scientific references and other sources such as official reports etc.

In our first search of the suitable references we concentrated on the hardware supply chain attacks. Only 12 different papers were found in Google Scholar using the keyword combination “hardware supply chain attacks” (plural) and only 8 different papers using “hardware supply chain attack” (singular). Moreover, the found references seem not to be applicable for the topic discussed in this report, as these word combinations were used either in a kind of short overview and/or for classification of the supply chain attacks or the papers concentrated on a specific task, for example on memory randomisation to prevent “hardware supply chain attacks”.

We extended the search, i.e. we decided to estimate the number of publications about insertion of malicious software and hardware. Therefore, we searched in Google Scholar with the following keywords:

- “Supply chain attacks”
- “Supply chain attacks” AND “software”
- “Supply chain attacks” AND “hardware”
- “Supply chain attacks” AND “PCB”
- “Supply chain attacks” AND “circuit boards”
- “Supply chain attacks” AND “integrated circuits”
- “Supply chain attacks” AND “ASIC”
- “Supply chain attacks” AND “FPGA”

Additionally, a very similar search was done using the key words “Supply chain attack” (singular) instead of “Supply chain attacks” (plural). Figure 3.2 shows the result of these searches. We marked the results corresponding to the search with the word “attacks” blue, and to the word “attack” (singular) grey, due to the fact that the set of the publications obtaining the singular form “attack” is not completely a subset of the other set.

![Figure 3.2: Google Scholar literature search results for “supply chain attacks” (blue bars) and for “supply chain attack” (grey bars) combined with search keywords “software”, “hardware”, “PCB”, “circuit boards”, “integrated circuits”, “ASIC”, and “FPGA” to get more detailed results.](image)

In order to provide insight into the overlap between the papers we analysed in detail and the mass of the relevant papers that we found in our literature search, we are discussing in the following which of these papers were reported back in our survey. An important fact here is that the term “supply chain attack” is rather new. In consequence, older publications, even though reporting such attacks, are not found when
searching with “supply chain attack” as key words. This holds true for example for the well-known Bloomberg paper [2] which is included only in the first 3 grey bars (the authors applied the word combination “supply chain attack” once but used the word “motherboard” many times). Also paper [41] reporting on a chip implant in a CryptoPhone cannot be found using “supply chain attack” as search key words, even though it discusses such a type of attack. We took both papers into account when analysing the state of the art. For more recent publications the above mentioned issues do not hold true, or at least they can be found with the applied key words.

The paper [42] overviewing malicious implants in PCBs throughout the supply chain is included in all the blue bars except the last one although FPGAs need to be placed on PCBs.

The Cyberdefense report about Software Supply Chain Attacks [43] is included only in the first 3 blue bars. This report was published in 01.2023 and references two important documents: the US MITRE Corporation report [44] and EU ENISA document [45].

The references [46]-[47] discussing supply chain attacks as an important aspect of recent cybersecurity threats are both in the first 6 bars of the charts only, because they contain the word combinations “supply chain attack”, and “supply chain attacks” as well as the words “software” and “hardware” but do not contain the keywords “PCB”, “circuit boards”, “integrated circuits”, “ASIC”, and “FPGA”. The document from the European Union Agency for Cybersecurity (ENISA) [45] are included in the first 3 blue and the first 3 grey bars. This ENISA document reports on known supply chain attacks, but does not concentrate on the hardware manufacturing process and its phases as the main supply chain attack target. It points to the MITRE documentation as the one concentrating on the different system development phases this is why we analysed the information from the MITRE documentation in more detail in this section.

The MITRE report published in 2013 [44] describes many supply chain attack scenarios. It is the basic document for all more recent documents that discuss the risks and analyse suitable countermeasures, providing guidelines for system engineers for risk mitigation [48]. It also introduces the Threat Assessment and Remediation Analysis (TARA) methodology developed by MITRE Corporation [49]. The MITRE report is contained in all bars excepted of the “PCB” one. The follow-up MITRE document [50] is covered by the first 6 bars but in addition also in the 11th and 12th bar.

In the rest of this chapter we are focusing on the MITRE report [44], due to the fact that it discusses different attack scenarios during the hardware manufacturing process. We selected those attack scenarios, which fit to our case study as a basis for the selection of the attack scenarios. Additionally, we refer to 15 other sources – (overview) papers as well as presentations – reporting about the attacks or describe processes and facts that can be exploited during an attack even if such an attack is not described in the literature, for example, inserting malicious hardware in masks sets for manufacturing of ASICs.

### 3.2.1 Attack scenarios selected from the MITRE report

In the MITRE document [44], attacks were classified according to several characteristics, and a catalogue of attack scenarios was developed. The catalogue includes 41 attack scenarios (numbered from A1 to A41). One of the characteristics of the scenarios is the “Attack Type”, which describes what type of malicious insertion – software (SW), hardware (HW), firmware, or system information - is the target of the attackers. The other characteristic is the phase of the manufacturing process.

There are 5 life cycle phases mentioned in the report:

- Material Solution Analysis (MSA)
- Technology Development (TD)
- Engineering and Manufacturing Development (EMD)

  Steps to be taken in this phase: iteratively design, develop, integrate, test, and deploy capabilities via regular releases [51]
• Production and Deployment (P&D) [52]

Steps to be taken in this phase: Low-Rate Initial Production; Limited Deployment, Initial Operational Test and Evaluation; Full Rate Production Decision or the Full Deployment Decision; Full rate production or full deployment

• Operations and Support (O&S)

The MSA and TD phases are considered as “pre-system acquisition”, which corresponds to Step 1 (Functionality Specification) in Chapter 2 of our report. The EMD and P&D phases are the “system acquisition” steps, corresponding to steps 2-5 in Chapter 2 of our report. The P&D phase is an additional step required for products to be received by military organisations, i.e. this phase is not a part of usual manufacturing processes but can be required when manufacturing critical systems sensitive to manipulations. Please note, that we concentrate in our report on the phases related to the “system acquisition” steps defined in Chapter 2, i.e. we concentrate mostly on the EMD and P&D phases in the MITRE classification. But as the phases TD and O&S are relevant, we are slightly widening the scope of our analysis and consider the TD phase as a part of the “System/functionality specification” step and the O&S phase as an “After the distribution” phase in the rest of this report.

According to the MITRE report, most of the attacks (25 hardware, 24 software, 16 firmware, and 6 system information attacks) can be conducted during the EMD and P&D phases, while some P&D phase attacks are also applicable in earlier phases.

All attacks described in the MITRE report are denoted using the letter “A” and a number. We concentrated on hardware attack scenarios, which we are discussing in detail in the rest of this section, i.e. we selected all attacks with “Target(Attack type)={Hardware}”:


Please note that we do not consider attacks targeting only the firmware, i.e. with “Target(Attack type)={Firmware}” but we observed additionally the attacks targeting system information, due to the fact that a maliciously altered system specification offers the possibility to easily insert additional functionality to the system. These attacks are: {A14, A16, A17, A30, A37}. In these attack scenarios the attackers are the “trusted insiders” from the main office or the main contractor.

The MITRE report observes 8 possible Attack Points. An attack point is defined as follows: it is the location at which, or the linkage through which, the supply chain attack is directed. For the attacks considered, the attack points are: {P1=main office, P2=main contractor, P3=subcontractor, P4=integration facility, P6=hardware developer-supplier, P7=physical flow, P8=information flow}:

A14, A16, A17 – “trusted insider”: P1
A7 – “trusted insider”: (P1, P2)
A30, A37 – “trusted insider”: P1, P2, P8
A36 – “trusted insider”: (P1, P2, P4, P8)
A8: - (P3, P4, P6)
A23 - “integration facility” only: P4
A2, A11, A15 – “physical flow” (only: P7)
A6, A22, A24 – “hardware developer” (only: P6)
A8, A5, A9, A10, A25, A28, A29, A31, A33 – “hardware developer” (P6) as one of possible attackers.

Attack point P5 focusses on a software developer/supplier and is not further presented in the selected set of the attacks. All selected attack scenarios are summarised in Table A3.1 in Appendix 3. These attacks and other attack examples reported in the literature describe the following malicious activities:

• Blocks with backdoors:
  - (without independent testing) (A6)

• Counterfeit component:
  - Supplied (A10, A28)
  - Implanted (intentional) (A25)
- Substituted (A34)

- Manipulated and/or badly tested cell/gate functionality, resulting in the use/acceptance of cells/gates outside the required functionality e.g.:
  - triggers were implemented using digital logic with analogue components—a capacitor and a few transistors wrapped-up in a single gate [53]
  - “usual” gate(s) used instead of radiation-hardened (see Appendix 4 and Appendix 5)

- Design specification manipulated (A14, A16, A17, A31)

- Data misconfigured or false data inserted (A30, A37)

- Design/fabrication is compromised (A22)

- Malicious integration facility: rogue processes for HW insertion (A29)

- Malicious SW integrated in replacement HW (A5)

- Malicious HW:
  - ASIC is designed with a malicious function (A24)
  - Malicious HW is substituted for a legitimate component:
    - during System test/integration (A9)
    - during Packaging/distribution at manufacturer (A33)
    - during Packaging/shipping/receiving/transportation during any transportation process (A2, A11, A15)
    - at the PDR (preliminary design review) timeframe (A8)
    - during sustainment (A23, A34)
    - by Direction (A7)
  - Malicious HW inserted/added:
    - NSA implanted beacons in CISCO routers [54]
    - Dell: HW implant exploits JTAG of Dell server’s processors [55]-[56]
    - USB hardware implant COTTONMAUTH provides a wireless bridge into a target network and the ability to load software onto a target PC [56]-[57]
    - listening device in a cryptophone [41], [56]
    - PCB altered
      - An implant can be added [1], [2], [58]
      - A component can be replaced (see Appendix 4 and Appendix 5)
      - Layout can be modified (see Appendix 4 and Appendix 5)
    - IC in package is possible (see Appendix 4, Appendix 5 and [59])
      - For example, using Through-Silicon VIAs [56]
    - As a “trusted” IP core [60]
      - it may contain an on-chip sensor as a Trojan that can eavesdrop on cryptographic operations across the whole device
  - IC modifications are considered possible in different ways, but the references are merely postulating such attacks but do not provide concrete examples:
    - Hard IP edit [56]
    - Netlist edit [56]
    - Masks edit [56], [61]
      - Dopant-Level Hardware Trojan [62]
      - replacing one of the mask layers, postulated in [63]
  - Firmware modification [64], (A7, A10, A15, A29, A33)
  - Additionally, implementation mistakes/design defects (accidental or intentional) can be exploited for attacks (see Appendix 4 and Appendix 5)

Additional information on malicious implants in PCBs via the supply chain can be found in [56], [57], [42], for IC Trojans please refer to [65]. The possibility to implant a chip into a PCB is discussed in [66]-[67].

The following graph gives an overview, in which system design phase the attacks listed above can be realised.
Supply chain attacks

<table>
<thead>
<tr>
<th>Insertion of malware, tainted HW</th>
<th>Substitution a bad/corrupt component for a good one</th>
<th>Modification to affect the performance or functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>System specification</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EMD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tiny Spy Chips in Hardware</td>
<td>Triggers with analog components [52]</td>
<td>Hard IP core modifications, Netlist edit [55]</td>
</tr>
<tr>
<td>[55], [58], [66]</td>
<td>Hardware IP core with additional malicious functionality [59]</td>
<td>Firmware modification [63]</td>
</tr>
<tr>
<td></td>
<td>PCB [55], IIP examples from interviews (Appendix 4-5)</td>
<td>Masks edit [55], [60]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dopant-Level Hardware Trojan [61]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Replacing one of the mask layers, postulated in [62]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PCB [1],[41],[57]</td>
</tr>
<tr>
<td>EMD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P&amp;D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>After distribution</td>
<td></td>
<td></td>
</tr>
<tr>
<td>logistic in any phase</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cisco routers [53]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dell server’s processor [54]-[56]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USB hardware implant [55]-[56]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.3: Overview, in which system design phase the attacks listed above can be realized.

The life cycle of a supply chain attack can be separated into 7 stages, corresponding to [50]: reconnaissance, weaponise, deliver, exploit, control, execute, and maintain. In the EMD (Engineering and Manufacturing Development) and P&D (Production and Development) phases all attack life cycle stages can be represented. Attacks against the supply chain can cause and exploit different delays. This gives more time for the other attacks that are specifically targeted at modifying HW [68]. Please note that the repair service of photomasks [69] and reverse engineering of photomasks [70] can – theoretically – be used maliciously, but these manipulations are time consuming operations. Attacks are most effective when started in an earlier stage of the acquisition life cycle and continue through all manufacturing phases to “operations and supports”, i.e.
attacks starting in the phase of the system specification are the most dangerous ones, due to their high success expectation. Many attack scenarios listed in Table A3.1 (see Appendix 3) assume involving malicious insiders. Countermeasures against the malicious insiders are important especially due to the fact, that attacks are most effective when started in earlier manufacturing process phase. We excluded the insider in this report (see Section 1), but more information about the insider aspects can be found in [71].

The complexity of the HW designs [56] needs to be considered as beneficial when inserting malicious HW. The functional verification of ASICs requires a lot of time and is a complex and expensive process, even denoted as the ASIC verification crisis [72]. The lack of meaningful metrics for measuring security at the system level is another aspect beneficial for the attackers [73]-[74]. In the literature we did not find an attack scenario in which an attacker prepared its attack without any knowledge of the mission/target device with the goal to spread a malicious component in so many devices as possible. Additionally, implementation mistakes or design defects (accidental or intentional) can be exploited for attacks (see Appendix 4 and Appendix 5).

[75] reports on attacks against CAD & manufacturing machines. An example of an attack against a 3D printer is a malicious Trojan that can compromise the quality of additively manufactured prints and reduce tensile strengths up to 50%. A malicious HDL Toolchain allowing to generate malicious designs from clean HDL sources for FPGA is reported in [76].

In Chapter 4 we will discuss the complexity of running attacks against hardware designs as well as potential countermeasures in detail.

### 3.3 Evaluation of Attack Scenarios

Analysing the probability that a certain attack will be run as well as assessing their severity is a tedious and challenging subject. Here, we focus on attack probabilities as their impact highly depends on the target. The probability that a certain attack is run depends on:

- The intention of the potential attacker, which is normally unknown, unless the attack was already executed and the attacker caught. Here typically the following dimension can be considered:
  - type of attack e.g. espionage, sabotage or impacting the reputation of the victim
  - long term or short term gain in the view of the attacker
  - the attacker’s position in the supply chain

- The skills an attacker has and the effort he can spend (time and money)

In order to allow for an intuitive assessment we developed a 3-dimensional scheme in which the probability of an attack at a certain step in the supply chain can be determined:

- Aiming accuracy: describes the probability that the attacker will hit a specific victim. This value expands between a single company or the largest possible number of companies
- Effort to run the attack: this depends on how much information and skills necessary to run the attack is available to the attacker,
- Design/development phase in which the attack will be launched

Based on our experience and discussions with experts we assigned a certain probability for an attack in the following steps:

- Very low: meaning that an attack is not reasonable, but cannot be excluded for sure
  - The classification is based on the following assumption: the attack does not or only marginally contribute to the attackers goals; the attack is extremely expensive in terms of cost, time or skills to be acquired, so most probably any other attack is more reasonable to the attacker.
- Low: meaning that based on the information available, an attacker would rather select a different type of attack or different step in the supply chain
The classification is based on the following assumption: the attack contributes only to a certain extent to the attacker’s goals; the attack is expensive in terms of cost or skills to be acquired, so most probably any other attack is more reasonable to the attacker.

- **Middle:** meaning there are pros and cons to run the attack but no clear indication if or if not an attacker would do it. Here minor issues will motivate an attacker to select this attack
  - The classification is based on the following assumption: the attack allows to achieve the attacker’s goals to a certain extent; the attack is feasible in terms of cost or skills to be acquired, so going for this attack depends on very specific factors e.g. attacker’s experience with the attack, access to information that allows to run this attack, i.e. it might be triggered merely by the opportunity to run the attack than by hard/measurable facts.

- **High:** meaning there are good reasons to run the attack for example because the attacker has a clear picture how to achieve its goals with this attack at this point in time
  - The classification is based on the following assumption: the attack allows to achieve the attacker’s goals; the attack is feasible in terms of cost or skills to be acquired.

- **Very high:** meaning the pros to run the attack clearly outweigh the cons
  - The classification is based on the following assumption: the attacker can be pretty sure that the attack will be successful, the attack is doable in terms of cost or skills the attacker is willing to spend and has acquired respectively.

The following paragraphs are illustrating the idea of the assessment of the attack probability in a certain design/development phase based on the motivating example from Section 3.1.1.

The attacker runs a PCB design house, so the selection of the design/development phase is predetermined. There are two types of impact the attacker can aim for:

**Impacting the reputation of a specific company:**

- If the customer i.e. the client is the target, this phase is close to ideal as the client is known. So here, the aiming accuracy is considered to be very high.
- All information to design a PCB are provided by the client, eventually additionally needed information can be extracted from the data provided by an experienced designer, i.e. here the hurdle to run the attack is very low.
- The skills the attacker needs to run the attack are available, as PCB design is the core business of the attacker i.e. there is no hurdle to run the attack with respect to skills and especially generating a PCB design of low quality is rather easy, i.e. the demand in skills is rather very low to low.

Based on the assessment above it can/needs to be considered that the probability that such an attack will be run is very high, see Figure 3.4.
Espionage on a specific end customer:

- The attack comes with a certain risk of not being successful, as the end-customers of the client might not be known by the attacker. So the aiming accuracy here is considered medium to low.

- All information to design a PCB are provided by the client, eventually additionally needed information, can be extracted from the data provided by an experienced designer, i.e. here the hurdle to run the attack is very low.

- The skills the attacker needs to run the attack are available, as PCB design is the core business of the attacker, but in addition:
  - a backdoor needs to be built in,
  - software that helps to find a such a modified PCB after its manufacturing can be found,
  - communication software needs to be built that allows to integrate a kind of a command and control software once the PCB is in use.

The aforementioned points require additional skills, so in total the necessary skills are considered medium to high.

The assessment above shows that there are pros and cons when planning this type of attack, so its probability can be considered to be medium and strongly depends on the skills available in the PCB design house, see Figure 3.5. Selected attack scenario evaluations are given in Appendix 6.
### Determining the Attack Probability

In this subsection the way to determine the probability of an attack is explained. There are two graphs presented which show the probabilities for different attack scenarios. In principle the way to determine a certain probability is straight forward. The two core factors are the complexity of the attack, that needs to be assessed and the accuracy of the attack at which the attacker aims for.

The following graphs represent two examples of assessing attack probabilities in different design phases. Here, the assessment is less specific than in the previous example, which was quite elaborated as it was used for an already defined attack scenario. The following two examples are thought to be more general to showcase how a risk assessment could be done in order to select appropriate and cost-efficient countermeasures.

1. **Assessment of the probability of an attack with the goal of espionage.**
   
   In this case the major factor influencing the attacker’s decision is considered to be how good the attacker wants to know which target he is hitting. For the ASIC design phase the probability is zero as we decided to exclude insider attacks in that phase for this report, see Chapter 2. There are two different approaches taken into account:
   
   - Trying to affect as many potential systems as possible and hoping some might be of interest, leading to high probabilities in rather early design phases.
   - Trying to hit selected targets while being sure exactly those will be affected, leading to very high probabilities in rather late stages of the development or even life cycle.

   For detailed results please see Figure 3.6.

2. **Damaging the reputation of the ASIC design house.**

   The main factor to decide about in which stage the attack shall be launched is the confidence of the attacker to hit the intended target. Here, all design and manufacturing steps after designing the ASIC are considered equally probable. This is based on the assumption that the ASIC design house also orders the PCB design and carries out the manufacturing. If the ASIC would be marketed as an ASIC, not as a system (in the sense of a board), the probabilities in the stages PCB design and PCB manufacturing would decrease. In that case, they might be as high as shown for the case when the target would be the company ordering the PCB design (cf. Figure 3.4) and its manufacturing respectively. For the first design phase, i.e. infecting IP cores, the probability of an attack is considered rather low if the attacker has a specific company in mind, otherwise it might be rather high. For detailed results please see Figure 3.7.

---

*Figure 3.5: Assessment of the attack probability of the motivating example when aiming at espionage of specific customers.*
Figure 3.6: Assessment of the attack probability per design phase when aiming the attacker aims at espionage of specific customers. Since the probabilities in most of the first stages (IP Core to PCB Fabrication) are inversely proportional to the respective aiming accuracy, some of the bars were covered by others, so we use a different angle of view in this figure compared to the other figures in this report.

Figure 3.7: Assessment of the attack probability per design phase when the attacker aims for damaging the reputation of the design house.
3.4 Attack costs

Determining the costs of an attack is extremely difficult. This is due to the fact that the cost is a mixture of different features such as available information, technical skills of the attacker etc. which are hard to quantify in terms of money. We could not find any concrete assessment in the literature. The only statement that we found is from the guest lecture of Andrew “Bunnie” Huang in MIT in 2022 [56]. But it provides rather vague and merely qualitative assessments. In addition, we undertook several IHP expert interviews to get their assessment of the complexity of different types of attacks in different stages of the life cycle (see Appendix 4 and Appendix 5). Both assessments are given in Figure 3.8. Those based on [56] are represented by a dark grey arrow reaching from 1 million USD and a couple of months to run the attack down to a few USD cents and seconds to run the attack. So, it is a coarse-grained estimation, which assumes that a complex attack requires not only a long execution time (months) but also expensive equipment (millions USD). Attacks aiming to add a component to a PCB are classified as a non-complex attack requiring only a few USDs and a couple of weeks for the execution.

Corresponding to IHP expert interviews, such attacks are possible if some preparation steps, i.e. a change of the PCB layout with the goal to insert an additional component later, were performed. In each case, a modification of PCBs, such as replacement of a specified element with a counterfeit one, can be executed easy and fast. The assessment based on the IHP expert interviews is only given in terms of time split into time intervals reaching from months down to hours (see light blue bar). For example, a netlist edition attack, i.e. altering the original netlist, can be performed within a few hours or can require some months of work depending on the complexity of the modification required. Similarly, a code modification can require a few minutes only or a couple of months, especially if the functionality caused by the code edition has to be included in the specification and tested. This is one of the reasons, why the code edition attack can require more time than a netlist edition attack. The time intervals in Figure 3.8 shows the time an experienced attacker would need to run the attack, if he has all required information at hand and does not consider the costs of the equipment. The attacks addressing different modifications during the design process are marked in light blue.

Also, the complexity to detect a certain manipulation is indicated in Figure 3.8. The earlier in the life cycle an attack is executed the harder it is to detect such a manipulation. This assessment holds true for both sources, i.e. [56] as well as information given in Appendix 4 and Appendix 5.

The only cost in this mixture that can be more or less easily determined are the cost of the equipment needed to run the attack. The cost of a focused ion beam station (FIB), which is a device that allows the modification of ICs at the transistor level, is significant i.e. in the range of about 1 million Euro, not to mention the cost for ensuring that the basis on which the FIB is mounted ensures to cancel any vibrations to allow proper operation of the FIB. In addition, an operator needs to be trained which adds to the cost. This cost assessment suggests that attacks requiring a FIB are prohibitive expensive. But, there are service providers that offer FIB based modifications of ASICs on an hourly rate of about 250 Euros. Another example showing that equipment costs are not prohibitive is the “self made” optical attack system based on a commercial CCD camera system published in [77] which allowed a photo emission attack against an AES implementation at a cost as low as 50.000€. This price assessment neglects the time and expertise needed to build the system. But even if additional 250.000 € are considered for 3 years of development effort the cost is still significantly lower than the one of a Triphemos system, produced by Hamamatsu, which was till then considered to be required to run photoemission attacks. The price of the Triphemos is in the range of 1 to 2.5 million € and additional cost for training an expert to operate it needs to be taken into account. This clearly shows that equipment cost is not prohibitive when it comes to the question whether or not a certain attack will be run.

---

7 We cannot provide a reference to this number, but it is what we recall from the presentation of [77] during CHES 2012
The most limiting cost factor is the information available and needed to run a certain attack. The more information about the ASIC and PCB under development are available the better an attack can be planned and realised. In our motivating example almost all information will be provided by the potential victim as it is required to design the PCB. So, in this example the information is available for free, and of course will not be preventive. Another rather extreme example is modifying the ASIC under development by reengineering mask sets. This requires skills to do reengineering, understand the functionality of the ASIC and finally to adapt the design. Then a new mask set needs to be manufactured. The complexity of these steps is pretty high (see Appendix 4 and Appendix 5) and under normal conditions the time interval in which they need to be completed is rather small.

Figure 3.8: Qualitative cost assessment based on [56] and IHP experts interviews (see Appendix 4 and Appendix 5).
The cost to acquire sufficient information to run an attack varies a lot depending on the development phase in which the attack shall be mounted.

To summarize it is almost infeasible to determine the cost of an attack in a general case, as it depends on the phase of the development and the skills an attacker already acquired before deciding to go for an attack. In addition, the cost an attacker is willing to accept depends on the attacker’s type i.e. individuals, criminal organisation, companies or states as well as on the attacker’s goals e.g. vandalism, sabotage, espionage etc. If the attacker has criminal intentions, cost may be prohibitive if the potential monetary gain is less than the cost. But, if the intention is terrorism or cyber war the attacker may be willing to accept any cost.
WP4: Prevention and detection options, risk assessment and recommendations

This chapter will cover the state of the art with respect to hardware Trojan detection. Here we are focussing on hardware Trojans in ASICs only, as the design and manufacturing flow when using FPGAs are separate [78], i.e. many of the issues discussed in this document are not applicable to FPGA based designs. We keep this literature review rather short as there are many references and the insight provided by exhaustive discussion of the research done is rather limited. Instead we will provide insight into two experiments we carried out with respect to integrating a hardware Trojan into a cryptographic design in an FPGA and to hiding an additional IC in a rather complex PCB. While the former is discussed in literature along the lines of hardware Trojans, i.e. addition of logic that is realising some kind of attack, the latter was inspired by the attack reported by Bloomberg [1]. In the experiment described here some effort was made to hide the additional IC which goes beyond what was reported by Bloomberg, despite in our experiment the IC is not functional.

4.1 Hardware Trojan Detection

The literature review provided here is mainly inspired by the following survey articles:

- Mark M. Tehranipoor and Farinaz Koushanfar: A Survey of Hardware Trojan Taxonomy and Detection, IEEE Design and Test of Computers 2010 (1500+ citations)
- J. Francq and F. Frick: Introduction to hardware Trojan detection methods, Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 2015, pp. 770-775, (81 citations)
- Ayush Jain, Ziqi Zhou and Ujjwal Guin: Survey of Recent Developments for Hardware Trojan Detection, IEEE International Symposium on Circuits and Systems (ISCAS) 2021 (23 citations)

4.1.1 Trojan detection power analysis

Agrawal et al. [79] discuss the construction of “fingerprints” to detect Trojans. Parameters that have been considered are power, temperature and electromagnetic irradiation. In order to have a golden device they propose invasive analyses of at least a few devices. The authors mention that their approach is not scaling to large ASICs. Wang et al. [80] propose to measure currents locally i.e. at selected PADs. The idea is that the additional power consumption can be detected better in this way as the overall power consumption per PAD is smaller than for the complete chip and by that the impact of the Trojan is much higher. A golden die can be identified by running exhaustive tests for several randomly selected dies. If all these dies deliver the same results they are considered to be Trojan free. Trojans are identified via comparing a pattern set recorded using localised power measurements at individual PADs.

4.1.2 Timing-based analysis

Li and Lach [81] proposed to integrate shadow registers to detect differences in chip latency. The results latched by the destination register and the shadow register are compared during every clock period. This approach will lead to significant overhead in area and in design time as all sensitive paths need to be identified and adapted. In addition, as the shadow registers are integrated in the design, an attacker can identify them and alter also the paths leading to the shadow registers. Jin and Makris [82] proposed to detect Trojans by analysing path delays. In principle they are generating “fingerprints” of the whole chip. In order to ensure that the fingerprints are benign they propose to reverse engineer the complete chip. In addition, they use statistical analyses to deal with process variations. One of the major issues with this approach is, that modern large designs can include millions of paths i.e. measuring all paths is not practical.
4.1.3 Trojan activation

Region-free Trojan activation techniques do not rely on the region but depend on accidental or systematic activation of Trojans. Jha and Jha [83] showed that it is feasible to generate a specific input pattern that generates a unique probabilistic signature of the chip. For detecting Trojans they applied this input pattern and compared it with the chip under authentication’s output. Yet it is still open if this approach can provide reliable results, especially if the Trojan is independent of the inputs. Banga and Hsiao [84] propose a vector based approach to detect Trojans. They apply each vector several times to the chips under examination. If there is a Trojan, a certain region will be more active and is later on analysed to identify the Trojan. The major downside of this approach is that the authors rely on having both, manipulated and genuine chips at hand, which is essential to detect if there is a region with enhanced activity.

If a Trojan circuit’s inputs come from the part of the circuit where they are functionally dependent (i.e. part of the same logic cone), the region-aware method can be effective. However, if the Trojan inputs are randomly selected from various parts of the circuit, region-free methods could increase the probability of detection [85]. The issue with both activation methods discussed so far is, that test engineers can only test functionality that they know of. In addition to the issue that they do not know if a Trojan is present in the first place they do not know the Trojan’s type or size, so both region-free and region-aware methods must be applied to be on the safe side.

The issue of Trojan activation is still a hot research topic and researchers are recently investigating the use of machine learning techniques [86]. The size of the design to be evaluated is still a challenge, and it is uncertain if a reasonable number of potentially inserted Trojans will really be triggered [86].

Machine learning was combined with computer vision algorithms in order to detect Trojans in a chip. Vashistha et al. [87] are proposing to use a trusted GDSII layout as a “golden” layout and SEM images of an IC under authentication to generate unique descriptors. These are then used to detect if there is a Trojan. The descriptors can take process variations into account to avoid false positives [87]. Another detection technique is using images of the chip under authentication [88]. The ICs are scanned from the backside, which allows to identify flipflops (FF). When the locations of FF used in the scan chain of the chip are known, additional FF are then indicating potential Trojans.

Beneath the Trojan detection techniques which have their specific shortcomings, approaches that rather focus on preventing the insertion of Trojans or to facilitate easier detection of Trojans are discussed in the literature [85]. Some of these approaches will be reviewed in the following paragraphs.

As mentioned earlier the activation of Trojans is an open issue. In Salmani, Tehraniipoor, and Plusquellic [89] the authors propose a methodology that helps increasing the probability of generating a transition in functional Trojan circuits. The core idea is to add dummy scan flip-flops to those parts of the chip that have a low transition probability. This approach can improve the detection of Trojans when power analysis is applied. Also Banga and Hsiao [90] are focussing on improving the detection of Trojans by increasing their activities. Their idea is to invert the voltage scheme so that states which occur rarely under normal conditions are occurring more often. The challenge here is that inverting the power supply i.e. switching between power supply voltage and ground for each gate on the circuit is normally not supported. A slightly different approach is the one of Abramovici and Bradley [91]. They are proposing to integrate additional logic into the chip which is then monitoring the behaviour of the chip and to launch countermeasures. The limitations are that an attacker can also manipulate the monitoring logic, and that it is tricky to implement such a monitoring logic without knowing what type of Trojan needs to be detected. A similar approach is introduced in Chakraborty, Paul, and Bhunia [92]. Their idea is to integrate additional logic which can be activated on demand and to which a specific input is applied. A Trojan will be detected if it influences the output that is expected if the specific input is applied. The challenge here is, as for other approaches as well, that the additional logic might be tampered by the attacker, and it is a real challenge to ensure that all potential manipulations will be detected.
Detecting hardware (HW) Trojans by optical inspection was discussed in Bhasin et al. [93]. Here the genuine GSDII data, that was used as hard macro is compared to images of the top metal layers recorded with an optical microscope with a 150x lens. The positive aspect is that this approach does not require reverse engineering but relies on comparison of images only. The open issue is if a Trojan can be detected if it is routed solely in the lower metal layers. Such modifications may go undetected as higher metal layers are blocking the view, or delayering is needed. Also Courbon et al. [94] propose to use image comparison to detect HW Trojans. The difference to Bhasin is that in [94] delayering is proposed to get images also of the transistor layer. But also the method introduced by Courbon does not require reverse engineering. It relies on comparing images taken from the chip to GDSII data or images of a golden die if available.

### 4.1.4 Runtime monitoring

Most of the approaches discussed in literature are focussing on HW Trojan detection at test time. As mentioned earlier this is challenging as it is unknown when a HW Trojan becomes active. So a reasonable approach is to go for online detection of HW Trojan activities. This avoids the issue of finding the right trigger and allows for observing all ICs. A potential shortcoming is however the effort for determining HW Trojan activities at run time. Bao et. al. [95] report on using thermal sensors as the means to determine HW Trojan activities. This is motivated by the fact that according to the authors modern designs already incorporate thermal sensors, which helps to keep the overhead low. No numbers are given for the computational overhead introduced by the Kalman filters used to decide whether or not a Trojan is active. Unfortunately, the authors do not report the size of the Trojans detected and their distance to the sensors. So there is still a certain risk that small Trojans go undetected as their influence on the thermal budget is too small. An approach that is very similar to the idea of x-modular redundancy in safety critical systems was introduced in [96]. Here the authors assume that a multicore system is to be protected. The main idea is that several processing cores are executing different but functionally equivalent software and that the results are compared. If they differ there is a certain probability that a Trojan is active. The authors assume that eventually all processing cores are infected by HW Trojans but that not all Trojans will strike at once, i.e. that they have at least slightly different trigger conditions. This approach is applicable only for multicore systems, i.e. it cannot be generalised, and there is still the risk that an attacker is also manipulating the detection logic.

### 4.2 Prevention

Bhasin et al. [93] discussed preventing the insertion of hardware Trojans by ensuring that the layout has a high density, above 80 per cent in their recommendation. The idea behind this is that an attacker cannot insert a HW Trojan in such a design without changing the layout or the logic. Although this does not really prevent the insertion of a HW Trojan, it might simplify its detection, as re-routing most probably will affect the top metal layers as well which then can be detected by optical inspection.

Ngo et al. [97] introduced the idea of using codes from signal processing to encode states in ICs. With this method mainly sequential parts of the ICs are encoded. This means each state is represented by at least x bits. The point here is that the attacker then needs to use all these x bits, which makes potential Trojans larger and therefore easier to detect, eventually even by optical inspection. Furthermore it becomes also more challenging to understand the IC under attack. The area overhead caused by this approach is significant. A code with x=13 (10 is considered the minimum length) doubles the number of sequential gates and leads to 5 times the number of combinational gates which in total results in 6 times the area. Since area is the crucial factor when it comes to the cost for manufacturing an ASIC such an increase in the chip area has a significant impact on the cost.

Dupuis et al. [98] propose to extend designs with an additional logic which helps to ensure that paths with a low probability of being activated are more often activated and thereby reduce the chances to integrate a stealthy HW Trojan. The design is working correctly only if the correct input values, named key, are applied to the additional logic. Similar approaches are named logic encryption or logic locking. Logic locking is increasing the complexity for integrating a hardware Trojan, as understanding the functionality of the IC is
getting more complex. Recent attacks against logic locking are using machine learning and genetic algorithms respectively. Logic locking cannot provide 100 per cent protection against HW Trojans manipulating the functionality of the attacked design. It also does not provide protection against manipulations of the quality of the IC after production. Furthermore providing additional logic results in extra area i.e. cost for manufacturing.

4.2.1 Split manufacturing

Split manufacturing refers to a method in the field of ASIC production in which the wafer passes through different fabs. With this type of production, it is not necessary for the participating fabs to have the complete mask data. The feasibility of this approach was reported in [99]-[101]. Also IHP offers such a service. As split manufacturing limits the information about the complete design it is considered to be a reasonable means to prevent the insertion of hardware Trojans [99], [102], [103]. In the past several techniques to reverse engineer the complete designs from incomplete data [104]-[106] as well as techniques to defend such attacks were researched [104], [105], [107]-[109].

A common approach is to separate the front-end and the back-end. The front-end, the active and passive components such as transistors, resistors, diodes up to and including the contacts, are manufactured in the first fab. The second fab is exclusively responsible for the back-end, i.e. all wiring levels. The manufacturer of the back-end only needs the mask data of the contact module, i.e. the placement and size of the respective contacts in active areas. In special cases, even this information could be dispensed with, but this means that the customer has to work very carefully when generating the data. In addition, the contacts on the wafer are also visible and thus not passing on the data on the contacts does not offer any further gain in security. The split can be made also at a certain metal layer, whereby splitting the information at higher metal layers does not provide that much security [103]. In the following we will focus on split manufacturing with a split between front-end and back-end.

For a reasonable combination of front-end and back-end in different fabs, the offered technologies or node sizes should be similar. In the cooperation between IHP and X-FAB8, 130 nm technologies are used in each case. The 130 nm BiCMOS front-end of the IHP fits well with the 8-layer back-end of the 130 nm CMOS technology of X-FAB, since a 7-layer back-end is used on the IHP side. In contrast, combinations of e.g. 22 nm front-end and 130 nm back-end technologies are not feasible because the contact sizes do not match. Nor is it possible to exploit the technological limits, because the small-scale nodes require significantly more wiring levels. In some cases, up to 20 metal levels are necessary. A combination of 130 nm front-end with a back-end for smaller technology nodes, e.g. 22 nm, is technically feasible. However, a large part of the additional wiring levels is idle here and the costs are unnecessarily high.

Furthermore, it must be taken into account that components are often integrated in the back-end. The most common is a plate capacitor between two metal planes, called a CMIM. While a CMIM is not important for digital applications, it is all the more important for analogue ASICs.

The wafers used in the process also play a role. They must be compatible in terms of material and size. In the best case, both fabs use the same wafer size. The use of larger wafers for the first sub-steps is conceivable. For the next sub-steps in another fab, the wafers can then be brought to the required diameter.

In addition to the node size, the materials used in the process are also important. A wafer that is part of a process with copper cannot be transferred to a fab that has aluminium in the line. At this point, contamination of the equipment with copper would take place, which is difficult or impossible to eliminate. To this end, appropriate investigations must be made in advance of split manufacturing to avoid such contamination.

In a standard process, in which the front-end and back-end are produced in the same fab, all technological processes are coordinated with each other. When setting up a split manufacturing process, it must be checked whether the back-end process influences the behaviour of the components in the front-end. For

---

8 https://www.xfab.com
example, a higher temperature during metallisation can lead to a shift in the transistor parameters and thus the operating point.

In summary, the wafers, the materials used in the process and the processes themselves must be compatible with each other. In the best case, neither the front-end nor the back-end processes need to be adapted. Each adaptation step requires a lot of effort and sometimes has an impact on reliability. In case ageing tests have to be repeated, for example, the costs and effort are high. Once the flow is established and proven stable, it can also be used on a larger scale, i.e. for higher quantities.

A PDK (Process Design Kit) is necessary for the design of an ASIC. It contains all relevant information about the components, wiring levels and also control checks like Design Rule Checks (DRC) and Layer-versus-Schematic (LVS). Typically, a PDK is supplied by the foundry. In the case of split manufacturing, a PDK is needed that correctly maps and checks both processes. Since the customer does not have the knowledge of the PDK implementation, it makes sense to put the PDK creation in the hands of one of the two fabs involved. If necessary, a tool manufacturer could be enlisted as a third-party provider for PDK creation.

Note: the creation of the PDK by one of the participating fabs does not represent a security risk.

Using two PDKs, i.e. one PDK for the front-end and one PDK for the back-end, is not practical. Rule checks (DRC) are possible, but no LVS can take place. This is indispensible for a successful tape out. Without such a check, the risk of a layout error is extremely high. Furthermore, without LVS, no extraction of parasitic effects (resistances, capacitances) is possible. The smaller the technology node used, the greater their influence. From 250 nm and smaller, an extraction of the parasitic elements should take place.

This results in additional work for the customer. In the standard flow, i.e. the production of the ASIC in one fab, the layout data is delivered to the fab and checked.

In the case of split manufacturing, the data of the final layout must be cleanly separated by the customer. This means front-end data for one fab and back-end data for the other fab. The data must then be delivered to the fab and checked there. An error in the front-end usually means that the back-end has to be adapted again. The design verification should be all the more careful and the data generation step for the final production is more complex.

There are many factors to consider for split manufacturing. In addition to the choice of suitable partners and materials, additional effort is required for PDK creation, compatibility testing and additional effort on the part of the customer for data verification.

### 4.3 Hardware Trojan Defense: A resume

All approaches analysed have their individual shortcomings. But even more important, some deficits are more general and in one way or the other apply to all of them:

- **All approaches focus on either detecting or triggering the hardware Trojan, but not on both. The issue is that measurements can reveal a HW Trojan only if it is active during the measurement. In addition, HW Trojans will not be detected during fabrication tests, as the test engineers are relying on the test vectors provided by the ASIC designers. This means under normal conditions they do not search for “hidden” functions, and even would not be able to do so.**

- **Most of the approaches rely on the availability of a golden device. This is – in our opinion– extremely unlikely, see also Francq [110]. So, on the one hand why should an attacker provide both devices especially as the production costs will be increased significantly by manufacturing both devices. In addition, it should be clear to the attacker that providing two types of devices simplifies the “defenders” task. There is a chance that non-manipulated devices are manufactured in a first run due to insufficient time to mount a sophisticated attack. The latter is then implemented before a second batch of devices is manufactured.**

- **Even if there are golden devices, determining the golden ones requires reverse engineering to detect a HW Trojan. With shrinking technology this becomes more and more challenging, see also**
Francq [110], and Gubbi [111]. Or the detection will be limited to top metal layers which clearly reduces the probability of detecting sophisticated or deeply buried manipulations.

- In cases in which third-party IP blocks have been used that were manipulated, no golden device or simulation will be available as all device as well as the simulations are affected by the manipulation.

- Some approaches rely on a “golden simulation” instead of a golden device. Simulations can be very close to the manufactured device, but this requires a high quality characterisation of the manufacturing technology. If the foundry is under suspicion and planning to implant a HW Trojan, it is more than probable that the foundry is also manipulating the characterisation data to ensure deviations caused by the implanted HW Trojan stay undetected. This may also happen if an outdated library is used for simulation while a more advanced process is used for manufacturing, see Gubbi [111]. Also process variations will cover deviations to a certain extent, and an open question still is what the minimal size of a HW Trojan is to make sure it will be detected. In other words if the HW Trojan is small enough it will go undetected independent of the availability of a golden die or golden simulation. An approach related to “golden simulations” utilizes a comparison between images of the reverse engineered target ASIC and GDSII data [125], but due to the efforts for the reverse engineering process this method can also only be applied to a limited number of devices. So if not all target ASICs are manipulated, the detection will most probably fail.

- ML based Trojan detection seems more reasonable than other approaches, but it still has significant shortcomings. It seems more or less plausible that ML can detect HW Trojans if there is a difference in the behaviour of ICs with and without Trojan, assuming the approach is capable to cope with process variations etc. But this also means that there needs to be an IC without Trojan or that the IC works for a reasonably long time without Trojan activity to ensure proper training data. It is still extremely challenging if not impossible to detect Trojans that are part of the design from the very first moment on e.g. because they are already implemented in IP blocks imported into the design.

- Prevention techniques may help to reduce the risk of HW Trojans to be inserted but they are requiring the hardware designer to do a tedious task under even harder constraints. The latter may lead to unwanted side effects such as prolonged design time and eventually even more faulty designs.

- Split manufacturing comes with serious demands when selecting the appropriate foundries to ensure compatibility with respect to PDK, wafer size and materials, but it is the most promising choice to reduce the probability of HW Trojans being inserted during the manufacturing process.

To summarize despite more than a decade of intensive research it is still an open issue how to reliably detect a HW Trojan. This is due to the fact that different types of Trojans might be implanted, the sizes of potential Trojans may vary significantly and that also the activation of a potential Trojan is unknown.

### 4.4 Implanting a Hardware Trojan into a cryptographic design

For the Hardware Trojan implanting experiment planned in this project we decided to implement a Trojan in an IHP hardware accelerator for elliptic curve (EC) point multiplication, shortly denoted as $kP$ operation. This is the main and most often attacked operation in EC cryptographic protocols. The goal of attackers is to reveal the (long binary) number $k$, for example the key.

The $kP$ algorithm can be observed as a sequence of special mathematical operations, i.e. additions, multiplications and squaring operations of elements of a finite field. Intermediate and end results have to be stored in registers.

Thus, the design consists of the blocks implementing the required mathematical functions and registers, as well as the block Controller which manages the sequence of the mathematical and storing operations. More
detailed, the Controller manages the data flow between the design blocks and defines which operation has to be performed in the current clock cycle. Due to this functionality of the block Controller, it is easy to extend it with the malicious functionality as follows:

Once per a given number of $kP$ operations – in our case, for each 3rd of 16 executions – the Controller commands to give out the secret value $k$ to a certain pin of the chip.

### 4.4.1 Trojan implementation: Performed Controller modifications

The IHP design, as all binary $kP$ designs, implements a sequence of operations, which depends on the currently processed bit value of the scalar $k$, i.e. the design has some lines of code like this:

```plaintext
if currently_processed_bit_of_scalar_k=1 do
  operation N1
  operation N2
  ...
  operation Nn
if currently_processed_bit_of_scalar_k=0 do
  operation N1'
  operation N2'
  ...
  operation Nn'
```

Thus, to implement the Trojan the condition about the state of the Trojan’s counter and the command to output the bit value = 1 has to be inserted in the sequence of operations N1, N2, ...Nn and the command to output the bit value = 0 into the sequence of operations N1', N2', ...Nn', respectively.

The IHP design consists of 4 such conditional pieces of code in the main loop iteration of the Montgomery $kP$ algorithm, each of them was extended with the same malicious functionality. Figure 4.1 shows the selected parts of the original and all maliciously modified Controller’s code, for the comparison. The code lines describing the Trojan are shown with green background.

The original IHP file describing the functionality of the block Controller consists of 1,183 lines of code including empty lines and lines containing only comments; the number of characters (excluding the space character) is 14,172 corresponding to the Microsoft Word statistics (see Appendix 7). For the description of the Trojan functionality only 16 lines of code were inserted in the original design. Some additional lines were used for the comments. Thus, including the comments, the maliciously modified file consists of 14,573 characters (no space) and 1,220 lines, in other words the manipulations led to an increase of the original design of about 3 per cent only. Figure 4.1 shows a part of the original and maliciously modified Controller’s code as an example: here the additional signal is defined, which will be used for outputting the value $k$. A complete comparison of the original code and the code with the Trojan is shown in Appendix 7.

![Figure 4.1](image)

Figure 4.1: Selected part of the original (left side) and maliciously modified (right side) Controller’s code, for the comparison. The code lines describing the Trojan are highlighted with a green background. For a complete comparison, please see Appendix 7.
4.4.2 Designs ported to FPGA Spartan-7

In our experiment we used the Xilinx Vivado v2018.3 Software suite (64-bit, SW Build: 2405991, IP Build 2404404). We ported the kP designs with and without the Trojan to a Cmod S7 board with a Xilinx Spartan-7 FPGA [112]. For the synthesis and implementation of both designs we used default settings, i.e. “Vivado Synthesis Defaults” and “Vivado Implementation Defaults” strategies, respectively. The design was synthesised for a target frequency of 100 MHz.

Figure 4.2 visualizes the FPGA resource utilisation for both kP designs. The maps were obtained using the Vivado software. The original design is shown on the left side and the design with the Trojan is shown on the right side of Figure 4.2. The Controller block is highlighted with yellow colour in both designs. The rest of the design blocks are represented in cyan.

As it is clearly shown in Figure 4.2, the placements of the both designs differ significantly, despite the small difference in the designs’ codes and completely the same settings applied for the designs’ synthesis. Moreover, the design with the Trojan is more compact, i.e. it occupies less LUTs and Slices in comparison to the original design:

- Original design: 5834 LUTs, 1904 Slices
- Design with the Trojan: 5461 LUTs and 1717 Slices

More details of the FPGA resource utilisation are given in Appendix 7. The additional difference is that the design with the Trojan has one pin more: 7 “bonded IOB” (input/output block) by the original design and 8 “bonded IOB” by the design with the Trojan. Detailed utilisation report data for both designs is given in the Appendix. Due to the fact that the design with the Trojan requires less resources, it can be (maliciously) presented as an optimised original design.

4.4.3 Investigation of the influence of the implanted Trojan on the design resistance to SCA

With the goal to demonstrate the functionality of the Trojan as well as to investigate if and how strong...
implanting the Trojan influences the energy consumption as well as the resistance of the cryptographic design to side-channel analysis (SCA) attacks, we performed measurements of the electromagnetic radiation of the FPGA closely to the chip while it was executing $kP$ operations, i.e. we measured electromagnetic traces and performed an analysis.

We decided to measure the electromagnetic traces (EMTs), due to the fact that such measurements do not require any changes of the attacked board and can be performed on the power supply capacitors which should be always available on the board for ensuring correct functionality of the chip.

### 4.4.3.1 Measurement place selection

Analysing the schematic of the attacked board [113] we selected the capacitor C45 as the best suitable for our measurements: it is a relatively small capacitor (47 nF) supplying the power of the FPGA core. Figure 4.3 shows a part of the schematic of the attacked FPGA board with the power supply capacitor selected for the measurements.

![Figure 4.3](image)

*Figure 4.3: A part of the attacked FPGA board schematic with the power supply capacitor selected for the measurements.*

Figure 4.4 shows the selected capacitor on the board as well as the magnet field probe placed close to the capacitor. For the measurements we used a Langer MFA-R 0.2-75 [114], which has a spatial resolution of 300µm. The electromagnetic traces were captured using a Teledyne Lecroy WavePro 604HD oscilloscope [115] applying a sampling rate of 10 GS/s. This results into 100 measured samples per a clock cycle at a clock signal frequency of 100 MHz. The placement and orientation of the magnet field probe was not changed during all measurements reported in this section.
Figure 4.4: Measurement of electromagnetic trace of kP executions on the attacked board: placement of the magnet field probe on the measurement place. The capacitor C45 has dimensions of 1 mm x 2 mm.
4.4.3.2 Measurement and analysis results

In our experiments we used following input values (given here as hexadecimal numbers) for the kP designs:

Elliptic curve points of the EC B-233:

P1=(x,y) with
\[ x=181856adc1e7df1378491fa736f2d02e8acff1b9425eb2b061ff0e9e8246 \]
\[ y=89fed47b796480499cbaa86d8eb39457c49d5bf345a0757e46e2582de6 \]

P2=(x,y) with
\[ x=46302be2f8b86c004dd16d925d09d4bc0cda66ebd01765a8148562c2 \]
\[ y=1c3c4e77a7dfb4f12ca0923f83df6e6f12e4f6c6d9dd194dec2d70b2 \]

232 bit long scalar k1= 9391925f4359f4c2be67dea456ef70a545a9c44d467f409f96cb52cc
233 bit long scalar k2=1b0c0d3f112c6ebcf541520a70b9824d7be5524599d7b4062c3a6f14ef

EMT were measured for the following kP executions:

- For the original design:
  - k1P1 (denoted as o_k1P1)
  - k1P2 (denoted as o_k1P2)
  - k2P1 (denoted as o_k2P1)
  - k2P2 (denoted as o_k2P2)

- For the design with the Trojan:
  - k1P1 with the Trojan counter state =1 (denoted as t_k1P1_1)
  - k1P1 with the Trojan counter state =2 (denoted as t_k1P1_2)
  - k1P1 with the Trojan counter state =3, i.e. with the active Trojan (denoted as t_k1P1_3)
  - k1P1 with the Trojan counter state =4 (denoted as t_k1P1_4)
  - k1P2 with the Trojan counter state =1 (denoted as t_k1P2_1)
  - k1P2 with the Trojan counter state =2 (denoted as t_k1P2_2)
  - k1P2 with the Trojan counter state =3, i.e. with the active Trojan (denoted as t_k1P2_3)
  - k1P2 with the Trojan counter state =4 (denoted as t_k1P2_4)

Figure 4.5 shows the EMT denoted as t_k1P1_1 (see green line), i.e. it is a trace captured from the malicious design but the value k1 of the processed scalar was not given as the output value. The yellow line in Figure 4.5 is the signal from the FPGA pin, which the Trojan uses for the output of the scalar.

![Figure 4.5: EMT of the kP design with Trojan. The Trojan is not active, i.e. it does not output the processed scalar to the reserved pin (trace name: t_k1P1_1).](image)

Figure 4.6 shows the EMT denoted as t_k1P1_3 (see green line) and the value of the processed scalar k1 as the output signal (see yellow line) on the reserved FPGA pin. Please note that the Trojan is inserted into the
main loop of the $kP$ algorithm, and the attacked IHP design processes the two most significant bits before the main loop iterations. Thus, for the scalar $k1=9391255fd_{16}=100100111001001100101010111111101..._2$ the output of the Trojan is $01001110010001100100100101010111111101..._2$ (‘1’ is the high signal level about 3.3 V, and ‘0’ is the low signal level about 0V).

Figure 4.6: EMT $t_{k1P1_3}$ (see green line) of the $kP$ design with the active Trojan: the Trojan gives out the value of the processed scalar to the reserved FPGA pin (see yellow line).

The original $kP$ design and the design with the Trojan require about 13000 clock cycles for the processing of the scalar, i.e. a $kP$ duration is about 130 µs at a frequency of 100 MHz (see Figure 4.5 and Figure 4.6). A single bit of the scalar $k$ is processed for 54 clock cycles in the main loop iteration, i.e. the processing of a single bit is represented with 5400 measured samples in our experiments.

We synchronised all 12 measured traces with the goal to visualize the differences in the traces. Figure 4.7 shows a part for each of the four EMTs of different $kP$ executions measured on the original design.

Figure 4.7: A part of all 4 EMTs of different $kP$ executions measured on the original design.

It is clearly seen that not only the processed scalar, i.e. the value $k1$ or $k2$, but also the EC point – $P1$ or $P2$ – influence the shape of the measured signal significantly. Despite of this data-dependent influence the resistance of the design to an automated simple analysis attack [116] is quite the same. We evaluated the resistance of the design to the attack performed as success rate of the scalar extraction, calculating the
relative correctness for each key candidate. For example, if 200 of 230 bits of a key candidate are identical to the corresponding values of the processed scalar, the relative correctness is 200/230*100%. Thus, if all bits of a key candidate are identical to the corresponding bits of the processed scalar, the relative correctness of the key candidate is 100%, i.e. the scalar is completely revealed and the attack is successful. A correctness of 0% means that all bits of the key candidates are not correct, but in this case the inverted key candidate is identical to the processed scalar. From the attacker’s point of view the relative correctness of 50% is the worst possible. Figure 4.8 represents the relative correctness, i.e. the number of key bits correctly revealed in per cent, for all four EM traces measured for the original kP design and each of the 5400 key candidates per kP trace. We used the same colour code as in Figure 4.7.

Figure 4.8: Graphic representation of the success rate of the automated simple analysis attack: all 4 EMTs of the original design were analysed.

Figure 4.9 shows a part of the EMTs measured on the design with the Trojan. The EMT of the same kP operation executing on the original design is given for the comparison. All "Trojan" traces processing the same inputs are quite identical to each other, i.e. the trace with the active Trojan cannot be easily distinguished from such a set of traces. The trace of the original design processing the same inputs as in the "Trojan" traces differs slightly from the set of the "Trojan" traces (see green line in Figure 4.9 which is slightly higher than all other traces).

Figure 4.10 represents graphically the success of the automated simple analysis attack for all 12 measured EMTs. This graph demonstrates the fact that the vulnerability to automated simple SCA is similar for the both designs, i.e. for the design with and without the Trojan, thus the main SCA leakage sources are the same. The additional vulnerability of the design with the active Trojan cannot be identified using the applied SCA.

An interesting fact is demonstrated in Figure 4.11. A comparison of the resistance obtained analysing the trace of the original design and of both Trojan design traces shows that the original design has stronger SCA leakage sources than the design with the Trojan. Please note that the design with the Trojan can be (maliciously) represented even as a re-designed original design with slightly increased resistance against performed SCA. The better resistance of the "Trojan"-design can be caused by the different – more compact than in the original design – placement of the design blocks. It slightly reduces signal time delays and power consumption by keeping the noise on the same level, i.e. slightly decreasing signal/noise ratio.

Please note that the Trojan described in this work can be hidden using a big counter, for example up to 1000, i.e. the key will appear on a certain pin only once per 1000 kP operations. Due to the high number of cryptographic operations to be performed nowadays, it is not a difficult task for an attacker to measure 1000 traces. However, the probability to detect the Trojan’s activity during a test of the malicious design by a test engineer is low, especially due to the fact that only specified design activity will be tested. Many other kinds of Trojans revealing the key can be developed, whereby the result of the Trojan’s activity – the output of the key value – can be hidden in the measured trace itself. For example, a Trojan can give out not the whole key value but only one single bit during each new kP operation, i.e. if the key is 256 bit long the attacker has to measure 256 traces. Such a Trojan contains a small counter (up to 256) and can reveal each key bit value ‘1’
via additional activation of a design block on a certain clock cycle at the end of $kP$ execution causing additional energy consumption\textsuperscript{9}, etc.

Figure 4.9: A part of all EMTs of different $kP$ executions measured on the design with the Trojan; the EMT of the same $kP$ operation executing on the original design is given for the comparison: all “Trojan” traces are quite identical to each other, i.e. the trace with the active Trojan cannot be distinguished from the set. The trace of the original design processing the same inputs as in the “Trojan” traces differs slightly from the set of the “Trojan” traces.

Figure 4.10: Graphic representation of the success rate of the automated simple analysis attack: all 12 measured EMTs were analysed. This graph demonstrates the fact that the vulnerability to automated simple SCA is similar for the both designs, i.e. for the design with and without the Trojan. The additional vulnerability of the design with active Trojan cannot be identified using the applied SCA.

\textsuperscript{9} Please note that such a kind of a “covert-channel” HW Trojans can be detected by comparing the traces of the manipulated design with the measured (or at least the simulated) traces of the “golden device” if available.
4.5 PCB-Experiments

The aim of the experiments is to provide an experimental assessment of the attack reported in Bloomberg [1]. So, the task performed was implanting a (small) chip into a PCB in order to assess the feasibility, cost and time of such manipulations as well as the conspicuousness of the manipulated areas. There are many possibilities to manipulate a PCB e.g. by hiding an IC within the layers of the PCB (cf. Chapter 1) or to use an IC that looks like a regular component, for example a capacitor, or to place the malicious IC under another component. In this experiment we implanted a small chip in the compound of a certain coil as well as under the housing of a second coil.

A large PCB (a non-functioning motherboard from a Dell Latitude E6410 laptop) was chosen for the experiments as the board into which the IC had to be implemented, see Figure 4.12.

Figure 4.12: Dell Latitude E6410 motherboard, front and back side.
The following 4 ICs have been considered as ICs to be implanted. (see Figure 4.13):

![ICs Images]

---

**Figure 4.13: ICs selected for experiments, i.e. considered as ICs to be implanted:**

a) NXP A1006UK/TA1NXZ secure authenticator  
b) NXP A1006TL/TA1NXZ secure authenticator  
c) Infineon OPTIGA™ Trust B Authentication IC (SLE952500000XTSA1) Authentication solution for improved security and reduced system costs  
d) An IHP Chip (due to the IHP IP properties the chip die is shown pixeled)

---

In the following we describe the steps taken to implant a die into the selected PCB.  
**Step 1:** X-raying the ICs. Figure 4.14 shows optical pictures as well as X-Ray pictures of all 4 chips.

![X-Ray Images]

---

**Figure 4.14: Optical and X-Ray pictures from all 4 ICs selected for experiments.**
Step 2: Identifying suitable places for manipulation (normal visual inspection). Suitable places are e.g. the places under larger components or glob top potting material, etc. In addition to give a better insight the board was also examined in the X-Ray machine: very dark areas are those with a lot of metal or materials with a high density, e.g. coils or capacitors. Places for the manipulation are then those that remain inconspicuous in both, i.e. normal optical inspection and X-Ray inspection. Two locations that were selected based on the inspections are Coil 1 and Coil 2 marked by yellow circles in Figure 4.15.

![Figure 4.15: Latitude Mother Board with locations to implant ICs.](image)

Coil 1 is covered with potting compound, see more details in Figure 4.16. Coil 2 is shown detailed in Figure 4.17.

![Figure 4.16: Optical and X-Ray images of position 1 (Coil 1) for implanting an IC.](image)
Step 3: Testing which chips might go undetected when X-raying the manipulated PCB. All 4 chips were placed on the PCB and X-Ray pictures were taken. All chips except for the IHP chip are visible, see Figure 4.18 - Figure 4.21.

Figure 4.17: Optical and X-Ray images of position 2 (Coil 2) for implanting an IC.

Figure 4.18: X-Ray image of the PCB and the NXP A1006UK/TA1NXZ.
In Figure 4.21 the IHP chips are almost invisible. The situation even improves from an attacker’s point of view once the optical noise level increases. The NXP A1006TL/TA1NXZ chip (see Figure 4.14-b and Figure 4.19) was excluded from further experiments: it has internal gold wires that are clearly visible in the X-Ray images.
Additional experiments were run with IHP chips to determine whether the aluminium bonding wires would show up in the X-Ray image, please see Figure A8.1 in Appendix 8 for the corresponding X-Ray images. The IHP chip with Aluminium wires was placed on the board and X-rayed, see Figure 4.22: The chip outlines are visible while the wires are not. The chip would also disappear in the noise if placed in the right spot.

Based on the experiments run so far we assume that the IHP chip becomes invisible in certain places when implanted, while the other chips will be easier to find.

In the following we describe the experiment for each of the coils in a separate subsection.
4.5.1 Experiments with Coil 1: Hiding chips in the compound

In the following figures we are displaying the steps taken to implement the IHP chip into the compound material of Coil 1 in the order of their execution, see Figure 4.23 a) – d). A direct comparison of optical and X-Ray images between the original and the manipulated coil is shown in Figure 4.24.

Figure 4.23: Steps to hide the IHP chip in Coil 1.
In our second experiment with Coil 1 we implanted the Infineon OPTIGA™ Trust B Authentication IC (SLE95250000XTSA1) using the same steps as for implanting the IHP chip (see Figure A8.2 in Appendix 8). Figure 4.25 shows X-raying images of the PCB after both manipulations. The Infineon chip can be easily seen and was marked with a red rectangle.

Figure 4.24: Verification of the manipulation and its detectability using optical and X-Ray images.
Figure 4.25: X-Ray images of the PCB after implanting the IHP and the Infineon chip, see the image in the middle. The yellow rectangles denote the area manipulated; the areas are shown zoomed in in the top and bottom images, respectively. The IHP chip is not visible. The Infineon chip is visible, marked by the red rectangle in the zoomed in image.
4.5.2 Experiments with Coil 2: Hiding chips under the housing

4.5.2.1 Implanting the IHP chip

In this experiment the approach differs a little from the one described before. We now try to hide the implanted chip under the metal housing of the coil. In Figure 4.26 – Figure 4.27 we are displaying the steps that were taken to implant the IHP chip into Coil 2 in the order of their execution. A direct comparison between the original and the manipulated coil is shown in Figure 4.27.

**Step 1:** Unsoldering Coil 2, place the IHP chip next to the solder tag and glue it on, see Figure 4.26.

![Figure 4.26: Coil 2 after unsoldering and with the IHP chip; its front and side view.](image)

**Step 2:** Coil 2 soldered again on the PCB, see Figure 4.27

![Figure 4.27: Picture of the PCB before and after un- and resoldering Coil 2. Please note that the visible difference in the solder can be easily obfuscated by an attacker, e.g. be resoldering all three coils or by artificially aging the soldering points.](image)

**Step 3:** Verification of the stealthyness of the manipulation using X-raying images, which revealed that the IHP chip is non-detectable by optical inspection, see Figure 4.28.
4.5.2.2 Implanting the NXP chip

Here we first describe the steps taken to manipulate Coil 2 and afterwards we display these steps in .

Step 1: Unsoldering Coil 2 and place the NXP A1006UK /TA1NXZ secure authenticator chip on the non-manipulated side of the coil.

Step 2: Milling the coil a little to be able to implant the 0.4 mm thick chip, i.e. avoid the manipulated coil to be higher than original.

Step 3: Soldering the coil back on the PCB.

In Figure 4.29 we are displaying the step 2 of the implanting of the NXP chip into Coil 2.
Figure 4.29: Coil 2 with the NXP A1006UK/TA1NXZ on top of it (upper images); Coil 2 after milling with the NXP A1006UK/TA1NXZ placed in the milling hole.

Figure 4.30 and Figure 4.31 show X-Ray images of the soldered Coil 2 after both manipulations. The IHP chip is not visible, but the NXP A1006UK/TA1NXZ chip is easily to be recognised (see red rectangles).
Figure 4.30: X-Ray image of Coil 2 at the position at which the IHP chip was implanted.

Figure 4.31: X-Ray image of Coil 2 (upper image) and Coil 2 at the position at which the NXP A1006UK/TA1NXZ chip was implanted, zoomed in (lower part).
4.5.3 Cost Issues

In order to assess the feasibility of an attack and to some extent also its probability the cost in terms of time, skills and equipment are essential.

The cost of the equipment used to execute the experiments is:

- Microscope Leica S9i [117], 7 000 Euro
- X-Ray Inspection-Device Nikon XT V 160 [118], 180 000 Euro
- Bonder F&S Bondtec 5632 (Wedge-Wedge, Al-Wires) [119], 140 000 Euro
- Dremel with milling head, 500 Euro
- Soldering iron and tweezers, 1000 Euro

So in sum about 330,000 Euro (more details about the applied equipment can be found in Table A8.1 in Appendix 8). Please note that the most expensive device is the X-Ray Inspection Device which is mainly needed to select the most appropriate chip for the manipulation and the verification and that it might be feasible to rent an X-raying service. In addition, this equipment is available to attackers due to their normal work e.g. equipping PCBs.

The following time was needed for the experiments (without documenting them, taking pictures, etc.):

- Milling/desoldering/soldering a coil: about 10 min.
- Placing the chip: up to 1 hour (the smaller and thinner, the more problematic)
- Close Coil 1 with potting compound: the time it takes for the potting compound to dry is crucial here and can take up to 24 hours.
- Bonding wires: depends on the chip, usually up to 1 hour.

So, the time needed for the manipulations is about 27 hours, while the majority of the time is needed for drying the compound if the task is clearly defined, i.e. if the chip and its placement are determined for the technical support staff. Please note that the manipulation will take longer especially for determining the positions where to integrate the chip(s) depending on their malicious tasks. Also connecting the chips with other components will take a significant amount of time and thoughts. But as can be deduced from the time the experiments took it is clear that the time to run such an attack is for sure not prohibitive. If the PCB design house is involved in the placement and routing process, this effort also decreases significantly.

So based on the experience we gained during these experiments, we consider an attack that manipulates a PCB – like the one discussed in the Bloomberg report without commenting on the correctness of the report – as plausible.
5 Conclusion

In this project we analysed the different steps in the development of a complex IT system covering all steps from the initial idea via design and manufacturing of an ASIC to the design and production of the PCB with respect to the probability of a successful attack. This probability depends on the knowledge and capabilities of a certain attacker and the effort such an attacker has. In order to determine this information, we gathered data from a thorough literature review, expert interviews and own experiments.

The result is that in each of the development stages successful attacks are possible. In which of the stages an attacker targets the design under development highly depends on the intentions of the attacker. The more targeted the attack shall be the higher is the probability that a later stage will be attacked. In these stages a potential victim can be defined more exactly. If an attacker is willing to spend some effort in order to “reach” many victims without a clear picture who they are, earlier development stages are more likely to be attacked.

The analysis of the literature as well as the expert interviews clearly showed that:

- Means to detect hardware Trojans are not easy to apply, as they require normally a “golden” device or at least a “golden” simulation which usually are not available except in scientific settings. The fact that it is normally unclear how to activate the Trojan makes its detection even harder. So, relying on hardware Trojan detection alone seems not to be sufficient. Nevertheless, for selected cases with a limited scope (e.g. trust anchor chips), Trojan detection methods can give at least a hint about potential manipulations. Therefore, it is important to invest in and have advanced chip analysis and detection capabilities available.

- Means to prevent hardware Trojans are limited. On the one hand such means are not easy to apply and will lead to significant overhead in terms of area and power consumption. On the other hand they cannot be applied to the whole design so an attack may still be capable to integrate a hardware Trojan in the non-protected parts.

- The most promising approach is split manufacturing as it limits the information available to a manufacturer about a certain design significantly. But the process of organising an appropriate manufacturing process including finding compatible manufacturers that are willing to support such a process is very complex.

Our experiments showed that a hardware Trojan can be integrated with rather limited effort and confirmed that such a Trojan can hardly be detected. The additional area and energy consumption are negligible. Also the integration of an additional IC in a complex motherboard was researched. The effort to accomplish this is rather negligible. Also in this case, the experiments showed that detecting the implemented chip is extremely difficult. Even x-raying the PCB does not guarantee to detect such a manipulation.

So, the main take home messages are:

- Attacks such as the manipulation of a motherboard as described by Bloomberg in 2018 [1] are feasible and most probably will go undetected. However, whether Bloomberg’s claim is correct or not remains open.

- The insertion of hardware Trojans is also feasible and it is very difficult to detect such a manipulation. It is especially difficult if the manipulation is integrated in an open hardware core, or if the design house is malicious. This means that purchasing equipment from a manufacturer that is not considered fully trustworthy always comes with a certain risk, as it will never be 100% assured that there is no malicious manipulation present.

- To limit scope and make this analysis feasible in terms of effort, we excluded insider attacks in our analysis. Yet, they need to be considered carefully. Bribing or blackmailing employees of a design house is a cost efficient way to get the expertise, knowledge and access to a certain design to mount a stealth attack.

66
Based on the results of this project our main recommendations from a business administration point of view are:

- **Trustworthy personnel**: Invest in personnel to set-up a trust relationship and avoid that employees are turned into malicious agents of an opponent.

- **Trustworthy IT infrastructure**: In case a corporate network gets hacked, an external attacker has the same means to manipulate designs. State of the art IDS, appropriate site-audits etc. are a must, and must in particular take this attack vector into consideration.

- **Trustworthy suppliers**: Trusted relationships with suppliers, ideally along the complete supply chain, are the best means to avoid stealthy manipulations.

- **Use Reliable suppliers** - even if it means higher cost - instead of subcontracting new partners every now and then.

But the results of this project are also of importance from a national economical point of view. The fact that attacks may be successful in different development steps means that:

- **Own Production Facilities**: Providing own production facilities does not necessarily prevent manipulation of products; nevertheless it reduces certain risks and political dependencies while increasing availability in case of a global supply shortage. Ensuring design capabilities along the complete supply chain is essential to detect and prevent manipulations, despite this is not sufficient.

- **Advanced Detection Capabilities**: Detection of manipulations can only be done on an individual case-by-case scenario for high-value circuits, e.g. trust anchors. Invest in advanced detection capabilities in order to handle such cases, to deter attackers in the first place, and in order to increase the cost and effort required for an attacker as trivial manipulations do not go undetected.

- **R&D in the field of detecting manipulations**: Several approaches seem to be promising and may increase the abilities of a user to detect manipulations significantly.

In any case the complete supply chain needs a thorough and individual assessment for all sensitive developments or products.
Bibliography


Bibliography


List of abbreviations

ASIC – Application-Specific Integrated Circuit
BIOS – Basic Input/Output System
BOM – Bill Of Materials
CAD – Computer-Aided Design
CAM – Computer Aided Manufacturing
CAN – Controller Area Network
CNC – Computer Numerical Control
DRC – Design Rule Checking
DSP – Digital Signal Processor
DUV – Deep Ultraviolet
DXF – Drawing eXchange Format
EC – Elliptic Curve
EDA – Electronic Design Automation
EMD – Engineering and Manufacturing Development
EMT – ElectroMagnetic Trace
ENISA – European Union Agency for Cybersecurity
ERC – Electrical Rule Checking
EUV – Extreme Ultraviolet
FI – Fault Injection
FIB – Focused Ion Beam
FPGA – Field-Programmable Gate Arrays
GDS – Graphic Database System
HW – Hardware
HDL – Hardware Description Language
IC – Integrated Circuit
I/O – Input/Output
IP – Intellectual Property
IT – Information Technology
LVS – Layout vs. Schematic, it is a logical equivalence checks
MSA – Material Solution Analysis
NIST – National Institute of Standards and Technology (U.S.)
O&S – Operations and Support
PCB – Printed Circuit Boards
PCI – Peripheral Component Interconnect
List of abbreviations

PDK – Process Design Kit
PKS – Physically Knowledgeable Synthesis
P&D – Production and Deployment
RTL – Register Transfer Level
SCA – Side-Channel Analysis
SPI – Serial Peripheral Interface
SoC – System on Chip
SW - Software
SiP – System in Package
TARA – Threat Assessment and Remediation Analysis
TD – Technology Development
UART – Universal Asynchronous Receiver-Transmitter
VHDL – Very high-speed integrated circuit Hardware Description Language
WP – Work Packages
Appendix 1: Examples of selecting IP Cores (IHP experience)

IHP provides some IP cores, for example an SPI Slave that is a communication interface. In different projects IHP applied the selected IP cores. Here, we give some examples how the IP cores were selected by IHP colleagues. Tables Table A1.1 - Table A1.4 represent shortly some interviews made at IHP for this topic to give an impression which the selection criteria were used for selecting IP cores.

In the last 15 years IHP developed about 50 different variants of SPI slaves for different versions of the IHP technology. There are implementations available in 0.25µm, 0.13µm and 0.13µm Cu technology. The majority was developed for internal customers but there also 10-15 external customers.

Table A1.1: SPI slave developed in IHP for different IHP technologies.

<table>
<thead>
<tr>
<th>Interview question</th>
<th>Answers</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP-block</td>
<td>Communication interface SPI Slave</td>
</tr>
<tr>
<td>Short description</td>
<td>Serial Peripheral Interface (SPI) is a synchronous serial communication interface specification used for short-distance communication, primarily in embedded systems. The interface was developed by Motorola in the mid-1980s and has become a de-facto standard. Typical applications include Secure Digital cards and liquid crystal displays.</td>
</tr>
<tr>
<td>Is it a soft or a hard core? For which target developed/planned to apply (for ASIC directly? for an FPGA? or developed for an FPGA but planned to use for an ASIC?)</td>
<td>ASIC, FPGA, VHDL core + digital CMOS hard core</td>
</tr>
<tr>
<td></td>
<td>Prepared: a project member</td>
</tr>
<tr>
<td></td>
<td>Verified: Project leader</td>
</tr>
<tr>
<td></td>
<td>Approved: Department Head</td>
</tr>
<tr>
<td></td>
<td>Released: Project leader MPW</td>
</tr>
<tr>
<td></td>
<td>The VHDL design implements an SPI slave, which is configurable in VHDL source code. It provides a configuration dependent number of output registers, which can be written via the SPI bus. Some configurations provide also inputs, which can be read via the SPI bus. Some versions have been synthesised and layouted for IHP’s 0.13 µm and 0.25 µm CMOS technologies.</td>
</tr>
</tbody>
</table>

The following tables show the experience from colleagues concerning the information used to select specific IP cores. These tables cover different types of IP cores.
### Appendix 1: Examples of selecting IP Cores

#### Table A1.2: Commercial IP-block: CAN controller applied at IHP.

<table>
<thead>
<tr>
<th>Interview question</th>
<th>Answers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IP-block</strong></td>
<td>CAN controller IP</td>
</tr>
<tr>
<td>What exactly was done during the selection</td>
<td>Looking into various IP providers by internet platforms, looking into IP SoC service</td>
</tr>
<tr>
<td>Actors/details/explanations</td>
<td>The project team was involved in selection, under the lead of project leader. Specification for an IP have been defined (standard compatibility, deliveries, availability etc.)</td>
</tr>
<tr>
<td>Is it a soft or a hard core? For which target platform was it developed/or which target platform shall it be used as ASIC or FPGA</td>
<td>VHDL soft core, for ASIC core. Project was defined already as ASIC project. The expected core should be platform independent, i.e. mapping for ASIC and FPGA should be possible</td>
</tr>
<tr>
<td>Why the IP is necessary?</td>
<td>This was a part of high level specifications of the IC, as requested from the customer</td>
</tr>
<tr>
<td>Is it planned as a part of an IHP-Chip/SoC?</td>
<td>This was part of industry development with the partner who defined the specs</td>
</tr>
<tr>
<td>Why was the IP-block selected? (criteria)</td>
<td>Functional compatibility (very important), according to available IEEE standard. External IO compatibility irrelevant since the CAN has specific external interfaces and internally was required that some bus interface is available which could be used in the system. The mapping of the IP should be such that easy software integration is possible. RTL model required, cycle to cycle verifiability is available.</td>
</tr>
<tr>
<td>- Function compatibility</td>
<td>We have required full documentation package</td>
</tr>
<tr>
<td>- I/O compatibility</td>
<td>Testbenches need to be available (standard test suite preferable, but was not mandatory)</td>
</tr>
<tr>
<td>- Software compatibility</td>
<td>Production ready IP</td>
</tr>
<tr>
<td>- Cycle-to-cycle compatibility</td>
<td>IP Vendor important question: preferable EU companies due to export restrictions (ITAR), prior experience with IP provider preferable. Also the complete IP portfolio has been evaluated.</td>
</tr>
<tr>
<td>- Information available about the IP</td>
<td>We needed unlimited commercial license, however for the price reasons limiting to single IC license was possible</td>
</tr>
<tr>
<td>- Testability of the functionality</td>
<td>Code change was not required</td>
</tr>
<tr>
<td>- IP bugs</td>
<td>Testbench package required.</td>
</tr>
<tr>
<td>- Evaluation of IP vendors: what was done?</td>
<td>Additional (not-required) functionality: is it a part of the selected IP? Was its correctness verified/tested?</td>
</tr>
<tr>
<td>- Testability (after manufacturing as (a part) of an ASIC)</td>
<td>Not relevant</td>
</tr>
<tr>
<td>- Portability and reusability</td>
<td>IP was integrated in the complete IC and testbenches have been integrated in the overall verification environment.</td>
</tr>
</tbody>
</table>
Appendix 1: Examples of selecting IP Cores

Table A1.3: Open source IP-block: PULPissimo microcontroller applied at IHP.

<table>
<thead>
<tr>
<th>Interview question</th>
<th>Answers</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP-block</td>
<td>an open source microcontroller PULPissimo (<a href="https://github.com/pulp-platform/pulpissimo">https://github.com/pulp-platform/pulpissimo</a>) PULPissimo is the microcontroller architecture of the more recent PULP chips, part of the ongoing “PULP platform” collaboration between ETH Zurich and the University of Bologna - started in 2013. PULPissimo is a single-core platform. It is used as the main System-on-Chip controller for all recent multi-core PULP chips, taking care of autonomous I/O, advanced data pre-processing, external interrupts, etc.</td>
</tr>
<tr>
<td>What exactly was done during the selection</td>
<td></td>
</tr>
<tr>
<td>Actors/details/explanations</td>
<td>IHP project Scale4Edge team members</td>
</tr>
<tr>
<td>Is it a soft or a hard core? For which target platform was it developed/for which target platform shall it be used as ASIC or FPGA</td>
<td>Soft-IP written in Verilog and SystemC. Synthesised and verified by IHP for FPGA and ASIC.</td>
</tr>
<tr>
<td>Why the IP is necessary?</td>
<td>Up until now we only make use of either commercial cores or our self-developed Core based on the Peakttop ISA. The open source RISCV-ISA and in particular the pulpissimo architecture offer a state of the art alternative with strong community background.</td>
</tr>
<tr>
<td>Is it planned as a part of an IHP-Chip/SoC?</td>
<td>Implemented as part of the TETRISC SoC (Scale4Edge Project)</td>
</tr>
<tr>
<td>Why was the IP-block selected? (criteria)</td>
<td></td>
</tr>
<tr>
<td>- Function compatibility - I/O compatibility - Software compatibility - Cycle-to-cycle compatibility</td>
<td>Comparison between Pulp and Chipyard framework. We already had a good idea of the architectural changes on the design – Pulpissimo was the more suitable option for our fault tolerant approach.</td>
</tr>
<tr>
<td>- Information available about the IP - Testability of the functionality - IP bugs - Evaluation of IP vendors - What was done?</td>
<td>Comparison between Pulp and Chipyard framework: Both Pulp and Chipyard are very well documented and have an active support community. But Pup is written in standard languages, which gave us an easier start.</td>
</tr>
<tr>
<td>- Licence - Portability and reusability - Restrictions (the possibility to change the code, etc.)</td>
<td>We assume a complete verification of the pulpissimo. Only our own parts were additionally verified by our designers.</td>
</tr>
<tr>
<td>Testability (after manufacturing as a part of an ASIC)</td>
<td>We are going to perform parametric and functional tests of the ASIC.</td>
</tr>
<tr>
<td>Additional (not-required) functionality: is it a part of the selected IP? Was its correctness verified/tested?</td>
<td></td>
</tr>
</tbody>
</table>
### Table A1.4: Open source IP-block: VexRiscv microcontroller applied at IHP.

<table>
<thead>
<tr>
<th>Interview question</th>
<th>Answers</th>
</tr>
</thead>
</table>
| IP-block                                     | **RISC-V ISA CPU** implementation  
VexRiscv is an FPGA friendly RISC-V ISA CPU implementation with following features:  
RV32IM instruction set  
Pipelined on 5 stages (Fetch, Decode, Execute, Memory, WriteBack)  
1.44 DMIPS/Mhz when all features are enabled  
Optimised for FPGA |
| What exactly was done during the selection   | Deep investigation of the VexRISCV design, spinal HDL and the tool chain                                                                                                                                  |
| Actors/details/explanations                  | The entire project team of VE-HEP                                                                                                                                                                    |
| Is it soft or hard core? For which target developed/planned to apply ( | Optimised for FPGA, planned for ASIC                                                                                                                                                                |
| Why the IP is necessary?                     | While the other project partners implement different security features, for us it is more interesting to see if the VexRISCV can be manufactured when only open source tools were used during the design and implementation phase. |
| Is it planned as a part of an IHP-Chip/SoC?  | Implemented as a standalone-chip                                                                                                                                                                     |
| Why was the IP-block selected? (criteria)    | - Function compatibility  
- I/O compatibility  
- Software compatibility  
- Cycle-to-cycle compatibility  
- Information available about the IP  
- Testability of the functionality  
- IP bugs  
- Evaluation of IP vendors  
- What was done?  
- Licence  
- Portability and reusability  
- Restrictions (the possibility to change the code, etc.)  
- Testability (after manufacturing as (a part) of an ASIC)  
- Additional (not-required) functionality: is it a part of the selected IP?  
- Was its correctness verified/tested?  

*It was selected because of its unique feature set, the spinal programming language and the design tool flow that was available open source.* |
| Testability (after manufacturing as (a part) of an ASIC) | We are going to perform parametric, scan and functional tests of the ASIC.                                                                                                                               |
| Additional (not-required) functionality: is it a part of the selected IP? | Yes, the VexRiscv contains several security features developed and verified by the project partners |
Appendix 2: Examples of pins and pads

A pad is a contact area on a chip/die or on a PCB for bonding/mounting/attaching components. A pin is a kind of wire, which will be mounted through the pads.

Figure A2.1 shows pins and pads of an IHP chip in a Ceramic Dual In-line package.

Figure A2.1: An IHP chip in a Ceramic Dual In-line package (C-DIP) – a), zoomed-in – b) and c). The chip consists two identical ICs inside a single package, i.e. the chip is an example of a system in package. In Figure c) only one IC is shown, zoomed-in.
# Appendix 3: Overview of MITRE hardware attack scenarios

## Table A3.1: Overview of attack scenarios selected from the MITRE technical report [44].

<table>
<thead>
<tr>
<th>Attack scenario</th>
<th>“what”/”how”:</th>
<th>&quot;who&quot;</th>
<th>“why”</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2</td>
<td>Adversary intercepts hardware from legitimate suppliers en route to contractor/integrator (in order to modify or replace it).</td>
<td>P7 Supply chain distribution personnel (packaging, shipping, receiving, or transfer)</td>
<td>disruption</td>
<td>corruption</td>
</tr>
<tr>
<td>A11</td>
<td>A maliciously altered hardware component is substituted for a tested and approved component.</td>
<td>P7 Component transfer personnel (e.g., shipping, receiving, and transferring) at a lower tier in the supply chain, including transportation companies.</td>
<td>disruption</td>
<td>destruction</td>
</tr>
<tr>
<td>A15</td>
<td>A hardware or firmware component can be intercepted by an adversary while in transit between supplier and acquirer, for the purpose of substitution or manipulation</td>
<td>P7 Any supplier personnel with undue access privileges</td>
<td>disruption</td>
<td>corruption</td>
</tr>
<tr>
<td>A6</td>
<td>A microprocessor (or other chip) with a secret backdoor is substituted for a legitimate hardware component, where the backdoor is in the actual chip itself rather than in the firmware installed on it.</td>
<td>P6 A microelectronics manufacturer deep in the supply chain.</td>
<td>disruption</td>
<td>corruption destruction</td>
</tr>
<tr>
<td>A22</td>
<td>The design and/or fabrication of hardware components is compromised.</td>
<td>P6 Hardware design and manufacture engineers at lower tier in supply chain.</td>
<td>disruption</td>
<td>corruption</td>
</tr>
<tr>
<td>A24</td>
<td>An ASIC for the system being acquired or maintained is designed and produced with malicious functionality built in.</td>
<td>P6 Hardware designer or fabricator at a lower tier in the supply chain: An adversary gains access to the hardware design and development processes within a Defense Microelectronics Activity (DMEA) accredited “trusted supplier” facility.</td>
<td>disruption</td>
<td>corruption destruction</td>
</tr>
<tr>
<td>A10</td>
<td>A counterfeit component is supplied from a lower-tier component supplier to a sub-system developer or integrator, which is then built into the system being acquired or sustained.</td>
<td>P4, P6 A small-company component supplier feeding into the acquisition or sustainment supply chain, with the ability to introduce counterfeit components into the procurement process in such a way that they are not thoroughly tested or otherwise verified for security.</td>
<td>disruption</td>
<td>corruption</td>
</tr>
<tr>
<td>A25</td>
<td>A counterfeit hardware component is implanted in the system being acquired. (This is different from: A counterfeit hardware component can end up in a system being acquired.)</td>
<td>P4, P6 Engineers and technicians at an assembly subcontractor site.</td>
<td>disruption</td>
<td>corruption</td>
</tr>
<tr>
<td>A31</td>
<td>Manipulation of design specifications to produce malicious hardware (e.g., the modification of transistor specifications for an integrated circuit).</td>
<td>P4, P6 Hardware engineers at a lower-tier (with access to design specifications during the hardware manufacturing process) to whom the manufacture of key components has been outsourced.</td>
<td>corruption</td>
<td></td>
</tr>
<tr>
<td>A29</td>
<td>Rogue processes in an integration facility are established in order to clandestinely insert maliciously altered components into the system.</td>
<td>P4, P6, P7 Organisation with the ability to establish deceptive processes: an adversary has an access to critical components as they are being integrated into the acquired system and can insert maliciously altered hardware or firmware into the system.</td>
<td>disruption</td>
<td>corruption</td>
</tr>
<tr>
<td>A5</td>
<td>Malware is embedded in a replacement server motherboard (e.g., in the flash memory) in order to alter server functionality from that intended.</td>
<td>P4, P6, P7 A software-savvy adversary with hardware procurement control deep in the supply chain: An adversary with access to the procurement, maintenance, and/or upgrade control of servers, during the server procurement or hardware update process.</td>
<td>corruption</td>
<td>disclosure</td>
</tr>
<tr>
<td>A9</td>
<td>A maliciously altered hardware component is substituted for a legitimate component during system test and integration.</td>
<td>P3, P4, P6 Test engineers and hardware integrators at a lower tier in the supply chain (with access to system components during system test and evaluation).</td>
<td>disruption</td>
<td>corruption</td>
</tr>
<tr>
<td>A33</td>
<td>A malicious component is substituted for a legitimate component during the packaging and distribution processes.</td>
<td>P6, P7 Technical and non-technical staff at an Original Equipment Manufacturer (OEM) facility (with access to services provided from a manufacturer to a supplier during packaging and distribution).</td>
<td>disruption</td>
<td>corruption destruction</td>
</tr>
<tr>
<td>Attack scenario</td>
<td>“what”/”how”:</td>
<td>“who”</td>
<td>“why”</td>
<td>Note</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------</td>
<td>-------</td>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>A8</td>
<td>A maliciously altered hardware component is substituted for a baseline component at the PDR (preliminary design review) timeframe.</td>
<td>P3, P4, P6</td>
<td>Assembly sub-contractor engineers and technicians.</td>
<td>disruption</td>
</tr>
<tr>
<td>A28</td>
<td>Insertion of maliciously altered hardware components into the grey market</td>
<td>P1, P6</td>
<td>The grey market components intended to be accepted as genuine from a reputable source</td>
<td>disruption; disclosure; destruction</td>
</tr>
<tr>
<td>A34</td>
<td>Malicious hardware is substituted for a legitimate component during lifecycle maintenance</td>
<td>P6, P7</td>
<td>Technical and non-technical stuff at a field support activity</td>
<td>disruption; corruption; destruction</td>
</tr>
<tr>
<td>A23</td>
<td>During sustainment, legitimate faulty hardware or firmware is replaced by hardware into which malicious subcomponents have been placed</td>
<td>P4</td>
<td>Technician with knowledge of and access to systems within the support supply chain</td>
<td>disruption; corruption; destruction</td>
</tr>
<tr>
<td>A36</td>
<td>Manipulation of any of the hardware/software baselines during Acquisition; functional baseline; allocated baseline; product baseline; or the product baseline updates during sustainment.</td>
<td>P1, P2, P4, P6</td>
<td>A “trusted insider”: configuration management personnel.</td>
<td>corruption</td>
</tr>
<tr>
<td>A7</td>
<td>A malicious component is substituted for an approved component by direction from the program office or the prime contractor.</td>
<td>P1, P2</td>
<td>A “trusted insider” positioned to direct program activity (program office or prime contractor engineer)</td>
<td>disruption; corruption; destruction</td>
</tr>
<tr>
<td>A14</td>
<td>Technology or component architecture descriptions are altered.</td>
<td>P1</td>
<td>A “trusted insider” (system development team) with access to documents that include descriptions of advanced technology and/or specific components</td>
<td>disclosure</td>
</tr>
<tr>
<td>A16</td>
<td>Descriptions of system capabilities are misrepresented or altered.</td>
<td>P1</td>
<td>System development team</td>
<td>corruption</td>
</tr>
<tr>
<td>A17</td>
<td>Mission data are altered.</td>
<td>P1</td>
<td>Systems engineers (&quot;trusted&quot; insiders)</td>
<td>corruption</td>
</tr>
<tr>
<td>A30</td>
<td>During the system build process, the system is deliberately misconfigured by the alteration of the build data.</td>
<td>P1, P2, P8</td>
<td>A “trusted insider”: Engineers who are performing the system build and configuration activities</td>
<td>disruption; corruption; destruction</td>
</tr>
<tr>
<td>A37</td>
<td>Corruption of critical operational data by injecting false but believable data into the system during configuration.</td>
<td>P1, P2, P8</td>
<td>A “trusted insider”: Engineers or technicians who are loading operational data during system configuration.</td>
<td>disruption; corruption</td>
</tr>
</tbody>
</table>
Appendix 4: Interview with IHP designers

The following questions (Q.) were discussed:

How difficult/easy/time consuming is it to implant a small circuit with additional functionality compared to the original specification at different design development stages?

- As a concrete example, the circuit: counter to $10^9$ clocks when the value is reached, a concrete output pin is constantly assigned with '1'.
- How costly/probable is it that this additional functionality will be discovered?
- What can be done to hide the additional functionality?
- What other changes that are difficult to detect can be implemented, e.g.:
  - favour physical/remote attacks (e.g. replace the original gates with gates with the same functionality but different energy consumption)?
  - cause interference with the original functionality?
  - Other suggestions?

Q.: What is the time required for the task of implanting the circuit with additional functionality when this task has been officially set:

Phase 1: Specification

- A few hours, mostly organisational work with documentation

Phase 2: Design development, the commissioner is a member of the team who has implemented the original functionality, i.e. has knowledge of the already developed design and access to VHDL code

- 1 week for implementing/implementing/putting changes into the specification + 1 week for adapting the verification environment + 1 week for testing the design with the new functionality. If no layout has been done yet, then go to layout possibly with the delay, otherwise redo layout.
- If the work on the additional functionality is to be hidden, then do not put any changes in the specification, but “hide” the new test environment may take 1 day. If the functionality is not described in the specification, most likely the verification engineer will not test the functionality up to 109 bars, if a smaller number of bars is sufficient for testing the original functionality.

Phase 3: Design development, the commissioner has access to the net list.

- Assumption: it is specifically stated to which pin '1' should be routed when the circuit with additional functionality is ready:

  No reverse engineering is needed for this. The additional counter can be integrated directly into the netlist, i.e. it is implemented separately in VHDL, its netlist is synthesised, then the clock signal is searched for in the original netlist, and the netlists are integrated by hand, i.e. the 'connection' to the 2 pins is established, and then the netlist is sent to the layout. (Time required: 1-3 days if the pins are known, with search - an undefined time. If it is a hierarchical design, this time can also be very short. If it is a “flat” design, especially with renaming the pins it increase the complexity of the task and – consequently - the time to perform this task cannot be estimated).

  - Q.: Does anyone find this additional circuit?
    - It may be that the Logic Equivalent Checking Test can show it. Depends on how attentive the test engineers are.

Phase 4: the commissioner has access to the layout data.

- Restore netlist, then as above, not much longer.
Phase 5: the appointee has access only to GDSII files.

- Under presumption that no access to the design kit - very difficult to imagine, so unrealistic.
- Under presumption that access to libraries is possible:
  Maybe it would be possible to create more space instead of filler cells, or by pushing cells (usually about 20% space is free). Then the counter block can be implemented and then integrated, which also means the wiring has to be done by hand.
  - Q.: Will the hand-laid wiring X be visible in a finished chip, or can it be laid so that the top metal layer is the same as the original?
    - Normally only the power supply rings and the metal fillers are in the top layers. Therefore, the top metal layer remains unchanged. Time required: 1-2 months (very optimistic estimate).
  - Q.: Can the counter also be distributed, i.e. not realised as a monolithic block, so that its parts can be integrated into the original design at different places?
    - Theoretically it is possible, but it is difficult to estimate, it depends very much on the design. Better would be to synthesise the netlist, implant the counter in the netlist, then redo layout, and “replace” the GDSII file.
- In GDSII files, the size of the individual transistors can be manipulated.

Phase 6: the attacker only has access to masks.

- GDSII files can be restored, but additional functionality by manipulating the mask set is difficult to imagine.
- Mask engineer (theoretically) can delete doping from individual transistors in the GDSII files.

Phase 7: Manufacturing process

- A process engineer can manipulate the doping, but this is manipulated in the same way for all transistors on the wafer, by changing the process parameter. It can happen that after about 6 months the chips no longer work.
  - Q.: What other harmful modifications are conceivable?
    - You can look for a layout tool, read the GDSII file and thin the wires (takes about 2 weeks). Alternatively, it would be better to thin VIAs to the power supply or delete some of them completely. This is less noticeable.
    - During design development, for example, the original gates can be replaced by gates with the same functionality but different power consumption.
    - This is not noticed because the functionality does not change. This manipulation affects either reliability or SCA resistance. Individual gates can be replaced by manipulating the size of their transistors (this can also be done only in GDSII files).
Appendix 5: Interview with IHP PCB experts

The task is to influence the quality of the PCB:
- Changing impedances and falsifying measurements/records is easy.
- Leaving moisture in the material, which then swells over time, oxidises, sets electrochemical processes in motion or simply blows up the PCB during assembly.
- Do not press properly, the PCB is more susceptible to moisture and loss of contact between the layers.

Task: implant a chip into a PCB (chip is thin; electrical circuit of original PCB known)
- during PCB production - possible
  - Such a chip could be integrated directly into the board as a die. If it lies between two metal layers, it is not so easy to detect even with an X-Ray machine.
  - Since such a chip could be installed without a package, the space required would also be very small.
- after PCB fabrication, e.g. as “repair” - not possible if the wiring is not there, or the bad chip works “autonomously” (few connections would be possible, depends on circuit/topology):
  - Precisely drill/mill a small hole to implant the bad chip is possible, especially if the inner PCB layers already have additional wiring (time needed: a few hours).
  - A case from practice: an FPGA with BGA package was soldered incorrectly (2 pins). By means of precise drilling, the wrong wiring was “separated”, additional holes were drilled and a thin wire was laid to make the correct electrical connection (time required: 1-2 days if all information is known).
  - A chip can be implanted by drilling generally:
    - Implanted into the PCB and hidden under components of the PCB or underneath PCB labels (e.g. as a “repair”); re-soldering of components by hand can in principle always be detected because it is optically distinguishable from machine soldering. However, such manipulations can be well camouflaged so that only experts see the re-soldering.
    - Implanted into the chip package from the back, i.e. the components can be manipulated; if a chip manipulated in this way is “soldered in”, it will be difficult to find it optically; X-Ray - possibly if there is wiring (aluminium is not well visible; copper better, gold is well visible); possibly CMR (computer tomography) can help to detect the implants.
    - Resoldering of BGAs is difficult to detect (time needed to resolder a BGA: several hours).

Q.: What other manipulations can be performed to implant an additional chip?
- During chip bonding
  - the bad chip is prepared accordingly: it has TSV (through-silicon via) and the size and PADs like the original chip, then the bad chip can be bonded on top of the original one (time needed: some hours, depends on the number of bonding sites); if the original chip has been maliciously manipulated, e.g. TSV have been implanted (e.g. during the manufacturing process or afterwards e.g. with FIB, the malicious chip can be bonded underneath the original one)
- during/immediately after chip fabrication, or before/during bonding
  - drill/etch a small hole in the back of the chip, insert an additional chip, polish the back of the manipulated chip (time required: several days)
Appendix 6: Attack scenario evaluations

Scenario 1:

Inspired by the Bloomberg files we analyse attacks that start in later development stages i.e. only after all ASICs are manufactured. In addition, the components are considered to be benign.

Attack point 1: Manipulation of the PCB design in the design phase

Who can do this: sub-contractor developing the PCB design

Potential and achievable attack goals:
- **What:** Impacting the reputation of the company that ordered the PCBs design/ eventually also the one that will manufacture it
- **How:** do poor design; plan components with wrong dimensions, wrongly sized wires etc.

Pros:
- all customers of the board will be affected
- wide impact possible
- will be implemented by each PCB manufacturer without knowing about it

Cons:
- none

Potential and achievable attack goals:
- **What:** Espionage i.e. stealing information
- **How:** adding ICs that allow to communicate with the attacker (comparable to malware and its communication with a command and control server); changing GDSII data after manipulation

Pros:
- All customers will be affected
- Somewhat Sophisticated manipulation feasible

Cons:
- Need to find the boards to activate the attack (eventually)
- Target (company) not essentially known

Potential and achievable attack goals:
- **What:** Sabotage i.e. implementing a kill switch
- **How:** adding components that can be activated to destroy the PCB or can be programmed to hit a certain point in time

Pros:
- Less effort than in earlier development stages
- Hits all customers

Cons:
- by far less sophisticated manipulations than in earlier development stages
- Target (company) not essentially known
- Need to find the boards to activate the attack (eventually)
Appendix 6: Attack scenario evaluations

Attack point 2: Manipulation of the PCB after the design phase

Who can do this: PCB Manufacturer

Potential and achievable attack goals:
- **What:** Impacting the reputation of the company that ordered the PCBs / eventually also the one that did the design
- **How:** e.g. exchanging components to less reliable ones (so eventually specific targets can be selected); changing GDSII data after reengineering

**Pros:**
- All customers will be affected
- Somewhat Sophisticated manipulation feasible
- Achievable with rather simple manipulations

**Cons:**
- Reengineering of the PCB design needed (even though less complex than for ASICs)
- Target (company) not essentially known

Potential and achievable attack goals:
- **What:** Espionage i.e. stealing information
- **How:** adding ICs that allow to communicate with the attacker (comparable to malware and its communication with a command and control server); changing GDSII data after reengineering

**Pros:**
- All customers will be affected
- Somewhat Sophisticated manipulation feasible

**Cons:**
- Reengineering of the PCB design needed (even though less complex than for ASICs)
- Target (company) not essentially known
- Need to find the boards to activate the attack (eventually)

Potential and achievable attack goals:
- **What:** Sabotage i.e. implementing a kill switch
- **How:** adding components that can be activated to destroy the PCB or can be programmed to hit a certain point in time

**Pros:**
- Less effort than in earlier development stages
- Hits all customers

**Cons:**
- Reengineering needed to learn how to manipulate the PCB
- by far less sophisticated manipulations than in earlier development stages
- Target (company) not essentially known
- Need to find the boards to activate the attack (eventually)
Appendix 6: Attack scenario evaluations

**Attack point 3: Manipulation of the PCB during its manufacturing**

Who can do this: PCB Manufacturer

Potential and achievable attack goals:

- **What:** Impacting the reputation of the company that ordered the PCBs / eventually also the one that did the design
- **How:** e.g. exchanging components to less reliable ones (so eventually specific targets can be selected); altering the wiring

Pros:
- Relatively easy to do

Cons:
- Target (company) not essentially known
- Hits only those targets that get boards from this manufacturer

Potential and achievable attack goals:

- **What:** Espionage i.e. stealing information
- **How:** adding ICs that allow to communicate with the attacker (comparable to malware and its communication with a command and control server)

Pros:
- Less effort than in earlier development stages

Cons:
- Reengineering needed to learn how to manipulate the PCB
- by far less sophisticated manipulations than in earlier development stages
- Target (company) not essentially known
- Hits only those targets that get boards from this manufacturer
- Need to find the boards to activate the attack

Potential and achievable attack goals:

- **What:** Sabotage i.e. implementing a kill switch
- **How:** adding components that can be activated to destroy the PCB or can be programmed to hit a certain point in time

Pros:
- Less effort than in earlier development stages

Cons:
- Reengineering needed to learn how to manipulate the PCB
- by far less sophisticated manipulations than in earlier development stages
- Target (company) not essentially known
- Hits only those targets that get boards from this manufacturer
- Need to find the boards to activate the attack (eventually)
Appendix 7: Implemented Trojan code

In experiment we implemented a Trojan in an IHP hardware accelerator for elliptic curve point multiplication, shortly denoted as $kP$ operation.

The $kP$ operation is the main operation for digital signature generation and verification approaches, key exchange protocols as well as other cryptographic protocols for Elliptic Curve Cryptosystems. The IHP $kP$ design is a hardware implementation of the $kP$ operation for the Elliptic Curve (EC) B-233, which was standardised by NIST [120] in the past, but recently belongs to “old” curves [121].

The IHP design implements the Montgomery algorithm for the $kP$ operation represented in [122] as Algorithm 2. It is a modification of the Montgomery ladder using Lopez-Dahab projective coordinates for representation of EC points [123], [124]. The inputs of the design are long binary numbers: the scalar $k$ and two affine coordinates of the EC point $P=(x,y)$. The outputs are the affine coordinates of the EC point $(x_1, y_1)=kP$. In authentication and digital signature protocols the scalar $k$ is a secret and the goal of attackers is to reveal this secret. The binary $kP$ algorithms implemented in hardware process the scalar $k$ bit-by-bit. The IHP $kP$ design processes the scalar $k$ starting from its most significant bit.

The Trojan was integrated in the controller of the IHP design, and Figure A7.1 and Figure A7.2 show a “line by line” comparison of the genuine and the manipulated code of the controller, i.e. selected parts of the original (left side) and maliciously modified (right side) Controller’s code. The code lines describing the Trojan are shown on the green background. The Trojan is shown here completely, including its comment lines.

*Figure A7.1: Comparison of the Controller’s source code (1): the original (left side) and maliciously modified (right side) Controller’s code analysed, for the comparison.*
The original IHP file describing the functionality of the block Controller consists of 1,183 lines of code including empty lines and lines containing only comments; the number of characters (excluding the space character) is 14,172 corresponding to the Microsoft Word statistics (see Figure A7.3). For the description of the Trojan functionality only 16 lines of code were inserted in the original design. Some additional lines were used for the comments. Thus, including the comments, the maliciously modified file consists of 14,573 characters (no space) and 1,220 lines, in other words the manipulations led to an increase of the original design of about 3 per cent only.

**Figure A7.2:** Comparison of the Controller’s source code (2): the original (left side) and maliciously modified (right side) Controller’s code analysed, for the comparison.

**Figure A7.3:** Microsoft Word statistics for the IHP files describing Controller’s functionality: the original (left side) and maliciously modified (right side) Controller’s code analysed, for the comparison.
Figure A7.4 and Figure A7.5 show details of the FPGA resource utilisation for the original and maliciously modified designs, for the comparison.

Figure A7.4: FPGA resource utilisation report for the original design.

Figure A7.5: FPGA resource utilisation report for the design with the Trojan. This design uses one pin more but is more compact, needs less resources, i.e. LUTs, slices, and multiplexers, i.e. the design with the Trojan can be (maliciously) represented as an optimised original design.
In the rest of this appendix the utilisation report data are given for the original and maliciously modified designs.

**Original design utilisation report data**

Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.

| Tool Version : Vivado v.2018.3 (win64) Build 2405991 Thu Dec 6 23:38:27 MST 2018 |
| Date : Mon Sep 18 15:55:24 2023 |
| Host : IHP901 running 64-bit major release (build 9200) |
| Command : report_utilization -file C:/Users/LaborA2.55/Desktop/util_orig -name utilization_1_original |
| Design : eccBD_wrapper |
| Device : 7s25csga225-1 |
| Design State : Routed |

Utilization Design Information

Table of Contents

1. Slice Logic
1.1 Summary of Registers by Type
2. Slice Logic Distribution
3. Memory
4. DSP
5. IO and GT Specific
6. Clocking
7. Specific Feature
8. Primitives
9. Black Boxes
10. Instantiated Netlists

1. Slice Logic

<table>
<thead>
<tr>
<th>Site Type</th>
<th>Used</th>
<th>Fixed</th>
<th>Available</th>
<th>Util%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slice LUTs</td>
<td>5834</td>
<td>0</td>
<td>14600</td>
<td>39.96</td>
</tr>
<tr>
<td>LUT as Logic</td>
<td>5834</td>
<td>0</td>
<td>14600</td>
<td>39.96</td>
</tr>
<tr>
<td>LUT as Memory</td>
<td>0</td>
<td>0</td>
<td>5000</td>
<td>0.00</td>
</tr>
<tr>
<td>Slice Registers</td>
<td>3706</td>
<td>0</td>
<td>29200</td>
<td>12.69</td>
</tr>
<tr>
<td>Register as Flip Flop</td>
<td>3706</td>
<td>0</td>
<td>29200</td>
<td>12.69</td>
</tr>
<tr>
<td>Register as Latch</td>
<td>0</td>
<td>0</td>
<td>29200</td>
<td>0.00</td>
</tr>
<tr>
<td>F7 Muxes</td>
<td>251</td>
<td>0</td>
<td>7300</td>
<td>3.44</td>
</tr>
<tr>
<td>F8 Muxes</td>
<td>12</td>
<td>0</td>
<td>3650</td>
<td>0.33</td>
</tr>
</tbody>
</table>

1.1 Summary of Registers by Type

<table>
<thead>
<tr>
<th>Total</th>
<th>Clock Enable</th>
<th>Synchronous</th>
<th>Asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>_</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>_</td>
<td>-</td>
<td>Set</td>
</tr>
<tr>
<td>0</td>
<td>_</td>
<td>-</td>
<td>Reset</td>
</tr>
<tr>
<td>0</td>
<td>_</td>
<td>Set</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>_</td>
<td>Reset</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>135</td>
<td>Yes</td>
<td>-</td>
<td>Set</td>
</tr>
<tr>
<td>2689</td>
<td>Yes</td>
<td>-</td>
<td>Reset</td>
</tr>
<tr>
<td>6</td>
<td>Yes</td>
<td>Set</td>
<td>-</td>
</tr>
<tr>
<td>876</td>
<td>Yes</td>
<td>Reset</td>
<td>-</td>
</tr>
</tbody>
</table>
### 2. Slice Logic Distribution

<table>
<thead>
<tr>
<th>Site Type</th>
<th>Used</th>
<th>Fixed</th>
<th>Available</th>
<th>Util%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slice</td>
<td>1904</td>
<td>0</td>
<td>3650</td>
<td>52.16</td>
</tr>
<tr>
<td>SLICEL</td>
<td>1239</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLICEM</td>
<td>665</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LUT as Logic</td>
<td>5834</td>
<td>0</td>
<td>14600</td>
<td>39.96</td>
</tr>
<tr>
<td>using 05 output only</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>using 06 output only</td>
<td>5074</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>using 05 and 06</td>
<td>760</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LUT as Memory</td>
<td>0</td>
<td>0</td>
<td>5000</td>
<td>0.00</td>
</tr>
<tr>
<td>LUT as Distributed RAM</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LUT as Shift Register</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slice Registers</td>
<td>3706</td>
<td>0</td>
<td>29200</td>
<td>12.69</td>
</tr>
<tr>
<td>Register driven from within the Slice</td>
<td>1948</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Register driven from outside the Slice</td>
<td>1758</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LUT in front of the register is unused</td>
<td>1044</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LUT in front of the register is used</td>
<td>714</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unique Control Sets</td>
<td>73</td>
<td></td>
<td>3650</td>
<td>2.00</td>
</tr>
</tbody>
</table>

*Note: Available Control Sets calculated as Slice Registers / 8, Review the Control Sets Report for more information regarding control sets.

### 3. Memory

<table>
<thead>
<tr>
<th>Site Type</th>
<th>Used</th>
<th>Fixed</th>
<th>Available</th>
<th>Util%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block RAM Tile</td>
<td>0</td>
<td>0</td>
<td>45</td>
<td>0.00</td>
</tr>
<tr>
<td>RAMB36/FIFO*</td>
<td>0</td>
<td>0</td>
<td>45</td>
<td>0.00</td>
</tr>
<tr>
<td>RAMB18</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>0.00</td>
</tr>
</tbody>
</table>

*Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate only one FIFO36E1 or one FIFO18E1. However, if a FIFO18E1 occupies a Block RAM Tile, that tile can still accommodate a RAMB18E1

### 4. DSP

<table>
<thead>
<tr>
<th>Site Type</th>
<th>Used</th>
<th>Fixed</th>
<th>Available</th>
<th>Util%</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSPs</td>
<td>0</td>
<td>0</td>
<td>80</td>
<td>0.00</td>
</tr>
</tbody>
</table>

### 5. IO and GT Specific

<table>
<thead>
<tr>
<th>Site Type</th>
<th>Used</th>
<th>Fixed</th>
<th>Available</th>
<th>Util%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bonded IOB</td>
<td>7</td>
<td>7</td>
<td>150</td>
<td>4.67</td>
</tr>
<tr>
<td>IOB Master Pads</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOB Slave Pads</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bonded IPADs</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0.00</td>
</tr>
<tr>
<td>PHY_CONTROL</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0.00</td>
</tr>
<tr>
<td>PHASER_REF</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0.00</td>
</tr>
<tr>
<td>OUT_FIFO</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>0.00</td>
</tr>
<tr>
<td>IN_FIFO</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>0.00</td>
</tr>
<tr>
<td>IDELAYCTRL</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Appendix 7: Implemented Trojan code

6. Clocking

<table>
<thead>
<tr>
<th>Site Type</th>
<th>Used</th>
<th>Fixed</th>
<th>Available</th>
<th>Util%</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFGCTRL</td>
<td>2</td>
<td>0</td>
<td>32</td>
<td>6.25</td>
</tr>
<tr>
<td>BUFIO</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>0.00</td>
</tr>
<tr>
<td>MMCME2_ADV</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>33.33</td>
</tr>
<tr>
<td>PLL2_ADV</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0.00</td>
</tr>
<tr>
<td>BUFRMCE</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0.00</td>
</tr>
<tr>
<td>BUFHCE</td>
<td>0</td>
<td>0</td>
<td>48</td>
<td>0.00</td>
</tr>
<tr>
<td>BUFR</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>0.00</td>
</tr>
</tbody>
</table>

7. Specific Feature

<table>
<thead>
<tr>
<th>Site Type</th>
<th>Used</th>
<th>Fixed</th>
<th>Available</th>
<th>Util%</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSCANE2</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0.00</td>
</tr>
<tr>
<td>CAPTUREE2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.00</td>
</tr>
<tr>
<td>DNA_PORT</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.00</td>
</tr>
<tr>
<td>EFUSE_USR</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.00</td>
</tr>
<tr>
<td>FRAME_ECCE2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.00</td>
</tr>
<tr>
<td>ICAPE2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0.00</td>
</tr>
<tr>
<td>STARTUPE2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.00</td>
</tr>
<tr>
<td>XADC</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.00</td>
</tr>
</tbody>
</table>

8. Primitives

<table>
<thead>
<tr>
<th>Ref Name</th>
<th>Used</th>
<th>Functional Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUT6</td>
<td>3820</td>
<td>LUT</td>
</tr>
<tr>
<td>FDCE</td>
<td>2689</td>
<td>Flop &amp; Latch</td>
</tr>
<tr>
<td>LUT4</td>
<td>1557</td>
<td>LUT</td>
</tr>
<tr>
<td>FDRE</td>
<td>876</td>
<td>Flop &amp; Latch</td>
</tr>
<tr>
<td>LUT5</td>
<td>662</td>
<td>LUT</td>
</tr>
<tr>
<td>LUT2</td>
<td>315</td>
<td>LUT</td>
</tr>
<tr>
<td>MUXF7</td>
<td>251</td>
<td>MuxFx</td>
</tr>
<tr>
<td>LUT3</td>
<td>235</td>
<td>LUT</td>
</tr>
<tr>
<td>FDPE</td>
<td>135</td>
<td>Flop &amp; Latch</td>
</tr>
<tr>
<td>MUXF8</td>
<td>12</td>
<td>MuxFx</td>
</tr>
<tr>
<td>FDSE</td>
<td>6</td>
<td>Flop &amp; Latch</td>
</tr>
<tr>
<td>LUT1</td>
<td>5</td>
<td>LUT</td>
</tr>
<tr>
<td>IBUF</td>
<td>4</td>
<td>IO</td>
</tr>
<tr>
<td>CARRY4</td>
<td>4</td>
<td>CarryLogic</td>
</tr>
<tr>
<td>OBUF</td>
<td>3</td>
<td>IO</td>
</tr>
<tr>
<td>BUF4</td>
<td>2</td>
<td>Clock</td>
</tr>
<tr>
<td>MMCME2_ADV</td>
<td>1</td>
<td>Clock</td>
</tr>
</tbody>
</table>
### 9. Black Boxes

<table>
<thead>
<tr>
<th>Ref Name</th>
<th>Used</th>
</tr>
</thead>
</table>

### 10. Instantiated Netlists

<table>
<thead>
<tr>
<th>Ref Name</th>
<th>Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>eccBD_util_vector_logic_0_0</td>
<td>1</td>
</tr>
<tr>
<td>eccBD_ecc_uart_0_0</td>
<td>1</td>
</tr>
<tr>
<td>eccBD_clk_wiz_1_0</td>
<td>1</td>
</tr>
</tbody>
</table>

---

**Trojan utilisation report data**

Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.

<table>
<thead>
<tr>
<th>Tool Version</th>
<th>Vivado v.2018.3 (win64) Build 2405991 Thu Dec 6 23:38:27 MST 2018</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>Mon Sep 18 15:58:49 2023</td>
</tr>
<tr>
<td>Host</td>
<td>IHP901 running 64-bit major release (build 9200)</td>
</tr>
<tr>
<td>Command</td>
<td>report_utilization -file C:/Users/Labor2.55/Desktop/Trojan_util -name Trojan_utilization_1</td>
</tr>
<tr>
<td>Design</td>
<td>eccBD_wrapper</td>
</tr>
<tr>
<td>Device</td>
<td>7s25csga225-1</td>
</tr>
<tr>
<td>Design State</td>
<td>Routed</td>
</tr>
</tbody>
</table>

---

**Table of Contents**

1. Slice Logic
   1.1 Summary of Registers by Type
2. Slice Logic Distribution
3. Memory
4. DSP
5. IO and GT Specific
6. Clocking
7. Specific Feature
8. Primitives
9. Black Boxes
10. Instantiated Netlists

1. Slice Logic

<table>
<thead>
<tr>
<th>Site Type</th>
<th>Site</th>
<th>Used</th>
<th>Fixed</th>
<th>Available</th>
<th>Util%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slice LUTs</td>
<td>5461</td>
<td>0</td>
<td>14600</td>
<td>37.40</td>
<td></td>
</tr>
<tr>
<td>LUT as Logic</td>
<td>5461</td>
<td>0</td>
<td>14600</td>
<td>37.40</td>
<td></td>
</tr>
<tr>
<td>LUT as Memory</td>
<td>0</td>
<td>0</td>
<td>5000</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Slice Registers</td>
<td>3708</td>
<td>0</td>
<td>29200</td>
<td>12.70</td>
<td></td>
</tr>
<tr>
<td>Register as Flip Flop</td>
<td>3708</td>
<td>0</td>
<td>29200</td>
<td>12.70</td>
<td></td>
</tr>
<tr>
<td>Register as Latch</td>
<td>0</td>
<td>0</td>
<td>29200</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>F7 Muxes</td>
<td>95</td>
<td>0</td>
<td>7300</td>
<td>1.30</td>
<td></td>
</tr>
<tr>
<td>F8 Muxes</td>
<td>12</td>
<td>0</td>
<td>3650</td>
<td>0.33</td>
<td></td>
</tr>
</tbody>
</table>
1.1 Summary of Registers by Type
----------------------------------------
<table>
<thead>
<tr>
<th>Total</th>
<th>Clock Enable</th>
<th>Synchronous</th>
<th>Asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>-</td>
<td>Set</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>-</td>
<td>Reset</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>Set</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>Reset</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>135</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2690</td>
<td>Yes</td>
<td>-</td>
<td>Reset</td>
</tr>
<tr>
<td>6</td>
<td>Yes</td>
<td>Set</td>
<td>-</td>
</tr>
<tr>
<td>877</td>
<td>Yes</td>
<td>Reset</td>
<td>-</td>
</tr>
</tbody>
</table>

2. Slice Logic Distribution
---------------------------
<table>
<thead>
<tr>
<th>Site Type</th>
<th>Used</th>
<th>Fixed</th>
<th>Available</th>
<th>Util%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slice</td>
<td>1717</td>
<td>0</td>
<td>3650</td>
<td>47.04</td>
</tr>
<tr>
<td>SLICEL</td>
<td>1132</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLICEM</td>
<td>585</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LUT as Logic</td>
<td>5461</td>
<td>0</td>
<td>14600</td>
<td>37.40</td>
</tr>
<tr>
<td>using O5 output only</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>using O6 output only</td>
<td>4890</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>using O5 and O6</td>
<td>571</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LUT as Memory</td>
<td>0</td>
<td>0</td>
<td>5000</td>
<td>0.00</td>
</tr>
<tr>
<td>LUT as Distributed RAM</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LUT as Shift Register</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slice Registers</td>
<td>3708</td>
<td>0</td>
<td>29200</td>
<td>12.70</td>
</tr>
<tr>
<td>Register driven from within the Slice</td>
<td>1870</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Register driven from outside the Slice</td>
<td>1838</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LUT in front of the register is unused</td>
<td>1063</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LUT in front of the register is used</td>
<td>775</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unique Control Sets</td>
<td>74</td>
<td></td>
<td>3650</td>
<td>2.03</td>
</tr>
</tbody>
</table>

* Note: Available Control Sets calculated as Slice Registers / 8, Review the Control Sets Report for more information regarding control sets.

3. Memory
---------
<table>
<thead>
<tr>
<th>Site Type</th>
<th>Used</th>
<th>Fixed</th>
<th>Available</th>
<th>Util%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block RAM Tile</td>
<td>0</td>
<td>0</td>
<td>45</td>
<td>0.00</td>
</tr>
<tr>
<td>RAMB36/FIFO*</td>
<td>0</td>
<td>0</td>
<td>45</td>
<td>0.00</td>
</tr>
<tr>
<td>RAMB18</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>0.00</td>
</tr>
</tbody>
</table>

* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate only one FIFO36E1 or one FIFO18E1. However, if a FIFO18E1 occupies a Block RAM Tile, that tile can still accommodate a RAMB18E1

4. DSP
-----
<table>
<thead>
<tr>
<th>Site Type</th>
<th>Used</th>
<th>Fixed</th>
<th>Available</th>
<th>Util%</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSPs</td>
<td>0</td>
<td>0</td>
<td>80</td>
<td>0.00</td>
</tr>
</tbody>
</table>
5. IO and GT Specific

<table>
<thead>
<tr>
<th>Site Type</th>
<th>Used</th>
<th>Fixed</th>
<th>Available</th>
<th>Util%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bonded IOB</td>
<td>8</td>
<td>8</td>
<td>150</td>
<td>5.33</td>
</tr>
<tr>
<td>IOB Master Pads</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOB Slave Pads</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bonded IPADs</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0.00</td>
</tr>
<tr>
<td>PHY_CONTROL</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0.00</td>
</tr>
<tr>
<td>PHASER_REF</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0.00</td>
</tr>
<tr>
<td>OUT_FIFO</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>0.00</td>
</tr>
<tr>
<td>IN_FIFO</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>0.00</td>
</tr>
<tr>
<td>IDELAYCTRL</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0.00</td>
</tr>
<tr>
<td>IBUFDS</td>
<td>0</td>
<td>0</td>
<td>144</td>
<td>0.00</td>
</tr>
<tr>
<td>PHASER_OUT/PHASER_OUT_PHY</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>0.00</td>
</tr>
<tr>
<td>PHASER_IN/PHASER_IN_PHY</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>0.00</td>
</tr>
<tr>
<td>IDELAYE2/IDELAYE2_FINEDELAY</td>
<td>0</td>
<td>0</td>
<td>150</td>
<td>0.00</td>
</tr>
<tr>
<td>ILOGIC</td>
<td>0</td>
<td>0</td>
<td>150</td>
<td>0.00</td>
</tr>
<tr>
<td>OLOGIC</td>
<td>0</td>
<td>0</td>
<td>150</td>
<td>0.00</td>
</tr>
</tbody>
</table>

6. Clocking

<table>
<thead>
<tr>
<th>Site Type</th>
<th>Used</th>
<th>Fixed</th>
<th>Available</th>
<th>Util%</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFGCTRL</td>
<td>2</td>
<td>0</td>
<td>32</td>
<td>6.25</td>
</tr>
<tr>
<td>BUFO</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>0.00</td>
</tr>
<tr>
<td>MMCME2 ADV</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>33.33</td>
</tr>
<tr>
<td>PLL E2 ADV</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0.00</td>
</tr>
<tr>
<td>BUFMRCE</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0.00</td>
</tr>
<tr>
<td>BUFHCE</td>
<td>0</td>
<td>0</td>
<td>48</td>
<td>0.00</td>
</tr>
<tr>
<td>BUFRR</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>0.00</td>
</tr>
</tbody>
</table>

7. Specific Feature

<table>
<thead>
<tr>
<th>Site Type</th>
<th>Used</th>
<th>Fixed</th>
<th>Available</th>
<th>Util%</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSCANE2</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0.00</td>
</tr>
<tr>
<td>CAPTUREE2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.00</td>
</tr>
<tr>
<td>DNA_PORT</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.00</td>
</tr>
<tr>
<td>EFUSE USR</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.00</td>
</tr>
<tr>
<td>FRAME ECCE2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.00</td>
</tr>
<tr>
<td>ICAPE2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0.00</td>
</tr>
<tr>
<td>STARTUPE2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.00</td>
</tr>
<tr>
<td>XADC</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.00</td>
</tr>
</tbody>
</table>

8. Primitives

<table>
<thead>
<tr>
<th>Ref Name</th>
<th>Used</th>
<th>Functional Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDCE</td>
<td>2690</td>
<td>Flop &amp; Latch</td>
</tr>
<tr>
<td>LUT6</td>
<td>2573</td>
<td>LUT</td>
</tr>
<tr>
<td>LUT4</td>
<td>1935</td>
<td>LUT</td>
</tr>
<tr>
<td>LUT5</td>
<td>985</td>
<td>LUT</td>
</tr>
</tbody>
</table>
Appendix 7: Implemented Trojan code

| FDRE   | 877 | Flop & Latch |
| LUT3   | 419 | LUT          |
| FDPE   | 135 | Flop & Latch |
| LUT2   | 114 | LUT          |
| MUXF7  | 95  | MuxFx        |
| MUXF8  | 12  | MuxFx        |
| LUT1   | 6   | LUT          |
| FDSE   | 6   | Flop & Latch |
| OBUF   | 4   | IO           |
| IBUF   | 4   | IO           |
| CARRY4 | 4   | CarryLogic   |
| BUFG   | 2   | Clock        |
| MMCME2_ADV | 1 | Clock      |

9. Black Boxes
--------------

<table>
<thead>
<tr>
<th>Ref Name</th>
<th>Used</th>
</tr>
</thead>
</table>

10. Instantiated Netlists
-------------------------

<table>
<thead>
<tr>
<th>Ref Name</th>
<th>Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>eccBD_util_vector_logic_0_0</td>
<td>1</td>
</tr>
<tr>
<td>eccBD_ecc_uart_0_0</td>
<td>1</td>
</tr>
<tr>
<td>eccBD_clk_wiz_1_0</td>
<td>1</td>
</tr>
</tbody>
</table>
Appendix 8: PCB experiment Additional details

Additional experiments were run with IHP chips to determine whether the aluminium bonding wires would show up in the X-Ray image, see Figure A8.1.

![Figure A8.1: X-Ray and optical images of the IHP chip selected for the experiments, with Aluminium bond wires.](image1)

Figure A8.2 shows optical images for the 2nd experiment with Coil 1: we implanted the Infineon OPTIGA™ Trust B Authentication IC (SLE952500000XTSA1) into the compound material of Coil 1 using the same steps as for implanting the IHP chip, i.e. as it described in Subsection 4.5.1.

![Figure A8.2: Optical images of the steps in 2nd experiment: a) potting compound carefully milled open; b) chip glued into the milled hole.](image2)
The following table gives a list of the equipment used in the experiments with the PCB.

**Table A8.1: Equipment used in the experiments with the PCB.**

<table>
<thead>
<tr>
<th>Device</th>
<th>Cost</th>
<th>Picture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microscope Leica S9i [117]</td>
<td>7 000 Euro</td>
<td><img src="image" alt="Microscope" /></td>
</tr>
<tr>
<td>X-Ray Inspection Device Nikon XT V 160 [118]</td>
<td>180 000 Euro</td>
<td><img src="image" alt="X-Ray Device" /></td>
</tr>
<tr>
<td>Bonder F&amp;S Bondtec 5632 (Wedge-Wedge, Al-Wires) [119]</td>
<td>140 000 Euro</td>
<td><img src="image" alt="Bonder" /></td>
</tr>
<tr>
<td>Dremel with milling head</td>
<td>~500 Euro</td>
<td><img src="image" alt="Dremel" /></td>
</tr>
<tr>
<td>Soldering iron and tweezers</td>
<td>~1 000 Euro</td>
<td><img src="image" alt="Soldering Iron and Tweezers" /></td>
</tr>
</tbody>
</table>