High Impedance: Analysis of Publicly Disclosed Vulnerabilities in FPGA IP Cores

Imtiaj A. Chowdhury
Vrije Universiteit Amsterdam[§]
Amsterdam, Netherlands
i.a.chowdhury@student.vu.nl

Eric Ackermann, Sven Bugiel
CISPA Helmholtz Center for Information Security
Saarbrücken, Germany
{*eric.ackermann,bugiel*}*@cispa.de*

***Abstract*—Field-Programmable Gate Arrays (FPGAs) facilitate a write-compile-debug cycle for digital circuits, resembling modern software development. Crucially, FPGA circuits comprise *modules* that can be *re-used* across projects. Many generic modules are available from commercial and open-source vendors as *Intellectual Property (IP) cores*.**

However, using third-party IP cores in FPGA designs poses critical supply chain security risks: FPGAs are increasingly being used in security- and safety-critical applications like automotive driver assistance or core networking. Thus, vulnerabilities in any component—homegrown or licensed—can have far-reaching consequences.

While susceptibility of FPGA IP to security-critical implementation bugs has been shown in the literature, the *prevalence* and *management* of security vulnerabilities in FPGA IP has never been studied quantitatively. In this paper, we present the first measurement study covering 19 representative IP cores with 1250 total known issues over three commercial vendors and one open-source vendor. We classify 284 issues as security critical, predominantly affecting availability (134) and integrity (184). We analyze the survivability of these vulnerabilities and their mitigations. We find that mean-time-to-fix is between 5 and 12 months for commercial vendors, with outliers of up to 70 months, and no available mitigations for certain vulnerabilities. Specifically, 14 issues have *fixes that support fewer FPGA devices than the vulnerable version*, including 11 issues that *vendors themselves label as critical but for which no workaround exists*. We conclude with concrete recommendations for the industry, based on lessons learned from software vulnerability management.

***Index Terms*—FPGA security, Vulnerability Survey, IP Core Security, Hardware Security**

1. Introduction

Developing a novel integrated circuit (IC) solution costs significant time and money. While still true for Application-Specific Integrated Circuits (ASICs), the proliferation of Field-Programmable Gate Array (FPGA) technology challenges this assumption: FPGA toolchains allow compiling (synthesizing) Hardware Description Language (HDL) source code into a circuit that can be programmed onto an FPGA *reversibly*. This allows FPGA developers to follow a write-compile-debug cycle, more similar to software development than traditional hardware engineering, speeding up development drastically [1].

§. Work predominantly conducted during internship at CISPA.

While sometimes only used for testing circuits that go on to become ASICs, case studies support that FPGAs are increasingly being placed on custom PCBs for use in real-world products. This is especially true for low-volume devices, where lower non-recurring engineering costs are more important than per-unit manufacturing costs. For example, FPGAs built by Microchip are used as a crucial bridge technology for AI-based healthcare applications [2]. FPGAs by Xilinx (now AMD) serve driver assistance like emergency braking in Subaru cars [3]. Several radiation-hardened Xilinx FPGAs have also been sent to Mars as a part of the Perseverance rover [4].

In addition to the faster development cycle, the technology-independent HDLs allow *sharing components across projects* as *Intellectual Property (IP) cores*. Thereby, instead of having to reinvent the wheel for each project, FPGA developers can *import* cost-effective off-the-shelf IP cores for solved problems and focus on novel components that provide stakeholder value. Naturally, sharing IPs in this manner can be facilitated between different legal entities with the help of *license agreements*. In essence, one can use *IP cores* in the same way as *third-party libraries* in software development.

In order to illustrate this argument, consider the hypothetical scenario of building a System-on-Chip (SoC) for an autonomous mobile robot that, e.g., transports items in a factory from a location A to B. The SoC can be built almost entirely out of existing general-purpose components: The basis for the SoC can be an AMD Zynq-7000 microprocessor with integrated FPGA [5] on a custom PCB. The FPGA can then implement a Camera Serial Interface (CSI) controller interfacing with the camera [6], [7] for navigation and an Ethernet controller [8] for communication with wired networks. Finally, custom hardware components can process the camera feed for navigation and obstacle detection and control the motors inside the robot. These custom components can interact with the SoC using standard buses such as Advanced eXtensible Interface (AXI) [9]. Infrastructure for this purpose is abundantly available in AMD's FPGA toolchain, Vivado [10].

Looking back at our example, which parts does the SoC vendor actually *have to* build from scratch? In fact, they only have to supply components that support their *project-specific business logic*, relying on commercial-off-the-shelf components for the remaining design. Thereby, the vendor can reduce the development timeline from potentially years to mere months.

However, this accelerated development cycle comes with supply chain risks: In addition to the intended func-

tionality, the vendor in the example is exposed to any implementation defects in the licensed IP cores. In particular, the vendor might have inadvertently imported *exploitable security vulnerabilities* into the design.

Qualitatively, it is well-documented that exploitable hardware vulnerabilities do exist and can have a severe impact on overall product security [11], [12], [13], [14], [15]. However, there are no *quantitative* studies that cover FPGA designs. Thereby, it is difficult to judge how serious this problem is in real-world FPGA designs. Drawing parallels to the development of security-critical software, supply-chain risks should never be taken lightly: large-scale surveys across publicly known issues in the National Vulnerability Database (NVD) spanning over a decade each highlight that the rate and severity of vulnerabilities being found in the software supply chain only ever increases, even in security-critical software [16], [17], [18].

The only quantitative analysis of hardware vulnerabilities solely covers x86 CPUs by AMD and Intel, which are *hardwired* ICs [19]. Due to their immutable nature, significant effort is invested in testing hardwired ICs like CPUs and ASICs to ensure the intended function is implemented correctly and securely [20], [21], [22], [23], [24], [25]. In contrast, as the name suggests, FPGAs can be updated in the field. Hence, any defect in FPGA designs or IP cores can just be patched over the air. Thereby, it stands to reason that verification *might* have lower priority, and vulnerabilities *might* thus be propagating into real-world devices more frequently. However, confirming or disputing this hypothesis requires *quantitative data*.

This paper addresses this research gap by performing a comprehensive measurement study in vulnerability management across major commercial and open source FPGA IP core vendors. We believe that the results of this study can pave the way for increased awareness of the security risks associated with FPGA IP and improve the handling of such vulnerabilities, both on the part of IP vendors and on the part of IP users. Similar studies have been conducted in the software domain, spanning domains such as cryptographic libraries or the Android ecosystem. These works have increased developer awareness when it comes to patching vulnerabilities and helped library developers direct attention to the right places when it comes to preventing vulnerabilities [16], [17], [18], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], just as we intend with this study.

Contributions. We make the following contributions:

- We conduct a systematic survey of publicly available implementation issues affecting FPGA IP cores. In total, we surveyed 1250 issues across 14 commercial and 5 open-source cores from 4 vendors. Our data set is available for future research [38].
- Using manual classification, we determine 284 of these issues to be security-relevant and provide insights into the violated security goals. We classify the security-relevant issues into 6 recurring *archetypes*.
- We derive insights into how vendors patch the aforementioned vulnerabilities:
  - We found that 103 out of 284 issues cannot be patched without upgrading to a new version of the FPGA toolchain, which may require the purchase of a new license or result in the loss of access to end-of-life IP cores. The survivability of identified issues varies between less than one month and a maximum of 70 months, with a median of 0 to 6 months, depending on the vendor.
  - We identify 14 out of 284 issues that potentially affect a legacy Intel device family, but provide no patch for the last version of the FPGA toolchain that supports this device family. According to the information provided, 11 of these issues *cannot be mitigated in any way* on these legacy devices *while being assessed as critical by Intel*.
  - Overall, our results indicate that it is practically infeasible for FPGA developers to ensure that mitigations for all issues affecting the cores they use are applied. Thereby, we identify an urgent need for the FPGA industry to adopt existing vulnerability management practices from the software domain and improve supply-chain security.

2. Background

This section provides a brief overview of logic design in general, designing logic for FPGAs specifically, and the use of IP cores. The reader is referred to the excellent books by Harris and Harris [39] and Taraate [1] for additional details.

2.1. Digital Logic Design

Using hardware description languages (HDLs) like Verilog or VHDL, engineers can specify the behavior of digital circuits in a technology-independent manner. HDLs structure circuits as a tree of *modules* with explicit inputs and outputs. Inputs and outputs of the top module are treated as IC electrical pins. Modules can use stateful (sequential, e.g., registers) or stateless (combinational, e.g., if-then-else statements) logic to express behavior.

The *design synthesis* process translates a design from human-readable code into a *netlist*. The netlist consists solely of primitive cells in the target technology and simple wire connections. To this end, the synthesis compiler *infers* primitives such as registers, combinational logic and memories from HDL based on its semantics.

The synthesized netlist is not directly manufacturable as an ASIC or loadable onto an FPGA, as it still abstracts from electrical details. Instead, *placement* and *routing* phases need to be completed. Placement determines a specific physical location for a cell in the netlist, e.g., a grid location in an ASIC or a *specific* primitive cell on an FPGA. Routing instantiates physical connections between placed cells as specified in the netlist using routing primitives provided by the technology. The routed design can then be manufactured or loaded onto an FPGA.

Circuits operate with a specified input clock that has physical characteristics such as period, duty cycle, and jitter. Combinational cells and routes have a *delay*, while registers have *setup and hold times*, necessitating input data to arrive no later than a point in time and to remain stable until a certain point in time. Therein, setup and hold times are always *relative to the clock edge*. Violating these timing requirements can cause *metastability*, an electrical phenomenon that describes registers being in an undefined state, or write *stale or inconsistent data*. Hence, FPGA

and ASIC toolchains use *static timing analysis* (STA) during place-and-route in order to ensure that timing requirements are met and the circuit functions correctly.

2.2. FPGA Technology

FPGAs internally consist of primitives like *registers* for state, *arithmetic primitives* like adders, and *lookup tables (LUTs)* for logic. LUTs essentially implement *truth tables* in hardware: they contain an exhaustive *mapping table* that describes the mapping of input tuples to output tuples and can be *reversibly* changed when *programming* the FPGA. LUTs and registers can be connected using reconfigurable *routing primitives* to form *arbitrary circuits*.

FPGA toolchains, such as AMD's Vivado [10], consist of the same components (synthesis compiler, placer, and router) as ASIC toolchains. However, they are normally specific to the vendor of the target FPGA.

2.3. FPGA Intellectual Property (IP) cores

HDL modules facilitate *design reuse* across projects. Hence, *generic components* are often developed and made available to other parties as *intellectual property (IP) core*. Thereby, IP cores are an equivalent of software libraries, allowing developers to focus on their primary business logic while incorporating existing solutions whenever possible akin to modern software development.

In the hardware domain, two types of IP core need to be distinguished: *hard* and *soft* IP cores. Hard IP cores refer to *primitives* of FPGAs that are instantiated as black boxes in a design and resolved by the placer. Soft IP cores on the other hand are provided as HDL or netlist, consisting of generic logic cells just like the rest of the design. In essence, soft IP cores resemble *software libraries* while hard IP cores are similar to *compiler intrinsics*.

3. Related Work

As already pointed out in Section 1, security vulnerabilities in hardware have been studied extensively *using qualitative methods*. Also, the orthogonal problems of *prevention* and *containment* of hardware bugs have been studied. Finally, we would like to emphasize the similarity of our work with research into *software libraries*.

Several taxonomies of hardware-specific attacks have been published [11], [12], [14], [40], [41], [42], [43]. However, in contrast to this work, these taxonomies are based on attacks presented in the academic literature.

A *quantitative* taxonomy of in-the-wild vulnerabilities in IP cores, targeting FPGAs or ASICs, does not yet exist. We believe both studies will benefit the community. We focus on FPGAs due to their *reconfigurability*: vulnerabilities in FPGA designs can be patched using an IP update, followed by a new synthesis, place, and route cycle, and reconfiguration of existing customer devices. Thus, vulnerability management, with an emphasis on patching, akin to best practices in software development, can ensure the security of end-user devices. However, ASICs are *immutable*. Thus, there may be no straightforward way to fix vulnerabilities that affect them. On the other hand, publishing vulnerabilities can enable in-the-wild exploitation of these bugs in customer devices. Hence, vendors might be hesitant to publish ASIC vulnerabilities, making a comprehensive study of ASIC vulnerabilities difficult. Bellay et al. argue that this may be one of the reasons why few CVEs that affect ASICs are reported to the NVD [44].

The closest work to our study is *RemembERR* by Solt et al. [19]. In their study, the authors systematically surveyed implementation issues in x86 CPUs by AMD and Intel. The authors classified the vulnerabilities and analyzed them for patterns, with the goal of preventing similar bugs in the future. Overall, our methodology is very similar to Solt et al., but instead of (hardwired) CPUs, we focus on FPGA IP cores.

An orthogonal problem to surveying IP core vulnerabilities is to detect or prevent vulnerabilities before shipping an IP core, such as *hardware trojans* [45]. Research studies with the intent to find illicit modifications or counterfeit ICs have also been conducted [44], [46]. Also, much work has been conducted with the intent to find *inadvertently introduced* logic bugs in *behavioral simulation* during development and testing [20], [21], [22], [23], [24], [25]. Researchers have also attempted to detect previously unknown vulnerabilities in publicly available IP cores [13], [41], [47]. Finally, with the goal of *preventing* the introduction of bugs, formal verification approaches have been explored [48], [49], [50].

A second orthogonal problem is handling vulnerabilities at run time without requiring re-synthesis and re-configuration of an FPGA or re-manufacturing of an ASIC. To this end, both patching [51] and containment [52], [53] of vulnerable IP cores has been studied.

Finally, we should point out the large corpus of similar studies conducted in the software space, many of which have proven highly insightful and affected real change in the industry. Over the last 20 years, researchers have repeatedly conducted large-scale analysis of publicly known software vulnerabilities [16], [18], [26], [27], [28], [29], [30], [31]. To this end, the National Vulnerability Database (NVD) operated by NIST has been used as a ground truth. Overall, the studies have analyzed the survivability or time between introduction and patching of vulnerabilities, as well as broad patterns in cause and type of vulnerabilities. The results were used to gauge the security posture of the security community as a whole and to determine where to place more effort during development. Overall, the methodology of these works is very similar to our study.

Similar investigations with a narrower scope and deeper analysis have also been conducted. Two particularly interesting examples are cryptographic software and the Android operating system. In cryptographic software, studies often focused on understanding problems with the developer mindset and development procedures that caused certain patterns in recurring issues [17], [32], [54]. The results of these studies helped direct focus during development. In the Android domain, the primary research objective was to investigate the survivability of vulnerabilities, with the goal of identifying methods to deliver critical security patches more efficiently. Additionally, research developed predictive methods to automatically detect and fix vulnerabilities, and attempted to identify especially vulnerable components that require more scrutiny [33], [34], [35], [36], [37].

4. Motivation and Research Questions

As pointed out in Section 3, many surveys of vulnerabilities and patching have been conducted in the software development domain. Thereby, research work has contributed to gaining a better understanding of security risks and their specific causes. Programmers also have gained insights into where to direct their attention during development and testing of security-relevant software.

As software depends on hardware to *guarantee any security at all*, vulnerabilities in any hardware component have the potential to be catastrophic. However, structured insights into vulnerabilities that affect hardware—specifically, FPGA IP cores—did not exist to a satisfactory degree at the time of writing [44]. Hence, we conduct a survey of FPGA vulnerabilities using the established methodologies from the software domain to close this gap.

In particular, we answer these research questions:

Disclosure of IP Core Vulnerabilities

1) Which ways of disclosing vulnerabilities in IP cores do FPGA IP vendors use?
2) Are there vendors whose vulnerability information is not publicly available?
3) Do vendors distinguish security-critical vulnerabilities from functional bugs?

Categorization of IP Core Vulnerabilities

4) Which security goals do FPGA IP core vulnerabilities affect?
5) Which types of vulnerabilities occur in FPGA IP cores?
6) Are there measurable patterns in the disclosure of FPGA IP core vulnerabilities?
7) Do FPGA IP vendors categorize security issues according to vulnerability classification schemes like Common Weakness Enumeration (CWE)?

Mitigations of IP Core Vulnerabilities

8) What types of mitigations for the found vulnerabilities were proposed?
9) Do mitigations involve re-synthesis of the FPGA design, or can the mitigations be applied at run time?
10) What is the survivability of vulnerabilities?
11) Are there vulnerabilities *without mitigations*?

5. Methodology

This section details our methodology for collecting and analyzing IP cores and their known issues. We start with a high-level overview over the methodology, before detailing the individual stages (identification of vendors, aggregation of per-core issues, filtering, and analysis).

5.1. Overview

We used a structured methodology for collecting IP cores and their issues, similar to previous works in the hardware [19] and software [54] domains.

First, we identified representative vendors of FPGA cores. We selected the vendors AMD (formerly Xilinx), Intel (formerly Altera), and Lattice Semiconductor based on *market share* in the FPGA industry. We used *GitHub stars* as a proxy metric for popularity for selecting the open-source IP core collection LiteX by enjoy-digital.

In a second step, we identified the total number of IP cores for each vendor. We used information from the vendors' websites or GitHub for this purpose.

We then identified the *total* number of known issues per core, using information provided by the vendor. We discuss this step separately for each vendor. In this step, the analysis of the issues had not yet been conducted.

We selected three representative IP cores per vendor. After reviewer feedback, we extended this to five IP cores. For AMD, Intel, and Lattice, we selected these cores by the *number of issues* found, excluding any end-of-life cores. For Lattice, only four cores had security-relevant issues. For the open-source vendor LiteX, we selected the five cores with the highest number of GitHub stars.

In a final filtering step, we ensured the collected issues truly pertained to the intended IP cores. This step was only necessary for Intel, which was the only vendor for which we had no structured way to retrieve these issues.

A PRISMA diagram [55] with annotations that describe the surveyed cores and issues per stage can be found in Figure 1. The issues in the last stage of the PRISMA diagram were classified by two independent reviewers.

5.2. Identification and Aggregation of Issues

Using two subsequent collection stages, we identified the total number of IP cores provided by each vendor, followed by the total number of issues per core. We illustrate the distribution of issues over the top i cores in fig. 2. We selected the top five cores for further analysis. For Lattice, due to the low number of issues, we surveyed *all* cores and found only four cores with security-relevant issues. Our collection comprises issues published no later than September 30$^{\text{th}}$, 2025.

5.2.1. Vendors Considered. We aimed at capturing a representative set of *big players* offering at least 5 IP cores. Unfortunately, we were unable to find publicly available market share information for *FPGA IP specifically*. However, FPGA *device* vendors usually provide a rich ecosystem of first-party IP cores alongside their toolchains. Hence, we selected FPGA IP vendors based on *FPGA device market share* [56]. The five main FPGA vendors are AMD (formerly Xilinx), Intel (formerly Altera), Lattice, Microchip and Achronix. Of these, Intel and AMD cover almost the entire market. Due to a lack of publicly available issues for Microchip and Achronix, our final selection comprised AMD, Intel, and Lattice.

For selecting open-source vendors, we started the identification process with the communities LiteX [57] (made by vendor enjoy-digital), PULP [58], OpenCores.org [59] and OpenHW Group [60] based on our *expert knowledge*. We discarded OpenCores.org, as the community has seen little to no activity in the past decade. We selected *LiteX* due to having the highest number of GitHub stars, our proxy metric for popularity.

IP cores and their associated issues were then collected using a combination of manual and automated crawling, which is discussed in the following sections for each individual vendor. For all vendors, our selection process

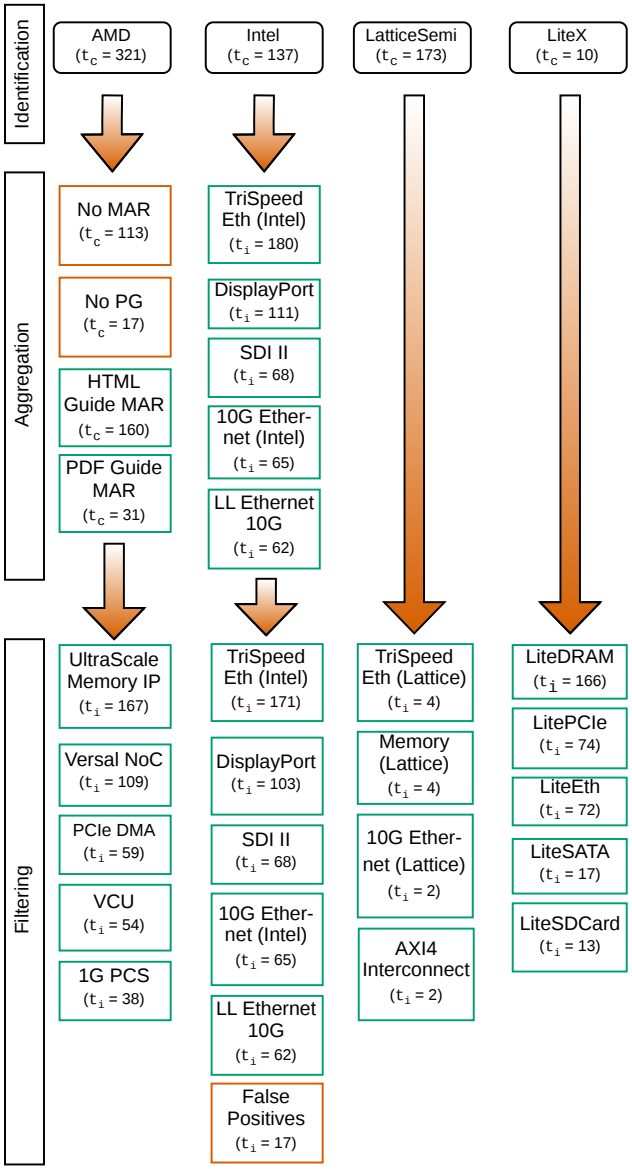

Figure 1: PRISMA diagram quantifying the paper collection process. We quantify the total cores per vendor t_c and total number of issues per core t_i. Green boxes mark cores and issues included in intermediate stages, orange boxes mark filtered issues and cores.

solely comprised *soft* and *mixed* IP cores, i.e., cores that are partially or fully distributed in source or netlist form. We will discuss this decision in Section 7.5.

5.2.2. AMD. *IP Core collection.* AMD provides one publicly accessible product guide (PG) in HTML or PDF format for each IP core, typically containing a link to an entry in AMD's knowledge base known as the *master answer record* (MAR). The master answer record lists all known issues for the respective core, with a unique answer number assigned to each issue, making the scraping of these issues a straightforward process.

To identify available IP cores, we utilized AMD's Support Keyword Search [61] portal with an empty argument and filtered the results for IP cores and product guides. In total, we identified 321 product guides in the portal. For 160 product guides, we found a HTML product guide page with a link to the master answer record. Out of the remaining 161 product guides,

- 142 product guides were PDF documents, 111 of which did not have any link to the associated master answer record of known issues,
- 17 product guides did not exist in AMD's database,
- 2 product guides were valid HTML product guides, but lacked the link to the master answer record.

After identifying all IP cores, we aggregated the issues for each core by parsing the 191 identified master answer records. To this end, we used an automatic crawler.

Challenges. First, we realized that AMD has official *errata* accessible through Support Keyword Search [61]. These errata mark security-related issues in the corresponding Xilinx answer records and, in certain cases, contain information about violated security properties and reference assigned CVEs. Unfortunately, the errata only comprise *FPGA primitives and functions* and ignore any *FPGA IP* that does not directly correspond to a primitive, making them irrelevant for this study.

Second, while collecting issues with the help of the master answer record worked well for most cores, the master answer record was not linked in the product guide for many of the IP cores. In this case, we tried to find it by searching for it in the AMD Technical Information Portal [62]. If we did not find the master answer record, we excluded that particular core from our analysis.

Additionally, we noted that for some issues, no *Answer Record* with a detailed description was available. We later excluded these issues from our analysis.

Also, for the *UltraScale Memory IP* core, the MAR linked MARs for other variants of the core. Hence, we had to manually parse these MARs recursively. Hence, the core has more issues (167) than initially crawled (52).

Finally, for some legacy IP cores, we did not find accessible issues. While the product guides of these cores often contained answer numbers, we were unable to resolve these numbers in AMD's support database. We assume that this documentation was lost in a migration of AMD's (then: Xilinx') support system around 2013. We requested confirmation from the vendor, who was unfortunately not able to confirm this.

5.2.3. Intel. *IP Core collection.* Intel provides an overview of all FPGA IP cores it offers [63]. In total, they list 74 IP cores. However, notably, some listed IP cores actually refer to IP core collections: For example, the *Video and Vision Processing Suite* contains 49 IP cores, and the *Video and Image Processing Suite* contains 20 IP cores. For the purposes of our study, each component of these suites was treated as a separate core, and issues were counted for one *specific* core, resulting in 137 IP cores.

Intel provides an *FPGA Knowledge Base* [64] site, which lists all known IP core issues. However, the site does not allow filtering for issues related to specific IP cores by default. Additionally, no equivalent of AMD's *master answer record*, which lists a set of issues for respective IP cores, exists. Hence, we had to resort to *full-text search* to identify all issues for a single IP core. After considering the listed IP cores and discarding IP cores for which we could not find any known issues in the knowledge base, we surveyed 87 IP cores.

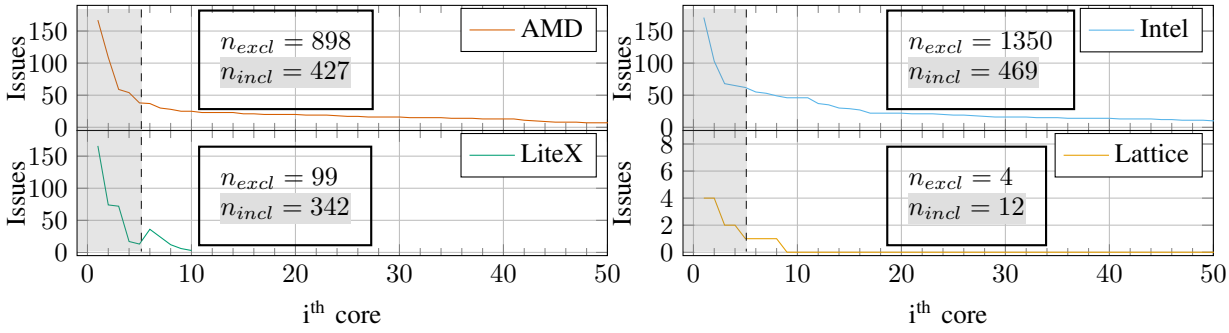

Figure 2: Issues per core over cores in ranked order (number of issues or stars, September 2025). Dashed line denotes cut-off for data collection. Total number of included and excluded issues per vendor listed.

When we collected issues for two additional IP cores in March 2026, we found that the *FPGA Knowledge Base* [64] did not return *any bug reports for any FPGA IP core any more*. It appears that between the first and second collection phases, Intel migrated known issues to the *Altera Documentation and Resources Center* [65]. Hence, we collected issues for the *SDI II* and *Low-Latency 10G Ethernet MAC* IP cores via the *Altera Documentation and Resources Center*, filtering for *Knowledge Articles* and using the same strategy as for the Intel database. The dataset for the other cores and the ranking was crawled exclusively from Intel's *FPGA Knowledge Base* [64].

Challenges. First, similar to AMD, Intel publishes explicit errata for device *primitives* only [66]. While the user guides of certain IP cores explicitly link errata, all of these links lead to 404 error pages at the time of writing.

Second, as pointed out earlier, identifying all issues with each individual IP core is unnecessarily complicated at times. We found that issues related to a particular IP core can only be found by utilizing free text search. To this end, we had to constrain the search string using quotes to ensure the *full search string* matched the results. At the same time, we noticed that the names of IP cores were used inconsistently. For example, for the "HDMI Intel® FPGA IP Core," issues referred to this core as "HDMI Intel® FPGA IP Core," "HDMI Intel® FPGA IP," or even only "HDMI IP." Thus, we always used the most generic version of the IP name and removed unrelated issues in the final filtering stage. The issue is not unique to the defunct Intel FPGA knowledge base [64], the new Altera knowledge base [65] behaves in the same way.

5.2.4. Lattice. Lattice lists 350 IP cores in their solutions page [67]. However, many of the listed cores are provided by Lattice's partners. Hence, we filtered for Lattice's IP cores only, resulting in a total of 173 cores.

Each IP core has a site with a *Documentation* section, which links to the core's user manual as a PDF. The *User Manual* contains descriptions of known issues. Due to this well-structured documentation, we were able to utilize an automatic crawler to identify the user guides for all IP cores and automatically parse the PDF documents to determine the total number of known issues. Overall, we only found 7 IP cores with listed known issues.

5.2.5. LiteX (OpenSource vendor). The LiteX Cores Ecosystem [68] covers all IP cores (10 in total) currently offered by the vendor. We were able to collect issues for each core from the GitHub issue tracker.

5.3. Selection of Cores to Survey

For the commercial vendors, we selected the five (or four, due to lack of cores with security-relevant issues for Lattice) cores with the highest number of issues. We chose this criterion for the following reason: We have no direct information about the *popularity* and *maturity* of a core. However, a core with a high number of issues has likely been in use for a longer period and has a higher number of users who encounter and report defects. Hence, by selecting such cores, we increase the generalizability and accuracy of our results.

For the open-source vendors, we instead selected the five cores with the highest number of GitHub stars, serving as a proxy metric for *popularity*. The reason for the different selection methodology was that the GitHub issues of these cores do not differentiate between defects of the core and general Q&A. Therefore, the total number of issues was not as reliable as for the commercial vendors.

We skipped the *LiteScope* [69] core in our analysis. The core is *solely used for debugging* circuits during development, serving as an alternative to vendor-specific embedded logic analyzers. Hence, we are confident that this core is *never* used in a security-sensitive setting.

5.4. Filtering of Issues

This step was only necessary for Intel cores. As explained in Section 5.2.3, we had to resort to text search to aggregate known issues of each core. Also, as known issues for Intel cores sometimes use different spellings of the core's name, we had to search using the most general version of the core's name, which is also distinctive enough to distinguish it from similarly named cores. Hence, our original data set contained a small number of issues that were mapped to the *incorrect IP core*, which we manually removed from our dataset in this step.

5.5. Analysis of Issues

Research questions 1, 2 and 3 concern how vulnerability information are disclosed *qualitatively* by the vendors and what information is available to the public. Hence, we were able to answer these questions using our insights

during the collection process. Furthermore, we extracted the following information from the collected issues in furtherance of our research questions:

- the security relevance for research question 4
- the type of security vulnerability and security goal affected for research questions 4, 5, 6 and 7
- type and timeline of patches and workarounds for research questions 8, 9, 10 and 11

All issues analyzed pertain to undesirable behavior of a core in some context, i.e., implementation bugs. Within the scope of our study, we were only interested in *security vulnerabilities*, i.e., bugs that can be *exploited* to compromise a security property of the system. Importantly, this heavily depends on *application context*: in some contexts, the inputs that trigger a bug can never be given to the core. Also, the *impact on security goals* may not be practically relevant in some applications. Related studies by Solt et al. [19] and Zonta-Roudes et al. [47] faced the same issue in their analysis. In both studies, the authors considered bugs that were *input-dependent* and *caused a deviation from documented behavior* to be security-relevant, without assuming any particular application context. We follow this methodology and consider all issues that can be triggered by external inputs and *whose effects can affect* any of the standard security goals *confidentiality*, *integrity*, and *availability* of the overall system to be *security relevant*.

Further, we considered issues only when they pertained to problems with the core itself *and* it was feasible to ship a design with the vulnerable core. Hence, all issues attributed exclusively to behavior in simulation, documentation, placement, or software stack failures were discarded. Likewise, issues that prevented a core from completing place-and-route were excluded. On the other hand, we did consider timing violations: While most of these problems are apparent after place-and-route, designers may ship designs with slight timing violations or (perhaps unintentionally) exclude certain paths from timing analysis altogether. Still, timing violations can cause incorrect behavior in certain conditions depending on the (physical) environment and the exact sequence of inputs, making them potentially exploitable to an adversary in order to change the behavior of the core [14].

For the classification of vulnerability types, we found the Common Weakness Enumeration (CWE) to be an inadequate system: The abstraction levels of CWEs pertaining to hardware vary drastically, and many vulnerabilities matched multiple or no CWEs. This is not a novel insight and has been argued before: Bellay et al. [44] argue that existing CWEs were derived from a limited set of publicly known vulnerabilities in the hardware domain, making it difficult to distill categories with general validity. Hence, we used the same iterative methodology as Solt et al. [19] for building our own classification scheme: Starting without any categories, we iterated through our data set. When we encountered an issue that did not match any of the existing categories, we attempted to generalize it and added it to the relevant categories. After a first pass through the vulnerabilities, we merged infrequent categories into a more general category where possible.

We relied on *manual analysis* for classifying security goals and vulnerability types. Author A_1 initially developed the categories, then discussed them with author A_2 until a consensus was reached. Authors A_1 and A_2 then *independently* rated the categories and affected security goal for each issue. We computed an inter-rater agreement (IRA) using Krippendorff's alpha [70] and Cohen's kappa [71] with the Jaccard metric [72]. We achieved κ: 0.72, α: 0.69 for issue categories and κ: 0.60, α: 0.43 for security goals. Notably, A_2 systematically treated security goals differently for timing vulnerabilities, resulting in 34 disagreements. Ignoring these, the IRA for security goals would be κ: 0.64, α: 0.51. A_1 and A_2 then reached consensus on the labels for categories and security goals. To this end, all issues with disagreements were discussed in full, and a fitting label was selected.

5.6. Security Model

We defined our security goals *confidentiality, integrity, availability* in the following way:

- *Confidentiality* issues allow access to data different from what the specification requires for an input, e.g., incorrect address mapping in a memory controller.
- *Integrity* issues pertain to generation of incorrect output data that the surrounding design cannot immediately detect, e.g., reading stale data from a bus.
- *Availability* issues pertain to scenarios in which a core cannot perform its specified function any more, e.g., a crash-fault or hang.

6. Results

This section comprises our results. Our structured vulnerability collection process, detailed in Section 5, allowed us to answer several research questions related to the dissemination of vulnerability information. We will discuss these findings in the order of discovery in Section 6.1, Section 6.2 and Section 6.3. We continue with an overview of the surveyed issues and affected security goals in Section 6.4. We then provide insights into trends in vulnerabilities by analyzing reporting times and vulnerability types according to our derived categories for each surveyed core, as presented in Section 6.5 and Section 6.6. This is followed by a discussion of vulnerability survivability in Section 6.7 and the type of mitigation provided in Section 6.8.

We provide an overview of the surveyed IP cores in Table 1. Note that despite its name, the *Intel® Agilex™ 7 and Intel® Stratix® 10 FPGA E-Tile Hard IP core* is actually a *mixed* core containing mostly soft logic, making it possible for Intel to make significant changes to its behavior in IP updates. Hence, we covered it in our collection. In total, out of 1250 issues, we classified 284 (≈22.7%) as security-critical.

6.1. Ways of Disclosing Vulnerabilities

We found that commercial vendors AMD and Intel collect known issues in a searchable database. For most AMD cores (59.5%), there is a *master answer record (MAR)* that lists all known issues in a table structure, e.g., [73]. Individual issues are described with a brief description of the symptom and the cause(s) of the problem, along with a suggested fix or solution, e.g., [74]. For Intel, individual issues are published in the same

format as for AMD. However, there is no structured MAR, and one must use a free-text search to find all answer records for a core. Lattice documents known issues with their IP cores in each core's user guide, e.g., [75]. Issue descriptions are notably shorter than for Intel or AMD. Finally, the surveyed open-source community LiteX uses GitHub issues to disseminate known issues with their cores, e.g., [76].

Result for Research Question 1

Which ways of disclosing vulnerabilities in IP cores do FPGA IP vendors use?

Different methods of disclosing vulnerabilities were employed: AMD and Intel provide proprietary searchable databases of known issues, where AMD's database is easier to search automatically than Intel's. Lattice discloses vulnerabilities in IP user guides. The open-source vendor LiteX uses GitHub issues to that end.

6.2. Access to Vulnerability Information

Vulnerability information was found to be publicly available for all surveyed vendors. However, we doubt the *durability* of vulnerability information. AMD and Intel only collect such information in proprietary databases, which are subject to change or deletion at any time. For Lattice, we found evidence of patched issues being removed from updated product guides [75].

Result for Research Question 2

Are there vendors whose vulnerability information is not publicly available?

All surveyed vendors made vulnerability information publicly available, but Lattice *removes known issues as they are patched*. This raises questions about the *durability* of vulnerability information.

6.3. Triage of Known Issues by Vendors

We found that AMD and Intel provide curated lists of known security vulnerabilities for *primitives of the FPGA devices themselves*. These are *hard macros* and were thus out of scope for this study. However, none of the surveyed vendors mark any (soft or mixed) IP core issues as security-critical. Of the four surveyed vendors, only Intel makes an effort to assess the severity of IP issues: Some issues are explicitly marked as *critical*. However, we are unsure which process is used to determine this rating. For example, consider the following issues with identical impact on the same core and similar preconditions, both reported in 2012:

> "[...] This will cause intermittent failure on the Ethernet data path, auto negotiation process, and Ethernet link synchronization." [77]
>
> "Intermittent Ethernet link failure and data corruption occurs when [...]" [78]

The former issue [77] is *not* marked critical, whereas the latter issue [78] is. Out of 469 surveyed issues for Intel, we found 155 (33%) of the issues to be security critical; however, Intel only reported 114 of these issues (73.5%) as critical. Therefore, although there is a substantial overlap, we found Intel's criticality rating to be unreliable. We inquired about the process for determining this label, but received no reply before submission.

Results for Research Questions 3 and 7

Do vendors distinguish security-critical vulnerabilities from functional bugs (3)?
Do FPGA IP vendors categorize security issues according to vulnerability classification schemes like Common Weakness Enumeration (CWE) (7)?

None of the vendors *explicitly* mark issues that affect security. No vulnerability classification scheme is used. Only Intel designates issues as *critical*. However, only 73.5% of the identified security issues for Intel were marked *critical*, raising questions about the usefulness and process for determining this label.

6.4. Security Goals Affected by Vulnerabilities

As Table 1 shows, the surveyed issues pertain mostly to integrity violations (184) or availability violations (134), with issues potentially affecting both security goals. This matches the qualitative results of Zonta-Roudes et al. [47], who tested AXI IP cores for previously unknown vulnerabilities: Most results from their survey fall into the same categories. We also found 6 issues, mostly in memory IP, that due to reordering or incorrect addressing can violate *confidentiality*, e.g., in conjunction with a CPU that supports userspace-kernelspace isolation. For 13 issues, we were unable to derive the affected security goal from the description. They are counted as *ambiguous*.

Result for Research Question 4

Which types of security vulnerabilities that affect FPGA IP cores have been disclosed?

Our survey reveals issues that predominantly jeopardize the *availability* (134) and *integrity* (184) of a hardware IP core. *Confidentiality* compromises depend on the context, e.g., using a memory IP with a CPU that separates kernel and user space.

6.5. Vulnerability Archetypes

We provide the final categories into which we classified the surveyed issues in Table 2. An overview of the distribution of vulnerability types per vendor and IP core is provided in Figure 3. Overall, the most common vulnerability types are *Data Corruption*, *Initialization Fault*, *Timing Violation*, and *Specification Mismatch*.

Of these problems, *Initialization Faults* are the most benign: According to our definition, this comprises an IP not completing an explicit initialization step and clearly indicating that it was not initialized properly. Thereby, these issues only impact availability.

The most insidious are *data corruption* vulnerabilities: Corrupted data may be detected based on incorrect checksums. However, in case data is corrupted *after* checksum verification, incorrect data can propagate through the system for a long time without causing a detectable issue,

TABLE 1: Surveyed IP cores and classification of aggregated issues across all vendors. We provide the full IP name and reference, vendor, release year, number of issues in total and number of security vulnerabilities for the security goals Integrity (*I*), Availability (*A*), Confidentiality (*C*) or ambiguous, as well as total security vulnerabilities and vulnerabilities assessed *critical* by the vendor. One issue might fall into multiple categories.

IP Core	**Vendor**	**Release**	**Total**	**Number of Security Vulnerabilities**					
				I	*A*	*C*	Ambiguous	Total Security	Critical
UltraScale Architecture-Based FPGAs Memory IP LogiCORE IP [79]—*UltraScale Memory IP*	AMD	2013	167	29	12	3	1	37 (22%)	-
Versal Adaptive SoC Programmable Network on Chip and Integrated Memory Controller 1.1 LogiCORE IP [80]—*Versal NoC*	AMD	2020	109	5	2	1	1	6 (6%)	-
DMA/Bridge Subsystem for PCI Express [81]—*PCIe DMA*	AMD	2016	59	5	11	0	1	15 (25%)	-
H.264/H.265 Video Codec Unit v1.2 Solutions LogiCORE IP [82]—*VCU*	AMD	2017	54	2	3	0	0	5 (11%)	-
1G/2.5G Ethernet PCS/PMA or SG-MII LogiCORE IP [83]—*1G PCS*	AMD	2012	38	4	10	0	1	14 (37%)	-
Triple-Speed Ethernet FPGA IP [84]—*Tri-Speed Eth (Intel)*	Intel	2007	171	35	16	0	2	44 (26%)	28 (64%)
DisplayPort Intel® FPGA IP Core [85]—*DisplayPort*	Intel	2012	103	31	38	0	1	55 (53%)	48 (87%)
SDI II FPGA IP [86]—*SDI II*	Intel	2012	68	10	9	0	0	13 (19%)	11 (85%)
Intel® Agilex™ 7 and Intel® Stratix® 10 FPGA E-Tile Hard IP [87]—*10G Ethernet (Intel)*	Intel	2019	65	15	11	0	1	23 (35%)	13 (57%)
Low Latency Ethernet 10G MAC FPGA IP [88]—*LL 10G*	Intel	2013	62	14	6	0	1	20 (32%)	14 (70%)
Tri-Speed Ethernet IP Core [89]—*Tri-Speed Eth (Lattice)*	Lattice	2019	4	3	0	0	0	3 (75%)	-
Memory Controller IP Core [75]—*Memory (Lattice)*	Lattice	2021	4	0	0	0	2	2 (50%)	-
10Gb Ethernet MAC+PHY IP Core [90]—*10G Ethernet (Lattice)*	Lattice	2023	2	2	0	0	0	2 (100%)	-
AXI4 Interconnect Module [91]—*AXI4 Interconnect*	Lattice	2022	2	1	0	0	0	1 (50%)	-
LiteDRAM [92]	LiteX	2015	166	14	3	1	3	19 (11%)	-
LitePCIe[93]	LiteX	2015	74	2	2	1	1	4 (5%)	-
LiteEth [94]	LiteX	2015	72	11	8	0	0	15 (21%)	-
LiteSATA [95]	LiteX	2015	17	0	0	0	2	2 (12%)	-
LiteSDCard [96]	LiteX	2017	13	1	3	0	0	4 (31%)	-

making it challenging to trace any crash back to its origin when it occurs. Furthermore, it is possible that data errors are never noticed by the application.

Timing violation vulnerabilities are also particularly problematic, as they can be *masked* in certain contexts, but cause severe problems in others: Assume a DRAM memory controller IP violates t_{RAS}, i.e., waits fewer cycles between the *row active* and *precharge* commands than its configuration indicates. Some memory modules will be able to tolerate a lower t_{RAS}—this parameter is fixed by the memory module vendor at design time according to a cost-performance trade-off. Therefore, a design that previously worked flawlessly might cease to function after changing the memory supplier. A different type of timing violation might be a slight setup or hold violation on a path in the design that only causes issues under high temperatures or low voltages, making them hard to identify and replicate.

The impact of timing violations is also difficult to predict: typically, they manifest as a register or memory writing stale or invalid data. Violations of, e.g., DRAM timings, might also cause *irreproducible* read errors. Similar to data corruption bugs, if and when these incorrect data cause a visible issue also varies between contexts.

Finally, the impact of *specification mismatch* vulnerabilities varies greatly. Crucially, the observable issue created by these vulnerabilities again depends on the context and possibly software or other hardware interfacing with the IP. The impact of such vulnerabilities can range from availability (e.g., an omitted interrupt) to full loss of confidentiality, integrity and availability.

Prevention strategies necessitated by these issues also vary: *Data Corruption* and especially *Specification Mismatch* bugs should be detectable during technology-independent verification, indicating insufficient verification efforts by the vendors [23]. *Initialization Fault* vulnerabilities may be specific to certain scenarios and, e.g., only appear when the IP is operating with a specific configuration and communicating with a specific counterpart, making them difficult to systematically rule out. Here,

TABLE 2: Classes of vulnerabilities iteratively derived.

Category	Description
Specification Mismatch (115)	Core does not adhere to its specification, *in a way different from other categories*.
Data Corruption (91)	Core *unexpectedly* modifies input data.
Timing Violation (70)	Core violates timing requirements, covering explicit timing constraints and implicit requirements in the specification.
Initialization Fault (33)	Core does not (successfully) complete an *explicit* initialization step.
Ambiguous (21)	Insufficient information provided.
Hang (15)	Core stops processing inputs *indefinitely* and *cannot be recovered* without reset.
Race Condition (6)	Core behaves incorrectly if inputs arrive in a specific order or timing sequence.

manual testing is needed. Finally, *timing violations* are a mixed bag. Intra- and many inter-clock timing violations are usually caught by static timing analysis (STA) after place-and-route. However, more complex scenarios, such as clock domain crossings, are excluded from STA using timing constraints. STA tools also do not understand, e.g., chip-specific DDR memory timings or phase relationships for a medium-independent interface. Overall, carefully studying the FPGA tool's output should help minimize timing vulnerabilities that propagate into customer designs. However, for certain scenarios, domain knowledge and creation of specialized test benches are required.

Result for Research Question 5

Which types of vulnerabilities occur in FPGA IP cores?

Data Corruption, *Initialization Fault*, *Timing Violation*, and *Specification Mismatch* are the most common vulnerability types observed in our data set. The prevalence of *Data Corruption* and *Specification Mismatch* vulnerabilities can be mitigated through design verification, leveraging existing techniques. However, while verification processes and the FPGA tools' static timing analysis can help, *Initialization Fault* and *Timing Violation* vulnerabilities require manual prevention and testing.

6.6. Vulnerability Reporting

We visualize reporting patterns over time for the surveyed vulnerabilities in Figure 4. Lattice does not provide these information, while AMD and Intel only provide these information for *some (mostly older) cores*. Hence, all Lattice cores and two cores from Intel and AMD, respectively, are missing in the chart.

The first main insight from the chart is that *vulnerabilities continue to be found over the lifetime of the IP core*. This was to be expected—as pointed out in Section 1, FPGA development is not dissimilar to software development. Hence, over the lifetime of an IP, vendors are under similar pressure to extend the scope of the original IP core as software developers: Customers expect IPs to support novel device families (e.g., Xilinx 6/7 series), protocols (e.g., MDIO/SGMII/XGMII for Ethernet, DDR4/5 for DRAM) or features (e.g., address mirroring for DRAM) over time. Vendors add these features to the core, which is why the codebase evolves continuously. At the same time, new bugs continue to be introduced.

Especially for AMD, there is a noticeable accumulation of reported issues early in the core's life. This might indicate that vendors do the best they can in terms of verification and rely on their customers to provide bug reports, capturing scenarios that the vendor did not predict during verification. At the same time, this insight emphasizes the need to exercise caution when upgrading an IP core, as developers may find themselves transitioning from a stable end-of-life core to a new design with new bugs. However, AMD and Intel often provide implausible or no release dates for their cores, which may slightly exaggerate the correlation in our data set.

Result for Research Question 6

Are there measurable patterns in the disclosure of FPGA IP core vulnerabilities?

We derived two main insights into the disclosure patterns of FPGA IP cores:

- Vulnerabilities are introduced over the entire lifetime of an IP core.
- Especially AMD experiences an accumulation of issues early in the life of an IP core.

6.7. Vulnerability Survivability

We analyze the survivability of surveyed vulnerabilities in Figure 5 and Table 3. We only consider cores for which plausible release and patch dates for vulnerabilities are provided. Lattice is omitted due to lack of data.

We detect a striking difference between LiteX and the commercial vendors: both the graph and statistical evaluation show that the mean time to fix for LiteX is *significantly lower*. In fact, most vulnerabilities raised for LiteX cores were fixed within less than one month, with a standard deviation below four months. The median time to fix for AMD and Intel is 5 and 6 months, respectively, with standard deviations around 10 months. The maximum reported time to fix for Litex (17 months) is also significantly shorter than for AMD and Intel (43 and 70 months, respectively). Due to limited insights into AMD's and Intel's internal processes, we can only speculate about possible reasons for their longer time to fix. However, we are fairly confident in the following contributing factors:

- As LiteX is open-source, it is easier for consumers of the cores to analyze incorrect behavior of LiteX cores in simulation or even after synthesis. Thus, many surveyed security-critical issues already contained a suspected root cause (45.45%) or proposed mitigation (52.27%).
- Similarly, many designs that use LiteX are themselves open-source. Thus, the design in which the core behaves unexpectedly can be shared with the vendor more easily, making it easier for the vendor to triage reported issues.
- AMD and Intel release IP cores alongside their FPGA toolchains, Vivado [10] and Quartus [97]. Depending on the time period and vendor, there are between two and four minor releases of the toolchain per year. Thereby, even if a fix for a vulnerability is developed

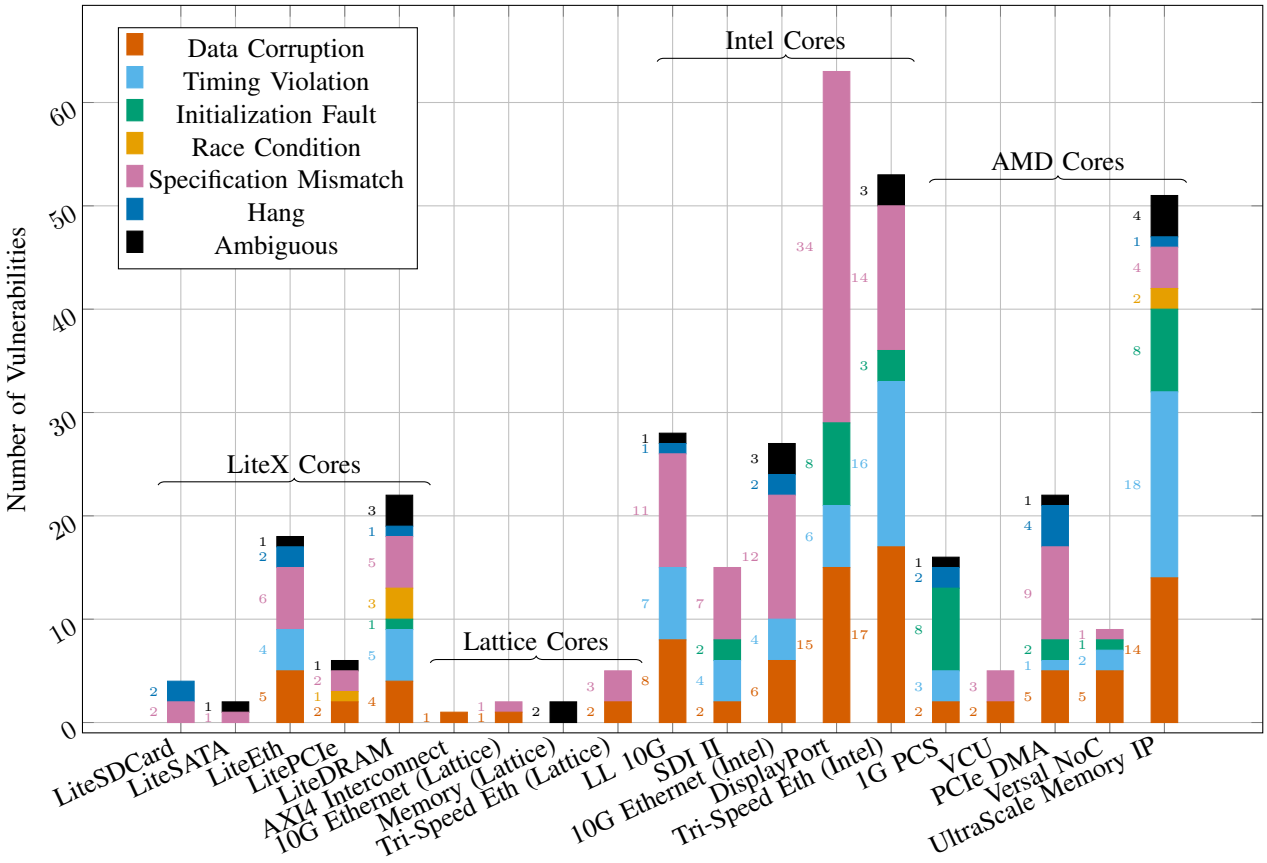

Figure 3: Types of vulnerability according to Table 2. IP core names abbreviated (see Table 1).

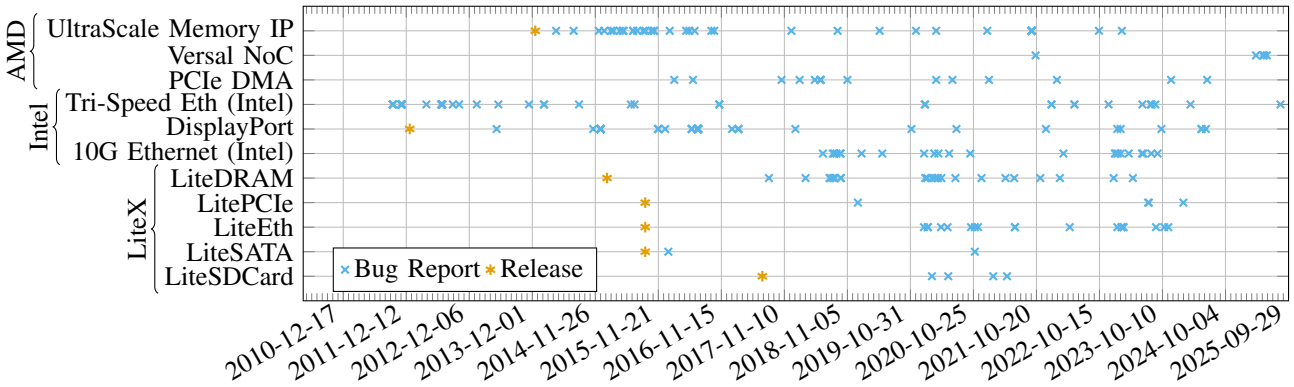

Figure 4: Report dates for all surveyed vulnerabilities and vendors. IP core names abbreviated (see Table 1). Approximate IP core release date provided if known and plausible. Cores without plausible data omitted.

immediately, it will only be available to the public the next time the FPGA toolchain is released. Thus, depending on when a bug is reported in the release cycle, the fix might be delayed for several months until the next release.

- In our survey, we discarded issues without a clear resolution. Thus, it is possible that fixes for discarded LiteX issues will be released in the future, increasing the mean time to fix.

Result for Research Question 10

What is the survivability of vulnerabilities?

The survivability of IP vulnerabilities varies among vendors. We found that LiteX has a lower survivability (median: 0, σ: 3.49, max: 17 months) than AMD (median: 5, σ: 9.04, max: 43 months) and Intel (median: 6, σ: 12.15, max: 70 months).

6.8. Vulnerability Mitigation

We classify provided vulnerability mitigations for all four vendors in Figure 6. We found four categories of mitigations as detailed in Table 4.

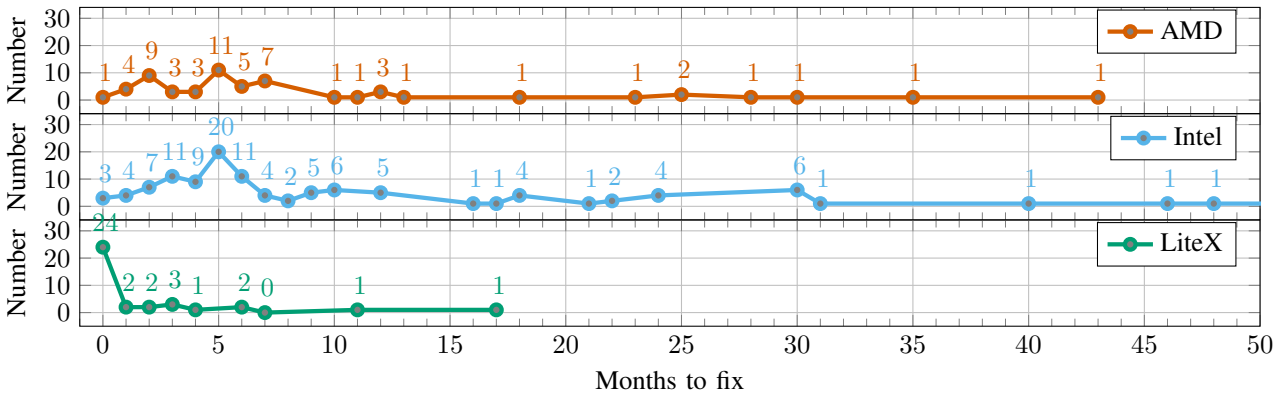

Figure 5: Survivability in months for discovered vulnerabilities across vendors. Lattice omitted due to lack of data. Ignoring two Intel outliers (58, 70 months) as well as currently unfixed issues (AMD: 2, Intel: 3, Litex: 8).

TABLE 3: Statistical evaluation of vulnerability survivability across vendors. No data for Lattice.

Category	AMD	Intel	LiteX
Median	5	6	0
σ	9.04	12.15	3.49
P_{90} (linear)	23.80	29.40	5.0
P_{99} (linear)	38.52	56.90	14.90
P_{100}	43	70	17

TABLE 4: Classes of mitigations found.

Mitigation	Description
Update (241)	New *official release* of the IP core, with a known date or promised in a future release.
Workaround (94)	Manual change in the configuration, netlist or sources of the IP core or the user design.
Patch (44)	Downloadable file that can be applied against the *current* version of the IP core.
Unknown/None (13)	Vendor does not provide mitigation for the issue or we could not determine the type.

Unsurprisingly, most issues with IP cores are remedied through updates of the core, usually through periodic toolchain updates (commercial vendors). However, patches or workarounds are often provided sooner, as there might be users who are unable to wait for the update.

Crucially, all mitigations require *re-synthesis*: an FPGA engineer must manually incorporate the mitigation into the design, run a full synthesis-place-and-route cycle, and ship an updated bitstream. Depending on the complexity of a design and workstation performance, place-and-route alone can take several CPU-days [98].

Also, updates might require the purchase of a new toolchain license. For Intel, we identified 14 issues (11%) that affect a version that drops support for a device, but are only patched in a version that *no longer supports the same device*. In particular, all of these issues affect Intel Quartus 13.1, which removes support for the Cyclone through Cyclone III, Stratix through Stratix III, Max 3000A and Max 7000, and HardCopy II through IV families. 11 of these issues also *provide no workaround or patch* for versions prior to the fix, whereas *all of them are marked as critical by the vendor*. AMD has also removed support for certain devices from its software, but we encountered no similar unpatchable issues.

Result for Research Questions 8,9,11

What types of mitigations for the found vulnerabilities were proposed (8)?
Do mitigations involve re-synthesis of the FPGA design, or can the mitigations be applied at run time (9)?
Are there vulnerabilities without mitigations (11)?

Most vulnerabilities are mitigated through updates, often tied to a new FPGA toolchain version. Vendors also often provide workarounds or patches. Irregardless, all mitigations require *re-synthesis*.
Some surveyed issues have no known mitigation at the time of writing. Also, not all patches are back-ported to earlier toolchain versions. In particular, we identified 14 issues that affect legacy Intel devices, but for which a patch is only provided for more recent toolchain versions that no longer support these devices.

7. Threats to Validity

In this section, we discuss the validity and generalizability of our results: We discuss limitations in the available source data set and vulnerability information in Section 7.1 and Section 7.2, the exploitability of surveyed issues in Section 7.3, the lack of root cause analysis in Section 7.4, and the collection scope in Section 7.5.

7.1. Disorganization of Source Data Sets

First and foremost, as we already pointed out in Section 5, effort needed for aggregating issues per core varied greatly between vendors. AMD provided structured lists of *all* known issues pertaining to specific cores in a single place, known as *Master Answer Record* (MAR). Therefore, collection of issues per core was a mostly automatic process. Even then, for appx. half of the surveyed cores, the MAR was absent, limiting the scope of our study.

For Intel, we had to use full-text search to aggregate issues per core. We already discussed in Section 5 that we took measures to prevent falsely attributing issues to a core, thereby reliably preventing false positives. However, it is also possible that a small number of issues per core was missed in the count, causing a small number of false negatives. Given the large number of issues we were able

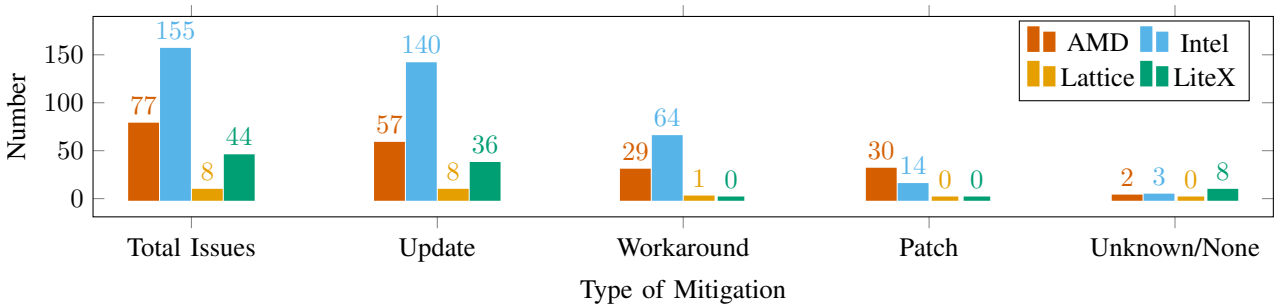

Figure 6: Type of mitigation across AMD, Intel, Lattice and LiteX. One vulnerability might fall into multiple categories.

to attribute to the selected cores, however, we are confident in the representativeness of our results.

For Lattice, compared to the other vendors, we found relatively few known issues with the cores. We found evidence that the issues listed only pertained to the latest release of the core, as known issues were noted to have been removed [75]. Thereby, unfortunately, our insights from analyzing Lattice issues specifically are limited and might not be representative.

In contrast to the commercial vendors, the surveyed open-source community LiteX mixes known issues with their cores with general Q&A. In particular, we found 13 cases of community members raising issues that *would be classified as security-critical*. However, the maintainers or developers contributing to the core did not respond to all of the issues, and even if they did, they did not always clarify whether the problem was with the core itself or whether it was caused by user error or integration issues. As noted in Section 5, we did not address such issues in our survey. Thereby, it stands to reason that the issues reported for LiteX are likely a slight undercount of the true number of issues. Again, we chose to prevent false positives and accepted the risk of false negatives.

7.2. Limited Issue Details

Across all four surveyed vendors, issue descriptions were scarce, often comprising only a single line. This is a stark contrast to, e.g., the CPU errata surveyed by Solt et al. [19]. Especially, input conditions for bugs were often omitted or summarized as *for certain inputs*. Also, issue consequences were sometimes overly generic, e.g., *"intermittent failure"* [78].

The lack of precision in the surveyed data limits our ability to conduct a deeper analysis of root causes and especially vulnerable components, akin to Solt et al. However, in most cases, both expert reviewers were able to agree on the security relevance and the security goal violated by each issue. As pointed out in Section 5, unclear cases were marked as *ambiguous*. Consequently, our findings are likely to be *lower* than the true number of security-critical issues.

7.3. Exploitability and Real-World Impact

Similar to the study of CPU errata by Solt et al. [19], we do not make any claims about *exploitability* or *real-world consequences* of the classified issues. As discussed by the related study, both heavily depend on the device context and usage scenario. However, FPGAs are increasingly used in critical domains, such as automotive [3], space [4], or critical networking infrastructure [99].

Thus, we had to be very conservative in both our assessment of exploitability and the consequences of exploitation: Issues that are benign in one context can be catastrophic in another context. In particular, the aforementioned application scenarios place heavy emphasis on *availability* and *integrity* of the FPGA design, whereas *confidentiality* is not needed. Hence, in these applications, all issues that solely affect *confidentiality* are irrelevant.

However, even issues that seem benign on the surface can have catastrophic effects in some applications. Consider, for example, AMD Issue 54667 for the Ethernet PCS/PMA core [100]: The issue comprises receiving incorrect data after a soft reset of the transceiver. In a general-purpose compute platform, this is a nuisance that can be remedied by a power cycle. However, in an automotive application, the issue can cause a crash-fault of a driver assistance system, impacting availability of that system.

Hence, we did not attempt to determine the exploitability of surveyed issues or score the consequences of exploitation. While this metric would have been interesting to explore, especially in connection with any differences in vulnerability survivability or patterns by vendor or IP type, the available source information simply did not allow for sound analysis. We also lack insights into application scenarios that utilize these vulnerable IP cores, making it impossible to analyze them in our study.

7.4. Limited Information about Root Causes

Similar studies in the software domain [17], [32], [54] often perform root cause analysis of published issues and analyze the fixes themselves in relation to the original software. This analysis can help prevent similar problems from re-occurring in the future and test software more effectively. To this end, the authors often analyze the *commit history* of the surveyed software.

For our study, the overwhelming majority of issues originated from closed-source IP cores. Therefore, we had no access to a version control system and were limited to working with the information provided in the issue reports. In particular, for closed-source IP cores, any fix is often shipped with the next release of the FPGA toolchain. A notable exception to this rule is AMD, which provides patches for the *current* version of Vivado for some issues. For patches incorporated into the next

version of the FPGA tool, however, we had to establish the issue's survivability based on the release dates of the FPGA toolchain. While this accurately describes the minimal wait period for downstream developers to patch their designs, we lack information about how vendors prioritize issues internally and approach fixes.

7.5. Scope of the Study

When designing our study, we chose to ignore hard IP cores, i.e., device-specific primitives. As pointed out in Section 5, hard IP cores need to follow the same development processes as the FPGA device itself. Thereby, we assume that hard IP development is very similar to ASIC development. The ASIC development methodology employed by FPGA vendors may or may not align with the FPGA core development methodology. This question absolutely deserves further study. However, if the methodologies differed, this would reduce the generalizability of our analysis results regarding vulnerability types. Additionally, since hard IPs cannot be patched themselves, there needs to be a different approach to patching, which would muddy our results. On the same note, it is conceivable that some issues in hard IP *cannot be patched* [44]. Thereby, it stands to reason that the reporting rate and strategy for such issues might be completely different than those for soft IP cores. Finally, hard IP cores can only be provided by FPGA vendors themselves, making them incomparable with open-source cores or cores from independent companies.

8. Conclusion and Future Work

Our study did not aim at *comparing* security practices of different vendors or singling out bad practices of individual companies. Instead, our goal was to investigate the overall security posture of the industry. The main takeaways from this study are twofold. First, security vulnerabilities in FPGA IP cores exist. As we have laid out, these vulnerabilities can drastically compromise the security of the overall system while having *no mitigation*. Even if mitigations are available, costs can be prohibitive: For issues that require updates, developers may need to *purchase new licenses* for the updated cores. Even for issues that feature patches or workarounds, a costly *re-synthesis* process is always required, and new FPGA bitstreams must be shipped to affected devices. Furthermore, Intel and AMD deprecate legacy devices in their toolchains, making it impossible to patch designs targeting such devices. For Intel, we found 14 vulnerabilities that affect legacy devices, but whose patches are only available in newer toolchain releases with reduced device support. 11 of these issues provide *no other means of mitigation* as well. For AMD, we did not encounter the same problem.

Second, the challenges we faced during the collection of issues show that it is difficult for any FPGA developer to stay informed about vulnerabilities in the IP cores they use. In particular, vendors do not curate lists of known security issues with their IP cores. Hence, FPGA developers need to assess security criticality themselves.

We would like to conclude our work with a cautionary tale and recommendations to the FPGA industry. Before 2014, OpenSSL was in a similar state as we assess today's FPGA ecosystem to be—there was an issue tracker, but no coherent security policy, no fixed process for dealing with security vulnerabilities, and no release strategy. In April 2014, the infamous *Heartbleed* bug was disclosed, raising serious questions about the security of OpenSSL. Only *after* Heartbleed, the OpenSSL project realized the need for proper vulnerability management [17].

We strongly believe that the FPGA industry should learn from this tale, and we propose the following concrete changes: The current struggles to access vulnerability information could be easily addressed by utilizing an established vulnerability reporting mechanism, such as CVEs; however, according to our findings, this is not being done. This is corroborated by Bellay et al. [44]: The study investigates which vulnerability reporting mechanisms for FPGA and ASIC IP exist and how they are utilized in practice, concluding that very few issues are publicly reported using these mechanisms. We found that all surveyed vendors have taken some steps towards this: AMD, LiteX, and Lattice curate *all* known issues for most of their IP cores in a proprietary database, but do not yet rate the severity of these issues. Intel rates some issues as *critical* (although we are not sure what this rating is based on), but lacks a central list per core.

Additionally, we suggest that vendors reconsider their patching strategy, reorienting themselves to best practices widely accepted in the software industry. First, delivering fixes for critical issues only with new releases of the FPGA toolchain can easily cause a dilemma for developers. *Delaying* the patch prolongs the exposure to attacks that exploit the vulnerability. *Upgrading* to a new version of the toolchain requires the purchase of a license and (in case of non-backwards-compatible changes to tools or IP cores) a considerable amount of work. Thus, we think the industry should strive to provide patches or workarounds for *all versions affected by a security vulnerability*, similar to AMD's existing practice for certain vulnerabilities.

Future research is also needed. Our study excludes hard IP cores, which deserve their own study. At the same time, our study excludes major vendors of IP cores that are suitable for FPGAs and ASICs, notably Cadence and Synopsys. Unlike AMD, Intel, and Lattice, neither vendor provides a publicly available issue database. Cadence and Synopsys, unfortunately, denied us access to any documentation about known issues with their IP.

Proactive Prevention of Harm

All vulnerabilities we discuss were collected from public disclosure documents. All issues have been public for months or even years at the time of writing.

A draft of this manuscript was provided to all vendors two weeks before the submission deadline, and we asked for confirmation of our findings and assumptions. We received no response from Intel and Lattice. The main LiteX maintainer acknowledged our results and did not comment on our findings. AMD reiterated that CVEs are assigned for device security issues (including hard IPs, but not soft IPs) and disclosed according to standard processes and did not comment on our other findings and questions.

At the time of submission, none of the authors is affiliated with any IP vendor discussed in the report or any of their competitors. We declare no conflict of interest.

Acknowledgment

We would like to thank the anonymous reviewers for their critical comments and questions, highlighting areas for improvement in our work. We would also like to thank our anonymous shepherd for their guidance during the revision process.

Our work was funded by zukunft.niedersachsen, the joint science funding program of the Lower Saxony Ministry of Science and Culture and the Volkswagen Foundation.

References

[1] V. Taraate, *ASIC Design and Synthesis: RTL Design Using Verilog*. Singapore: Springer, 2021, ISBN: 978-981-334-641-3. DOI: 10.1007/978-981-33-4642-0. Accessed: Aug. 27, 2024. [Online]. Available: http://link.springer.com/10.1007/978-981-33-4642-0.

[2] *PolarFire® FPGA Ethernet Sensor Bridge*. Accessed: Sep. 22, 2025. [Online]. Available: https://www.microchip.com/en-us/products/fpgas-and-plds/boards-and-kits/ethernet-sensor-bridge.

[3] *Subaru Using AMD Tech for AI-Based EyeSight Safety Systems*. Accessed: Sep. 22, 2025. [Online]. Available: https://www.amd.com/en/resources/case-studies/subaru-eyesight.html.

[4] *NASA Rover Exploring Mars Using AMD Adaptive Computing*. Accessed: Sep. 22, 2025. [Online]. Available: https://www.amd.com/en/resources/case-studies/nasa.html.

[5] Advanced Micro Devices, Inc., *Zynq 7000 SoC Technical Reference Manual*, UG585, Technical Reference Manual, Jun. 2023. Accessed: Sep. 25, 2025. [Online]. Available: https://docs.amd.com/r/en-US/ug585-zynq-7000-SoC-TRM/Register-XUSBPS_EPRDY_OFFSET-Details.

[6] Advanced Micro Devices, Inc., "MIPI D-PHY LogiCORE IP Product Guide," Advanced Micro Devices, Inc., LogiCORE IP Product Guide PG202, Apr. 2024. [Online]. Available: https://docs.amd.com/r/en-US/pg202-mipi-dphy.

[7] Advanced Micro Devices, Inc., "MIPI CSI-2 Receiver Subsystem Product Guide," Advanced Micro Devices, Inc., LogiCORE IP Product Guide PG232, May 2025. [Online]. Available: https://docs.amd.com/r/en-US/pg232-mipi-csi2-rx.

[8] Advanced Micro Devices, Inc., "AXI 1G/2.5G Ethernet Subsystem v7.2," Advanced Micro Devices, Inc., LogiCORE IP Product Guide PG138, May 2024. [Online]. Available: https://docs.amd.com/r/en-US/pg138-axi-ethernet.

[9] Arm Ltd., *AMBA AXI and ACE Protocol Specification*, Oct. 2011. [Online]. Available: https://developer.arm.com/documentation/ihi0022/d/.

[10] Advanced Micro Devices, Inc., *Vivado*, Nov. 2024. [Online]. Available: https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/2024-2.html.

[11] N.-F. Polychronou, P.-H. Thevenon, M. Puys, and V. Beroulle, "A Comprehensive Survey of Attacks without Physical Access Targeting Hardware Vulnerabilities in IoT/IIoT Devices, and Their Detection Mechanisms," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 27, no. 1, 1:1–1:35, Sep. 2021, ISSN: 1084-4309. DOI: 10.1145/3471936. Accessed: Sep. 23, 2025. [Online]. Available: https://doi.org/10.1145/3471936.

[12] W. Hu, C.-H. Chang, A. Sengupta, S. Bhunia, R. Kastner, and H. Li, "An overview of hardware security and trust: Threats, countermeasures, and design tools," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 6, pp. 1010–1038, 2020. DOI: 10.1109/TCAD.2020.3047976. Accessed: Sep. 23, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9310331/.

[13] G. Dessouky et al., "HardFails: Insights into Software-Exploitable Hardware Bugs," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 213–230, ISBN: 978-1-939133-06-9. Accessed: Sep. 23, 2025. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/dessouky.

[14] P. Prinetto and G. Roascio, "Hardware Security, Vulnerabilities, and Attacks: A Comprehensive Taxonomy.," in *ITASEC*, 2020, pp. 177–189. Accessed: Sep. 23, 2025. [Online]. Available: https://core.ac.uk/download/pdf/327177450.pdf.

[15] B. Tan, R. Elnaggar, J. M. Fung, R. Karri, and K. Chakrabarty, "Toward Hardware-Based IP Vulnerability Detection and Post-Deployment Patching in Systems-on-Chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 6, pp. 1158–1171, Jun. 2021, ISSN: 1937-4151. DOI: 10.1109/TCAD.2020.3019772. Accessed: Oct. 16, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9178758.

[16] F. Li and V. Paxson, "A Large-Scale Empirical Study of Security Patches," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, Dallas Texas USA: ACM, Oct. 2017, pp. 2201–2215, ISBN: 978-1-4503-4946-8. DOI: 10.1145/3133956.3134072. Accessed: Sep. 24, 2025. [Online]. Available: https://dl.acm.org/doi/10.1145/3133956.3134072.

[17] J. Walden, "The Impact of a Major Security Event on an Open Source Project: The Case of OpenSSL," in *Proceedings of the 17th International Conference on Mining Software Repositories*, Seoul Republic of Korea: ACM, Jun. 2020, pp. 409–419, ISBN: 978-1-4503-7517-7. DOI: 10.1145/3379597.3387465. Accessed: Sep. 24, 2025. [Online]. Available: https://dl.acm.org/doi/10.1145/3379597.3387465.

[18] E. Iannone, R. Guadagni, F. Ferrucci, A. De Lucia, and F. Palomba, "The secret life of software vulnerabilities: A large-scale empirical study," *IEEE Transactions on Software Engineering*, vol. 49, no. 1, pp. 44–63, 2022. DOI: 10.1109/TSE.2022.3140868. Accessed: Sep. 23, 2025. [Online].

Available: https://ieeexplore.ieee.org/abstract/document/9672730/.

[19] F. Solt, P. Jattke, and K. Razavi, "Rememberr: Leveraging microprocessor errata for design testing and validation," in *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, IEEE, 2022, pp. 1126–1143. DOI: 10.1109/MICRO56248.2022.00081. Accessed: Sep. 23, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9923881/.

[20] S. R. Rajendran, N. F. Dipu, S. Tarek, H. M. Kamali, F. Farahmandi, and M. Tehranipoor, "Exploring the Abyss? Unveiling Systems-on-Chip Hardware Vulnerabilities Beneath Software," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 3914–3926, 2024, ISSN: 1556-6021. DOI: 10.1109/TIFS.2024.3372800. Accessed: Sep. 23, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10458674.

[21] S. Yang, R. Wille, D. Große, and R. Drechler, "Coverage-Driven Stimuli Generation," in *2012 15th Euromicro Conference on Digital System Design*, Sep. 2012, pp. 525–528. DOI: 10.1109/DSD.2012.37. Accessed: Jul. 15, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/6386936.

[22] S. Ahmadi-Pour, M. Logemann, V. Herdt, and R. Drechsler, "Synergistic Verification of Hardware Peripherals through Virtual Prototype Aided Cross-Level Methodology Leveraging Coverage-Guided Fuzzing and Co-Simulation," *Chips*, vol. 2, no. 3, pp. 195–208, Sep. 2023, ISSN: 2674-0729. DOI: 10.3390/chips2030012. Accessed: May 28, 2024. [Online]. Available: https://www.mdpi.com/2674-0729/2/3/12.

[23] C. Spear, *SystemVerilog for Verification: A Guide to Learning the Testbench Language Features*. Springer Science & Business Media, Apr. 2008, ISBN: 978-0-387-76530-3.

[24] M. Teplitsky, A. Metodi, and R. Azaria, "Coverage Driven Distribution of Constrained Random Stimuli," in *Proceedings of the Design and Verification Conference and Exhibition US (DVCon)*, 2015. Accessed: Jul. 15, 2024. [Online]. Available: https://www.semanticscholar.org/paper/Coverage-Driven-Distribution-of-Constrained-Random-Teplitsky-Metodi/d0eaf654fe6f4f80d2f9ddd20083eca037048d7b.

[25] Onur Guzey and L.-C. Wang, "Coverage-directed test generation through automatic constraint extraction," in *2007 IEEE International High Level Design Validation and Test Workshop*, Irvine, CA, USA: IEEE, 2007, pp. 151–158, ISBN: 978-1-4244-1480-2. DOI: 10.1109/HLDVT.2007.4392805. Accessed: Jul. 15, 2024. [Online]. Available: http://ieeexplore.ieee.org/document/4392805/.

[26] D. R. Kuhn, M. S. Raunak, and R. Kacker, "An analysis of vulnerability trends, 2008-2016," in *2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, IEEE, 2017, pp. 587–588. DOI: 10.1109/QRS-C.2017.106. Accessed: Sep. 23, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8004385/.

[27] M. Tang, M. Alazab, and Y. Luo, "Big data for cybersecurity: Vulnerability disclosure trends and dependencies," *IEEE Transactions on Big Data*, vol. 5, no. 3, pp. 317–329, 2017. DOI: 10.1109/TBDATA.2017.2723570. Accessed: Sep. 23, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7968482/.

[28] M. A. Williams, S. Dey, R. C. Barranco, S. M. Naim, M. S. Hossain, and M. Akbar, "Analyzing evolving trends of vulnerabilities in national vulnerability database," in *2018 IEEE International Conference on Big Data (Big Data)*, IEEE, 2018, pp. 3011–3020. DOI: 10.1109/BigData.2018.8622299. Accessed: Sep. 23, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8622299/.

[29] A. Ozment and S. E. Schechter, "Milk or wine: Does software security improve with age?" In *USENIX Security Symposium*, vol. 6, 2006, pp. 10–5555. Accessed: Sep. 24, 2025. [Online]. Available: https://www.usenix.org/legacy/events/sec06/tech/full_papers/ozment/ozment.pdf.

[30] M. Shahzad, M. Z. Shafiq, and A. X. Liu, "A large scale exploratory analysis of software vulnerability life cycles," in *2012 34th International Conference on Software Engineering (ICSE)*, IEEE, 2012, pp. 771–781. DOI: 10.1109/ICSE.2012.6227141. Accessed: Sep. 24, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6227141/.

[31] E. Rescorla, "Is finding security holes a good idea?" *IEEE Security & Privacy*, vol. 3, no. 1, pp. 14–19, 2005. DOI: 10.1109/MSP.2005.17. Accessed: Sep. 24, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1392694/.

[32] D. Lazar, H. Chen, X. Wang, and N. Zeldovich, "Why does cryptographic software fail?: A case study and open problems," in *Proceedings of 5th Asia-Pacific Workshop on Systems*, Beijing China: ACM, Jun. 2014, pp. 1–7, ISBN: 978-1-4503-3024-4. DOI: 10.1145/2637166.2637237. Accessed: Sep. 24, 2025. [Online]. Available: https://dl.acm.org/doi/10.1145/2637166.2637237.

[33] D. R. Thomas, "The Lifetime of Android API Vulnerabilities: Case Study on the JavaScript-to-Java Interface," in *Security Protocols XXIII*, B. Christianson, P. Švenda, V. Matyáš, J. Malcolm, F. Stajano, and J. Anderson, Eds., Cham: Springer International Publishing, 2015, pp. 139–144, ISBN: 978-3-319-26096-9. DOI: 10.1007/978-3-319-26096-9_14.

[34] D. R. Thomas, A. R. Beresford, and A. Rice, "Security Metrics for the Android Ecosystem," in *Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices*, Denver Colorado USA: ACM, Oct. 2015, pp. 87–98, ISBN: 978-1-4503-3819-6. DOI: 10.1145/2808117.2808118. Accessed: Sep. 24, 2025. [Online]. Available: https://dl.acm.org/doi/10.1145/2808117.2808118.

[35] M. Linares-Vásquez, G. Bavota, and C. Escobar-Velásquez, "An empirical study on android-related vulnerabilities," in *2017 Ieee/Acm 14th International Conference on Mining Software Repositories (Msr)*, IEEE, 2017, pp. 2–13. DOI: 10.1109/MSR.2017.60. Accessed: Sep. 23, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7962350/.

[36] M. Jimenez, M. Papadakis, T. F. Bissyandé, and J. Klein, "Profiling android vulnerabilities," in *2016 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, IEEE, 2016, pp. 222–229. DOI: 10.1109/QRS.2016.34. Accessed: Sep. 23, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7589802/.

[37] S. Farhang, M. B. Kirdan, A. Laszka, and J. Grossklags, "An Empirical Study of Android Security Bulletins in Different Vendors," in *Proceedings of The Web Conference 2020*, Taipei Taiwan: ACM, Apr. 2020, pp. 3063–3069, ISBN: 978-1-4503-7023-3. DOI: 10.1145/3366423.3380078. Accessed: Sep. 23, 2025. [Online]. Available: https://dl.acm.org/doi/10.1145/3366423.3380078.

[38] I. A. Chowdhury, E. Ackermann, and S. Bugiel, *Artifacts for the paper "High Impedance: Analysis of Publicly Disclosed Vulnerabilities in FPGA IP Cores (EuroS&P 2026)"*. Apr. 2026. DOI: 10.5281/zenodo.19456164.

[39] S. L. Harris and D. M. Harris, *Digital Design and Computer Architecture*, RISC-V edition. Cambridge: Morgan Kaufmann, 2022, ISBN: 978-0-12-820064-3.

[40] D. Chatterjee, S. Maitra, N. Mishra, S. Shukla, and D. Mukhopadhyay, "Hardware Security in the Connected World," *WIREs Data Mining and Knowledge Discovery*, vol. 15, no. 3, e70034, 2025, ISSN: 1942-4795. DOI: 10.1002/widm.70034. Accessed: Sep. 23, 2025. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.70034.

[41] Z. Pan and P. Mishra, "A survey on hardware vulnerability analysis using machine learning," *IEEE Access*, vol. 10, pp. 49 508–49 527, 2022. Accessed: Sep. 23, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9770796/.

[42] A. Muñoz, "Cracking the Core: Hardware Vulnerabilities in Android Devices Unveiled," *Electronics*, vol. 13, no. 21, p. 4269, Jan. 2024, ISSN: 2079-9292. DOI: 10.3390/electronics13214269. Accessed: Sep. 23, 2025. [Online]. Available: https://www.mdpi.com/2079-9292/13/21/4269.

[43] A. P. Fournaris, L. Pocero Fraile, and O. Koufopavlou, "Exploiting Hardware Vulnerabilities to Attack Embedded System Devices: A Survey of Potent Microarchitectural Attacks," *Electronics*, vol. 6, no. 3, p. 52, Sep. 2017, ISSN: 2079-9292. DOI: 10.3390/electronics6030052. Accessed: Sep. 23, 2025. [Online]. Available: https://www.mdpi.com/2079-9292/6/3/52.

[44] J. Bellay, D. Forte, R. Martin, and C. Taylor, "Hardware vulnerability description, sharing and reporting: Challenges and opportunities," *GOMACTech*, 2021. Accessed: Oct. 17, 2024. [Online]. Available: https://par.nsf.gov/servlets/purl/10237521.

[45] X. Zhang and M. Tehranipoor, "Case study: Detecting hardware Trojans in third-party digital IP cores," in *2011 IEEE International Symposium on Hardware-Oriented Security and Trust*, Jun. 2011, pp. 67–70. DOI: 10.1109/HST.2011.5954998. Accessed: Sep. 23, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5954998.

[46] J. Rajendran, V. Vedula, and R. Karri, "Detecting malicious modifications of data in third-party intellectual property cores," in *Proceedings of the 52nd Annual Design Automation Conference*, ser. DAC '15, New York, NY, USA: Association for Computing Machinery, Jun. 2015, pp. 1–6, ISBN: 978-1-4503-3520-1. DOI: 10.1145/2744769.2744823. Accessed: Oct. 16, 2024. [Online]. Available: https://doi.org/10.1145/2744769.2744823.

[47] M. Zonta-Roudes et al., "eXpect: On the Security Implications of Violations in AXI Implementations," *43rd IEEE/ACM International Conference on Computer-Aided Design (ICCAD 2024)*, 2024. DOI: 10.1145/3676536.3676844. Accessed: Oct. 30, 2024. [Online]. Available: https://n.ethz.ch/~sshivaji/publications/expect_iccad24.pdf.

[48] K. Nienhuis et al., "Rigorous engineering for hardware security: Formal modelling and proof in the CHERI design and implementation process," in *2020 IEEE Symposium on Security and Privacy (SP)*, May 2020, pp. 1003–1020. DOI: 10.1109/SP40000.2020.00055. Accessed: Feb. 2, 2024. [Online]. Available: https://doi.org/10.1109/SP40000.2020.00055.

[49] D. Zhang, Y. Wang, G. E. Suh, and A. C. Myers, "A Hardware Design Language for Timing-Sensitive Information-Flow Security," in *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '15, New York, NY, USA: Association for Computing Machinery, Mar. 2015, pp. 503–516, ISBN: 978-1-4503-2835-7. DOI: 10.1145/2694344.2694372. Accessed: Jan. 28, 2024. [Online]. Available: https://doi.org/10.1145/2694344.2694372.

[50] D. Zhang, A. Askarov, and A. C. Myers, "Language-based control and mitigation of timing channels," *ACM SIGPLAN Notices*, vol. 47, no. 6, pp. 99–110, Jun. 2012, ISSN: 0362-1340. DOI: 10.1145/2345156.2254078. Accessed: Jan. 28, 2024. [Online]. Available: https://doi.org/10.1145/2345156.2254078.

[51] W.-K. Liu, B. Tan, J. M. Fung, R. Karri, and K. Chakrabarty, "Hardware-supported patching of security bugs in hardware IP blocks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 1, pp. 54–67, 2022. Accessed: Oct. 17, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9759459/.

[52] A. T. Markettos, J. Baldwin, R. Bukin, P. G. Neumann, S. W. Moore, and R. N. M. Watson, "Position Paper:Defending Direct Memory Ac-

cess with CHERI Capabilities," in *Proceedings of the 9th International Workshop on Hardware and Architectural Support for Security and Privacy*, ser. HASP '20, New York, NY, USA: Association for Computing Machinery, Oct. 2021, pp. 1–9, ISBN: 978-1-4503-8898-6. DOI: 10.1145/3458903.3458910. Accessed: Mar. 4, 2024. [Online]. Available: https://dl.acm.org/doi/10.1145/3458903.3458910.

[53] E. Ackermann, N. Mauthe, and S. Bugiel, "Work-in-Progress: Northcape: Embedded Real-Time Capability-Based Addressing," in *Workshop on Operating Systems and Virtualization Security*, Vienna, Jul. 2024. DOI: 10.1109/EuroSPW61312.2024.00083. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10628723.

[54] J. Blessing, M. A. Specter, and D. J. Weitzner, *You Really Shouldn't Roll Your Own Crypto: An Empirical Study of Vulnerabilities in Cryptographic Libraries*, Jul. 2021. DOI: 10.48550/arXiv.2107.04940. arXiv: 2107.04940 [cs]. Accessed: Sep. 23, 2025. [Online]. Available: http://arxiv.org/abs/2107.04940.

[55] M. J. Page et al., "The PRISMA 2020 statement: An updated guideline for reporting systematic reviews," *BMJ*, vol. 372, n71, Mar. 2021, ISSN: 1756-1833. DOI: 10.1136/bmj.n71. Accessed: Sep. 29, 2025. [Online]. Available: https://www.bmj.com/content/372/bmj.n71.

[56] MarketsandMarkets, *Field Programmable Gate Array (FPGA) Market Size, Latest Trends, 2024-2029*, Oct. 2025. Accessed: Nov. 6, 2025. [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/fpga-market-194123367.html.

[57] enjoy-digital, *Enjoy-digital/litex*, Sep. 2025. Accessed: Sep. 25, 2025. [Online]. Available: https://github.com/enjoy-digital/litex.

[58] *Pulp-platform*, Sep. 2025. Accessed: Sep. 29, 2025. [Online]. Available: https://github.com/pulp-platform.

[59] *Home :: OpenCores*, Sep. 2025. Accessed: Sep. 29, 2025. [Online]. Available: https://opencores.org/.

[60] *OpenHW Group*, Sep. 2025. Accessed: Sep. 29, 2025. [Online]. Available: https://github.com/openhwgroup.

[61] *Support Keyword Search*, Sep. 2025. Accessed: Sep. 29, 2025. [Online]. Available: https://www.xilinx.com/search/support-keyword-search.html.

[62] *AMD Technical Information Portal*, Sep. 2025. Accessed: Sep. 29, 2025. [Online]. Available: https://www.docs.amd.com/.

[63] *Find Intel® FPGA Intellectual Property (IP) Cores*, Sep. 2025. Accessed: Sep. 29, 2025. [Online]. Available: https://www.intel.com/content/www/us/en/products/programmable/intellectual-property/find-fpga-ip.html.

[64] *FPGA Knowledge Base Articles Search*, Sep. 2025. Accessed: Sep. 29, 2025. [Online]. Available: https://www.intel.com/content/www/us/en/support/programmable/kdb-filter.html.

[65] *Search Results—Altera Documentation and Resources*, Mar. 2026. Accessed: Mar. 16, 2026. [Online]. Available: https://docs.altera.com/search.

[66] *FPGA Documentation Index — Altera*, Sep. 2025. Accessed: Sep. 29, 2025. [Online]. Available: https://www.intel.com/content/www/us/en/support/programmable/support-resources/fpga-documentation-index.html.

[67] *Lattice Semiconductor Solutions Search — Complete Your Design*, Sep. 2025. Accessed: Sep. 29, 2025. [Online]. Available: https://www.latticesemi.com/en/solutionsearch.

[68] *Cores Ecosystem*, Sep. 2025. Accessed: Sep. 29, 2025. [Online]. Available: https://github.com/enjoy-digital/litex/wiki/Cores-Ecosystem.

[69] enjoy-digital, *Enjoy-digital/litescope*, Oct. 2025. Accessed: Oct. 6, 2025. [Online]. Available: https://github.com/enjoy-digital/litescope.

[70] K. Krippendorff, *Content Analysis: An Introduction to Its Methodology*. Sage publications, 2018. Accessed: Nov. 6, 2025. [Online]. Available: https://books.google.com/books?hl=en&lr=&id=nE1aDwAAQBAJ&oi=fnd&pg=PP1&dq=Content+analysis:+An+introduction+to+its+methodology,&ots=y_dgYnkM8z&sig=HWtaCpm10PcX91ReaXuPm-KHCzc.

[71] J. Cohen, "A Coefficient of Agreement for Nominal Scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, Apr. 1960, ISSN: 0013-1644, 1552-3888. DOI: 10.1177/001316446002000104. Accessed: Nov. 6, 2025. [Online]. Available: https://journals.sagepub.com/doi/10.1177/001316446002000104.

[72] P. Jaccard, "Étude comparative de la distribution florale dans une portion des Alpes et du Jura," Imprimerie Corbaz & Comp., 1901. DOI: 10.5169/SEALS-266450. Accessed: Nov. 6, 2025. [Online]. Available: https://www.e-periodica.ch/digbib/view?pid=bsv-002:1901:37::790.

[73] *54690 - 40G/50G Ethernet SubSystem - IP Release Notes and Known Issues for LogiCORE 40G/50G Ethernet Core for Vivado 2016.1 and Forward*, Sep. 2025. Accessed: Sep. 29, 2025. [Online]. Available: https://adaptivesupport.amd.com/s/article/54690?language=en_US.

[74] *75754 - 10G/25G and 40G/50G Ethernet - 2020.2 - GTM - Multi-lane core - Additional GT datapath resets needed after main sys_reset/gt_reset_all*, Sep. 2025. Accessed: Sep. 29, 2025. [Online]. Available: https://adaptivesupport.amd.com/s/article/75754?language=en_US.

[75] Lattice Semiconductor, "DDR Memory Controller IP Core," User Guide FPGA-IPUG-02208-1.6, Jun. 2025. [Online]. Available: https://www.latticesemi.com/view_document?document_id=53685.

[76] *Wrong DQS pattern preamble/postamble · Issue #225 · enjoy-digital/litedram*, Sep. 2025. Accessed: Sep. 29, 2025. [Online]. Available: https://github.com/enjoy-digital/litedram/issues/225.

[77] Intel Corporation, *Incorrect receiver clock usage in 1000Base-X with GXB Transceiver*, Oct. 2012. Accessed: Oct. 21, 2025. [Online]. Avail-

able: https://www.intel.com/content/www/us/en/support/programmable/articles/000086135.html.

[78] Intel Corporation, *Intermittent Ethernet Link Failure and Data Corruption when Enable SGMII Bridge Option is set to OFF*, Sep. 2012. Accessed: Oct. 21, 2025. [Online]. Available: https://www.intel.com/content/www/us/en/support/programmable/articles/000081709.html.

[79] Advanced Micro Devices, Inc., "UltraScale ArchitectureBased FPGAs Memory IP," Advanced Micro Devices, Inc., LogiCORE IP Product Guide PG150, May 2025. [Online]. Available: https://docs.amd.com/viewer/book-attachment/flt3b6Mn8NdXiyvlSsCAgA/7k9FysAWXFALlRYcsCRfJw-flt3b6Mn8NdXiyvlSsCAgA.

[80] Advanced Micro Devices, Inc., "Versal Adaptive SoC Programmable Network on Chip and Integrated Memory Controller," Advanced Micro Devices, Inc., LogiCORE IP Product Guide PG313, May 2025. [Online]. Available: https://docs.amd.com/viewer/book-attachment/RCjUit7DhjogIzIQCcLoFQ/mFHOFwGu0KYtPXDOhhKDXA-RCjUit7DhjogIzIQCcLoFQ.

[81] Advanced Micro Devices, Inc., "DMA/Bridge Subsystem for PCI Express," Advanced Micro Devices, Inc., LogiCORE IP Product Guide PG195, May 2025. [Online]. Available: https://docs.amd.com/viewer/book-attachment/ORWFnEP8p~qn6L1rV1xqBg/Wts5i1CLd2d9XmGUgNImUg-ORWFnEP8p~qn6L1rV1xqBg.

[82] Advanced Micro Devices, Inc., "H.264/H.265 Video Codec Unit v1.2," Advanced Micro Devices, Inc., LogiCORE IP Product Guide PG252, May 2025. [Online]. Available: https://docs.amd.com/r/en-US/pg252-vcu/H.264/H.265-Video-Codec-Unit-v1.2.

[83] Advanced Micro Devices, Inc., "1G/2.5G Ethernet PCS/PMA or SGMII LogiCORE IP," Advanced Micro Devices, Inc., LogiCORE IP Product Guide PG047, Dec. 2025. [Online]. Available: https://docs.amd.com/r/en-US/pg047-gig-eth-pcs-pma?tocId=l0gJzWDV5tw9NKlQSvuX8Q.

[84] Intel Corporation, *Triple-Speed Ethernet FPGA IP*, Oct. 2025. Accessed: Oct. 6, 2025. [Online]. Available: https://www.intel.com/content/www/us/en/products/details/fpga/intellectual-property/interface-protocols/triple-speed-ethernet-mac.html.

[85] Intel Corporation, *DisplayPort Intel® FPGA IP Core*, Oct. 2025. Accessed: Oct. 6, 2025. [Online]. Available: https://www.intel.com/content/www/us/en/products/details/fpga/intellectual-property/interface-protocols/displayport-ip.html.

[86] Altera Corporation, *SDI II FPGA IP*, Mar. 2026. Accessed: Mar. 19, 2026. [Online]. Available: https://www.intel.com/content/www/us/en/products/details/fpga/intellectual-property/interface-protocols/sdi2.html.

[87] Intel Corporation, *Intel® Agilex™ 7 and Intel® Stratix® 10 FPGA E-Tile Hard IP*, Oct. 2025. Accessed: Oct. 6, 2025. [Online]. Available: https://www.intel.com/content/www/us/en/products/details/fpga/intellectual-property/interface-protocols/agilex-e-tile-hard-ip.html.

[88] Altera Corporation, *Low Latency Ethernet 10G MAC FPGA IP*, Mar. 2026. Accessed: Mar. 19, 2026. [Online]. Available: https://www.altera.com/products/ip/po-3084/low-latency-ethernet-10g-mac-fpga-ip.

[89] Lattice Semiconductor, "Tri-Speed Ethernet IP," User Guide FPGA-IPUG-02084-2.3, Jun. 2025. [Online]. Available: https://www.latticesemi.com/view_document?document_id=52476.

[90] Lattice Semiconductor, "10G Ethernet MAC + PHY IP," User Guide FPGA-IPUG-02245-1.6, Jun. 2025. [Online]. Available: https://www.latticesemi.com/view_document?document_id=52691.

[91] Lattice Semiconductor, "AXI4 Interconnect Module," User Guide FPGA-IPUG-02196-1.8, Jun. 2025. [Online]. Available: https://www.latticesemi.com/view_document?document_id=53572.

[92] enjoy-digital, *Enjoy-digital/litedram*, Oct. 2025. Accessed: Oct. 6, 2025. [Online]. Available: https://github.com/enjoy-digital/litedram.

[93] enjoy-digital, *Enjoy-digital/litepcie*, Oct. 2025. Accessed: Oct. 6, 2025. [Online]. Available: https://github.com/enjoy-digital/litepcie.

[94] enjoy-digital, *Enjoy-digital/liteeth*, Oct. 2025. Accessed: Oct. 6, 2025. [Online]. Available: https://github.com/enjoy-digital/liteeth.

[95] enjoy-digital, *Enjoy-digital/litesata*, Sep. 2025. Accessed: Sep. 25, 2025. [Online]. Available: https://github.com/enjoy-digital/litesata.

[96] enjoy-digital, *Enjoy-digital/litesdcard*, Sep. 2025. Accessed: Sep. 25, 2025. [Online]. Available: https://github.com/enjoy-digital/litesdcard.

[97] Intel Corporation, *Intel® Quartus® Prime Lite Edition Design Software*, Mar. 2025. [Online]. Available: https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/2024-2.html.

[98] K. E. Murray, S. Whitty, S. Liu, J. Luu, and V. Betz, "Timing-driven titan: Enabling large benchmarks and exploring the gap between academic and commercial cad," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 8, no. 2, pp. 1–18, 2015.

[99] *Fujitsu uses AMD Zynq™ RFSoCs for power-efficient 5G radios*. Accessed: Mar. 12, 2026. [Online]. Available: https://www.amd.com/en/resources/case-studies/fujitsu.html.

[100] Advanced Micro Devices, Inc., "LogiCORE IP 1G/2.5G Ethernet PCS/PMA or SGMII - Release Notes and Known Issues for Vivado 2013.1 and newer tools," Advanced Micro Devices, Inc., LogiCORE IP Product Guide 54667, May 2024. [Online]. Available: https://docs.amd.com/viewer/book-attachment/RCjUit7DhjogIzIQCcLoFQ/mFHOFwGu0KYtPXDOhhKDXA-RCjUit7DhjogIzIQCcLoFQ.

Appendix A.
Data Availability

We have made our full data set available publicly under a Creative Commons 4.0 Attribution license [38]. This entails the data set itself as a human-readable spreadsheet as well as the automatic crawlers used for retrieving issues and their output. Furthermore, we provide scripts used to calculate statistics. In order to make future reproduction of our results easier, we have also created *snapshots* of the issue descriptions at the time of review in PDF format and included both the descriptions themselves and the script used to make them in the artifact.