Hardware Cost Evaluation in Systems Security

Jesse De Meulemeester
jesse.demeulemeester@kuleuven.be
COSIC, KU Leuven
Leuven, Belgium

Quinten Norga
quinten.norga@kuleuven.be
COSIC, KU Leuven
Leuven, Belgium

Frank Piessens
frank.piessens@kuleuven.be
DistriNet, KU Leuven
Leuven, Belgium

Ingrid Verbauwhede
ingrid.verbauwhede@kuleuven.be
COSIC, KU Leuven
Leuven, Belgium

Marton Bognar
marton.bognar@kuleuven.be
DistriNet, KU Leuven
Leuven, Belgium

ABSTRACT

When proposing new or extended hardware components for integrated circuits, it is common practice to evaluate the associated hardware cost by synthesizing to an FPGA or ASIC and compare it with related work. Ideally, these evaluation figures should provide a reproducible and fair measure of the associated overhead.

In this paper, we focus on the field of systems security and study published hardware designs and extensions of research processors. Through a literature review, we find that many publications lack the necessary details to conduct a fair re-evaluation. To demonstrate how these omissions can result in significant differences in the measured area metrics, we conduct an evaluation case study on extensions proposed for a research processor. Finally, as a first step towards more transparent and consistent evaluation results, we propose a preliminary open-source evaluation toolchain based on best practices from the field of cryptographic hardware. Our results highlight the shortcomings in current evaluation techniques and propose a way forward toward more reproducible and fair hardware cost estimation in systems security.

CCS CONCEPTS

• **Hardware**; • **Security and privacy → Systems security**; • **General and reference → Metrics**;

KEYWORDS

Hardware Evaluation, Replicability

ACM Reference Format:
Jesse De Meulemeester, Quinten Norga, Frank Piessens, Ingrid Verbauwhede, and Marton Bognar. 2025. Hardware Cost Evaluation in Systems Security. In *ACM Conference on Reproducibility and Replicability (ACM REP '25), July 29–31, 2025, Vancouver, BC, Canada.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3736731.3746155

1 INTRODUCTION

Quantifying efficiency and cost is essential when proposing, modifying, or extending hardware designs. Two key dimensions by which new designs are evaluated are performance (e.g., maximal clock frequency, latency, throughput) and hardware cost (e.g., lookup tables (LUTs), flip flops (FFs), gate equivalent (GE)). To obtain these metrics, the designs are synthesized to a field-programmable gate array (FPGA) or application-specific integrated circuit (ASIC), providing figures that can be compared with the state-of-the-art. However, while evaluation metrics are important to gauge the merit of the design, incorrect experimental setups can skew the results or lead to misleading conclusions [7]. For this reason, it is important to follow established evaluation practices and examine evaluation setups when comparing with prior work.

This challenge is not unique to hardware designs. In computer science, performance evaluations for software face a similar issue. Measured performance can be affected by factors such as compiler version and optimization levels, the underlying hardware, or the runtime environment. As a result, papers optimizing for performance typically report compiler settings and the software environment to ensure transparency and reproducibility. One relative advantage, however, is that the metric of interest is usually unambiguous, typically wall-clock time or execution cycles.

The objectivity of hardware cost. In contrast, evaluating the "cost" of hardware designs is considerably more complex and remains an open challenge. Hardware designs are usually described using hardware description languages (HDLs) and are then either simulated, synthesized to an FPGA, or implemented as an ASIC. These implementations can also be optimized for different objectives, such as maximum clock frequency, area, or power consumption, which will impact the other metrics. There even exist tools to discover these different tradeoffs, further illustrating the inherent variability and complexity of this process [5]. Prior work in cryptographic hardware [1, 3] has already acknowledged that cost estimates can vary significantly depending on factors such as the fabrication process, the technology node, and even manufacturer requirements. In essence, even when a hardware design is synthesized for the same platform, comparing hardware costs remains non-trivial.

In this study, we focus on papers that use hardware cost as a generic evaluation metric for a complex design, e.g., in addition to a security proof or software runtime evaluations—papers typical in systems security. Papers whose main contribution is the optimization of a small hardware design for a given platform are out of our scope and usually already use robust evaluation methods. Our main goal is to survey how papers choose to evaluate their hardware cost and how certain factors, such as differences in the

ACM REP '25, July 29–31, 2025, Vancouver, BC, Canada

target platforms and toolchains or omitted implementation details, can lead to variable or unfair comparisons across different designs.

Systems security. In this paper, we focus on the field of systems security, where proposing and implementing hardware changes is a relatively recent phenomenon, partially driven by the emergence of open-source CPU cores such as openMSP430 [6] and BOOM [12]. These cores enable security researchers to model and implement countermeasures or extensions and obtain concrete estimates of performance and hardware overhead. However, as such contributions become more prevalent, ensuring a rigorous methodology becomes important to ensure fair and consistent evaluation.

We start with a literature analysis, comparing a flagship systems security conference with a cryptographic hardware conference in their practices for hardware cost evaluation, finding that papers in the former often lack details necessary for the accurate reproduction and comparison of results. We then experimentally demonstrate in a case study how the omitted evaluation details can lead to significant differences in the obtained evaluation results. Finally, we encourage authors in systems security to apply lessons learned in the hardware security community and show with an open-source workflow how these practices can be applied to our case study.

Contributions. In summary, our contributions are the following:

- We conduct a literature analysis comparing hardware cost evaluation strategies across two research fields, showing deficiencies in the reporting in systems security papers.
- We experimentally show that the evaluation details communicated in systems security for hardware cost are often insufficient and can lead to unfair comparisons.
- We propose a uniform and transparent evaluation strategy moving forward and release a preliminary open-source ASIC-based toolchain to ease adoption.

Our data set and toolchain are archived at https://doi.org/10.48804/WCDDKG. The latest version of the toolchain is available at https://github.com/KULeuven-COSIC/eval-hd.

2 TRENDS IN THE LITERATURE

To understand current hardware evaluation practices in system security research, we surveyed recent papers from USENIX Security, a top-tier systems security conference. We compared these to papers from the most recent edition of CHES, a well-established hardware security conference with a longer history of evaluating hardware designs. The goal is to identify common practices, highlight differences in evaluation between the two research fields, and identify best practices to ensure reproducible results.

Methodology. We manually identified all papers proposing new hardware designs or extensions from the last five editions of USENIX Security (2020-2024) and the most recent edition of CHES (2024) by searching the proceedings for common keywords. We examined each paper and corresponding repository where available, noting the methodology of the evaluation, the details in the description of the evaluation strategy, and the reported metrics. For each paper, we recorded which implementation metrics were reported, both for FPGA (LUT, FF, DSP, BRAM, critical path, etc.) and ASIC (GE/area, critical path, etc.). Besides the raw metrics, we noted the extent to which each paper details its implementation flow. In particular, we tracked whether the paper specifies the toolchain used to synthesize the design, the platform for which it was synthesized, and the toolchain configuration. Here, we expect papers to report the exact synthesis toolchain with version, specific FPGA or ASIC cell library, optimization strategies, and any other relevant settings that were used. Finally, we noted whether the papers re-synthesized prior work using the same flow when comparing implementation metrics and whether the authors released their hardware designs as open source and participated in artifact evaluation.

Table 1: Hardware implementation details reported by papers at USENIX Security (2020–2024) and CHES (2024), including FPGA- and ASIC-specific metrics where available. The table indicates whether each category was fully specified (●), partially specified (◐), or not specified (○).

	Implementation Metrics									Metadata					
	FPGA						ASIC								
Paper	LUT	FF	DSP	BRAM	Crit. Path	Other	GE / Area	Crit. Path	Other	Toolchain	Platform	Config	Re-synth.	Open source	Artifact eval.
2020	●	●	○	○	○	○				○	●	○	●	●	◐
2020	●	●	○	○	○	○				◐	●	○	○	●	◐
2020	◐	○	○	○	○	○	●	○	●	◐	●	○	●	●	◐
2021	●	●	○	●	○	○				○	●	○	–	○	○
2021	●	●	○	○	○	○				◐	◐	○	●	○	○
2021							●	●	○	●	●	◐	●	●	○
2022	●	●	○	○	○	○				◐	●	○	●	●	○
2022	●	○	○	○	●	○				●	●	○	●	●	●
2022							●	●	●	●	●	○	–	○	●
2023	●	●	○	○	○	●				◐	●	○	○	●	●
2023	●	●	○	○	●	○				◐	●	○	●	●	●
2023							●	●	○	●	●	●	●	○	○
2024	●	●	○	○	○	○				●	●	○	○	●	○
2024	●	●	○	○	○	○				●	●	○	●	●	●
CHES 2024	●	●	●	●	●	○				○	●	○	○	●	○
	●	●	○	●	●	○				○	●	○	◐	●	○
	●	●	●	●	●	○				◐	●	○	◐	●	○
	●	●	●	○	○	●				●	●	○	○	○	○
	●	●	●	●	○	○				●	●	○	◐	○	○
	●	●	●	●	●	○				●	●	●	–	○	○
	●	●	●	○	●	●				●	●	●	◐	●	◐
	●	●	○	○	○	○				●	●	●	●	●	○
							●	○	○	○	●	◐	●	○	○
							●	○	○	●	●	○	–	○	○
							●	●	●	●	●	○	●	○	○
							●	○	○	●	●	○	●	●	○
							●	●	●	●	●	●	–	○	○
							●	●	○	●	●	●	●	●	○
							●	○	○	●	●	●	●	●	●
							●	○	○	●	●	●	●	●	○
	●	●	●	●	●	○	●	○	○	◐	◐	○	○	○	○
	●	●	●	●	○	○	●	○	○	◐	●	○	◐	●	●
	●	●	○	○	●	○	●	●	○	●	●	○	◐	●	○
	●	●	○	○	○	○	●	○	○	●	●	●	◐	●	○
	●	●	○	○	○	○	●	●	○	●	●	●	●	●	●

In total, we surveyed 35 papers: 14 presented at USENIX, and 21 at CHES. These papers represent 0.9% of all USENIX and 21% of CHES papers in these years, showing that CHES is a more established venue for hardware modifications. Table 1 summarizes our results.

Reported metrics. In general, we saw that CHES papers more frequently report ASIC implementation results (13/21), while USENIX papers tend to use FPGA synthesis (only 4/14 reporting ASIC). Additionally, among the papers including FPGA results, we noticed that CHES papers tend to report a broader set of metrics on top of the LUT and FF count typically included in USENIX papers.

Reproducibility. There are also systematic differences in reporting the evaluation metadata. Around 70% of CHES papers fully specify the toolchain and target platform, compared to less than half of USENIX papers. Additionally, 9/21 CHES papers describe the toolchain configuration, compared to only one USENIX paper.

An important result from papers proposing or extending designs is the comparison with previous state-of-the-art. To avoid unfair comparisons, prior designs should be re-synthesized or evaluated using the same toolchain and configuration. We found that most papers from both conferences re-synthesize the designs they are comparing to, or indicate that raw numbers are copied in case the source is not available. However, we also found examples of evaluations where the previous results were copied without notice, which is problematic if synthesized for different FPGAs or cell libraries. In Section 3, we will experimentally show how different configurations can lead to greatly diverging comparison results.

Of course, re-evaluating the results from previous papers is only possible when the designs are available as an open-source artifact. While the majority of papers we surveyed did provide the source code for their designs, a significant portion did not; we found that USENIX papers were slightly more likely to include an artifact (71%) compared to CHES (57%), and were also more likely to participate in artifact evaluation (57% compared to only 19% for CHES).

Discussion. Our findings highlight the differences in evaluation practices between systems security and cryptographic hardware research. Notably, CHES papers target ASIC implementations more often than USENIX papers, which, as we will point out in Section 4, is a more suitable target because it suffers less variability compared to FPGA synthesis. A second observation is that toolchain settings are often underspecified, particularly in systems papers. In the next section, we demonstrate how different toolchain configurations can vastly change the implementation results. Finally, we observed that USENIX papers more regularly open-source their designs and participate in artifact evaluation, but observed no strong correlation between this fact and the reported metrics. There is also no obvious historical trend in the five years for these papers.

3 EXPERIMENTAL PROBLEM EXPOSITION

As highlighted in Section 2, systems security papers often omit critical hardware cost evaluation details. In this section, we experimentally demonstrate how FPGA synthesis-related options, such as optimization strategies, can significantly influence the obtained results, making it difficult to reproduce or fairly compare different approaches. These optimization strategies are part of the synthesis toolchains and enable hardware designers to tailor their designs for different use cases. For example, Xilinx Vivado 2024.1, a popular synthesis toolchain used for this case study, offers seven synthesis and thirty implementation strategies, optimized for objectives such as low area, low latency, or low runtime of the synthesis itself.

Table 2: Synthesis results and extension overhead for various optimization strategies (overhead for identical strategies).

Design	LUTs (overhead)			f_{max} [MHz] (change)		
	min	avg	max	min	avg	max
ProSpeCT	19137 (12.7%)	20524 (16.2%)	21131 (19.7%)	32.287 (-4.17%)	32.492 (-0.95%)	33.113 (+1.70%)
AMi-I	2847 (14.2%)	3093 (17.4%)	3197 (20.2%)	58.483 (-3.27%)	58.888 (-0.68%)	60.017 (+2.26%)
AMi-O	19172 (16.0%)	21110 (21.5%)	22383 (26.5%)	32.279 (-3.51%)	32.598 (-0.48%)	33.428 (+2.76%)
Libra	17613 (8.78%)	20359 (13.1%)	23104 (16.5%)	26.752 (-3.81%)	27.067 (-0.07%)	27.880 (+2.52%)

To illustrate the impact of these choices, we perform a case study on extensions to an open-source RISC-V core, a practice common in systems security such as microarchitectural attack and defense research. We select the Proteus [2] core, which has served as the basis for several top-tier systems security publications [4, 9, 10], some of which were published or compared to in the papers we previously analyzed. It is important to note that our aim is not to scrutinize or judge the feasibility of these papers but rather to use their hardware designs to showcase the importance of reproducible evaluations and the problematic trends we identified in Section 2.

Methodology. We synthesized three Proteus extensions, targeting the same FPGA (`XC7A35TICSG324-1L`) as the original papers, synthesizing each design using all 210 strategy combinations. To ensure consistency with the original work, we used the timing constraints reported in the papers. We only report results that meet these constraints, but note that a significant number of strategy combinations (between 29-65% for the different designs) failed them.

To compare the hardware cost of the proposed extensions to the base core, we also synthesize the base core using the same strategies. Since these extensions fork the base design at different versions, re-synthesizing the correct version of the base core under the same conditions is necessary to ensure a meaningful comparison.

Results. Table 2 summarizes the synthesis results in terms of LUTs and maximal clock frequency for the different Proteus extensions, while Appendix B contains further data on FFs and visualizations. These results show that LUT count is particularly sensitive to the optimization strategy. For instance, the overhead in LUTs for Libra nearly doubles when switching the synthesis strategy from `Flow_PerfOptimized_high` to `Flow_AreaMultThresholdDSP`. The flip-flop count and maximal clock frequency, on the other hand, stay relatively constant across the different strategies. None of the implementations used any DSPs or BRAM.

Discussion. These findings highlight the need for detailed reporting of FPGA synthesis settings. Such details are required to ensure reproducible results and fair comparisons between different

Table 3: Area and maximum frequency results with `EVAL-HD`.

Design	Area (overhead)	f_{max} (change)
ProSpeCT	0.219 mm^2 (20.87%)	348 MHz (-9.4%)
AMi-I	0.045 mm^2 (6.53%)	500 MHz (-1.1%)
AMi-O	0.199 mm^2 (10.30%)	360 MHz (-6.3%)
Libra	0.244 mm^2 (6.71%)	381 MHz (-0.0%)

designs. Note that this variability in results is not unique to Proteus and its extensions, but rather stems from the configuration options in synthesis toolchains. Our results show that even when the base and extended designs are synthesized using the same strategy, the measured overhead can fluctuate significantly across different strategies. This problem is exacerbated when results are copied and may thus be synthesized using a different environment or optimization settings. Figure 1 in Appendix B shows the distribution of LUT counts for the extensions using different strategies, showing that comparisons of measurements obtained with inconsistent strategies can lead to very different—even negative—overhead numbers. Addressing this issue requires detailed evaluation reporting and a toolchain that minimizes variability in hardware cost evaluation.

4 AN OPEN-SOURCE EVALUATION STRATEGY

Hardware cost evaluation is inherently complex due to the large design space and variety of target platforms and technologies, while a fair and meaningful cost comparison requires reproducible and consistent results. As shown in Section 3, FPGA synthesis introduces variability due to different optimization settings, making direct comparisons across designs difficult. Furthermore, FPGA synthesis commonly uses proprietary tools that are difficult to automate in a continuous integration (CI) pipeline or an artifact evaluation script.

In this section, we propose an open-source toolflow, `EVAL-HD`, inspired by the trends in the cryptographic hardware community. `EVAL-HD` is a preliminary design, currently focusing on area and critical path measurements, to enable and promote accessible and fair evaluation and comparison of hardware designs in the future. `EVAL-HD` builds on the open-source `yosys` ASIC synthesis framework [11] and targets the widely-used, open-source `nangate-45nm` cell library [8]. We note that it is extensible to other (open-source) cell libraries and technology nodes. However, synthesizing different designs for cost comparison should be done using identical synthesis scripts, compilers, optimization settings, and cell libraries.

The area count is computed by mapping a synthesized netlist to the desired technology and summing the area of cells as listed in the targeted library. Compared to FPGA metrics (LUTs, FFs, DSPs, BRAM, ...), which are further complicated by differences of LUT technologies from different vendors and FPGA families, ASIC synthesis enables vendor- and target-independent evaluation and allows for a singular expression of area cost. This provides a more accurate and easy-to-interpret representation of the real-world cost when targeting a chip. Additionally, the critical path can be determined via the delays between two registers in the synthesized netlist. The delay for each cell is specified in the targeted cell library and enables determining the maximal operating frequency.

Results. We used `EVAL-HD` (with `yosys v0.51`, `nangate-45nm`) to re-synthesize all designs from Section 3, with the results shown in Table 3. Notably, the ASIC area overhead for most designs [9, 10] is lower than the FPGA overhead reported in the original papers. In contrast, ProSpeCT [4] shows a higher area overhead.

Additionally, we determined the maximal operating frequency, derived from the critical path, of all Proteus extensions and the decrease from their baseline. Interestingly, for some designs [9, 10], the critical path is not impacted by the countermeasure, and the maximal operating frequency remains practically unchanged. However, other results [4, 10] divert from those claimed in the original papers. This might be due to the different timing and routing mechanisms in ASIC compared to FPGAs, complicating direct comparisons between the two and motivating further analysis.

Discussion and Future Work. While our toolflow reports the ASIC area cost as area in mm^2, an alternate metric is GEs, which is defined as the amount of 2-input NAND (NAND2) gates required to build the circuit. We use the total area as the primary metric because several tools, including `yosys`, do not support a direct computation of GE. Moreover, these two metrics cannot be directly translated into one another, making the area more accessible.

In future work, we envision our workflow to be extended with other metrics used in the literature, such as power measurements. Additionally, our goal is to integrate a wider variety of open-source cell libraries in the automated toolflow.

5 CONCLUSION

Evaluating the cost of hardware designs in a reproducible and comparable manner is challenging. In this work, we conducted a literature analysis of the systems security and cryptographic hardware communities to identify current practices in these fields. We found that many papers omit crucial details, making the reproduction of results difficult. Additionally, we showed how the omission of such details and the lack of a unified evaluation strategy can lead to inconsistent and even unfair comparisons. To address these issues, we propose a transparent, open-source toolflow for hardware cost evaluation, using ASIC synthesis and standard cell libraries to achieve more consistent results compared to FPGA synthesis. We hope that our analysis and initial `EVAL-HD` toolflow can motivate work towards more reproducible, rigorous, and transparent hardware evaluations in systems security and beyond.

ACKNOWLEDGMENTS

We would like to thank the reviewers for their useful suggestions. This research was partially funded by the ORSHIN project (Horizon Europe grant agreement 101070008), the European Commission through Horizon 2020 (ERC 101020005 BELFORT), the Research Foundation – Flanders (FWO) via grant G081322N, the Research Fund KU Leuven, and the Flemish Cybersecurity Research Program (VOEWICS02). Jesse De Meulemeester is funded by an FWO fellowship (11PFE24N).

REFERENCES

[1] Stéphane Badel, Nilay Dagtekin, Jorge Nakahara Jr., Khaled Ouafi, Nicolas Reffé, Pouyan Sepehrdad, Petr Susil, and Serge Vaudenay. 2010. ARMADILLO: A Multi-purpose Cryptographic Primitive Dedicated to Hardware. In *Cryptographic*

Table 4: Synthesized flip-flop counts and extension overhead for various optimization strategies. The overhead is calculated for identical strategies.

Design	FFs (overhead)		
	min	avg	max
ProSpeCT	12634 (5.93%)	12704 (6.07%)	12757 (6.12%)
AMi-I	3197 (21.5%)	1665 (21.7%)	1772 (21.8%)
AMi-O	22383 (8.54%)	13003 (8.87%)	13135 (9.06%)
Libra	14833 (9.24%)	14894 (9.38%)	14955 (9.47%)

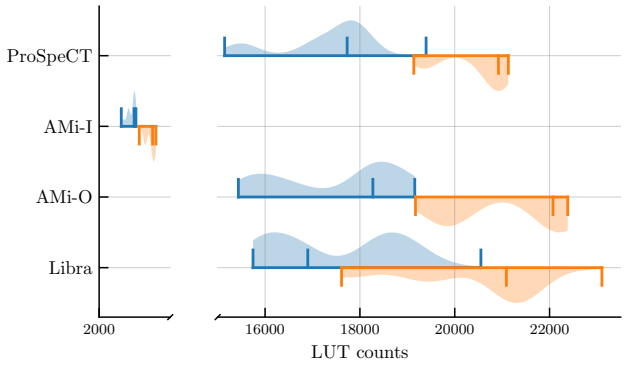

Figure 1: Overview of the obtained LUT counts for the considered Proteus extensions and corresponding base cores.

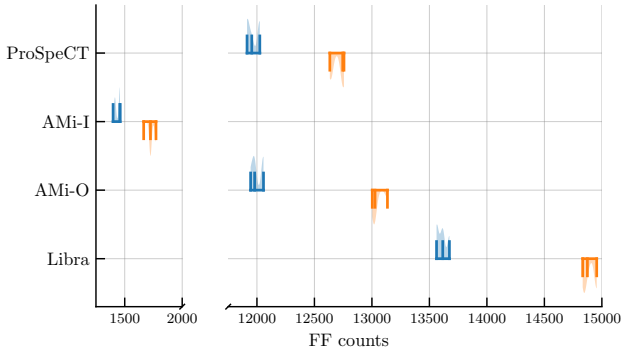

Figure 2: Overview of the obtained FF counts for the considered Proteus extensions and corresponding base cores.

Hardware and Embedded Systems (CHES). https://doi.org/10.1007/978-3-642-15031-9_27

[2] Marton Bognar, Job Noorman, and Frank Piessens. 2023. Proteus: An Extensible RISC-V Core for Hardware Extensions. In *RISC-V Summit Europe '23*.

[3] Christophe De Cannière, Orr Dunkelman, and Miroslav Knezevic. 2009. KATAN and KTANTAN - A Family of Small and Efficient Hardware-Oriented Block Ciphers. In *Cryptographic Hardware and Embedded Systems (CHES)*. https://doi.org/10.1007/978-3-642-04138-9_20

[4] Lesly-Ann Daniel, Marton Bognar, Job Noorman, Sébastien Bardin, Tamara Rezk, and Frank Piessens. 2023. ProSpeCT: Provably Secure Speculation for the Constant-Time Policy. In *USENIX Security Symposium*. https://www.usenix.org/conference/usenixsecurity23/presentation/daniel

[5] Farnoud Farahmand, Ahmed Ferozpuri, William Diehl, and Kris Gaj. 2017. Minerva: Automated hardware optimization tool. In *International Conference on ReConFigurable Computing and FPGAs (ReConFig)*. https://doi.org/10.1109/RECONFIG.2017.8279804

[6] Olivier Girard. 2017. openMSP430. https://github.com/olgirard/openmsp430/blob/master/doc/openMSP430.pdf.

[7] Todd Mytkowicz, Amer Diwan, Matthias Hauswirth, and Peter F. Sweeney. 2009. Producing wrong data without doing anything obviously wrong!. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. https://doi.org/10.1145/1508244.1508275

[8] Christopher Torng. 2019. freepdk-45nm. https://github.com/mflowgen/freepdk-45nm.

[9] Hans Winderix, Marton Bognar, Lesly-Ann Daniel, and Frank Piessens. 2024. Libra: Architectural Support For Principled, Secure And Efficient Balanced Execution On High-End Processors. In *ACM Conference on Computer and Communications Security (CCS)*. https://doi.org/10.1145/3658644.3690319

[10] Hans Winderix, Marton Bognar, Job Noorman, Lesly-Ann Daniel, and Frank Piessens. 2024. Architectural Mimicry: Innovative Instructions to Efficiently Address Control-Flow Leakage in Data-Oblivious Programs. In *IEEE Symposium on Security and Privacy (S&P)*. https://doi.org/10.1109/SP54263.2024.00047

[11] C. Wolf. 2012. Yosys Open SYnthesis Suite. https://yosyshq.net/yosys/.

[12] Jerry Zhao, Ben Korpan, Abraham Gonzalez, and Krste Asanovic. 2020. SonicBOOM: The 3rd Beneration Berkeley Out-of-Order Machine. In *Fourth Workshop on Computer Architecture Research with RISC-V*, Vol. 5. 1–7.

A DATA AVAILABILITY

Our literature survey data set, the code of the designs used in Sections 3 and 4, the scripts to generate these results, and a snapshot of the `EVAL-HD` tool is archived at https://doi.org/10.48804/WCDDKG.

Furthermore, we are continuing the development of the `EVAL-HD` open-source toolflow based on `yosys` and the `nangate-45nm` cell library at https://github.com/KULeuven-COSIC/eval-hd.

B DETAILED COMPARISON RESULTS

This section contains additional evaluation results and visualizations from the case study evaluation in Section 3. First, Table 4 contains the flip-flop counts of the synthesized FPGA designs, showing much less variability than the LUT counts from Table 2. Second, Figure 1 and Figure 2 show a visual representation of the distribution of the obtained LUT and FF counts respectively for all optimization strategies. These figures show that there is no overlap in the obtained FF counts for the different extensions and their respective baselines, while there is a much broader range of results for LUTs with large overlaps.