Security Helper Chiplets: A New Paradigm for Secure Hardware Monitoring

Pooya Aghanoury, *Student Member, IEEE*, Santosh Ghosh, *Member, IEEE*, and Nader Sehatbakhsh, *Graduate Student Member, IEEE*

***Abstract*—Hardware-assisted security features are a powerful tool for safeguarding computing systems against various attacks. However, integrating hardware security features (*HWSFs*) within complex System-on-Chip (SoC) architectures often leads to scalability issues and/or resource competition, impacting metrics such as area and power, ultimately leading to an undesirable trade-off between security and performance. In this study, we propose re-evaluating HWSF design constraints in light of the recent paradigm shift from integrated SoCs to chiplet-based architectures. Specifically, we explore the possibility of leveraging a centralized and versatile security module based on chiplets called *security helper chiplets*. We study the *cost* implications of using such a model by developing a new framework for cost analysis. Our analysis highlights the cost tradeoffs across different design strategies.**

***Index Terms*—Chiplet systems, hardware security monitoring.**

I. Introduction

THE chipletization paradigm shift enables designers to reassess fundamental assumptions about the design and manufacturing of computing systems. This involves reevaluating perspectives on computing, memory, and other critical concepts such as security. While there has been significant focus on addressing various technological and design aspects of chipletization, the impact of chipletization on hardware security remains largely unexplored. Toward this goal, *this study explores the cost implications of chipletization on hardware security services* in modern systems.

With security becoming increasingly paramount, modern systems feature an array of defensive measures, such as malware detection and dynamic information monitoring, to maintain security [1], [2], [3]. In an SoC, security services have to be either embedded within the core or tightly coupled as a coprocessor, all within the same monolithic SoC [1], [4]. This is shown in Fig. 1(a) and (b). A *key insight* is that the recent transition from monolithic SoCs to a chiplet-based architecture presents an opportunity to *reconsider how hardware security should be implemented.* In this paradigm, a system is formed by assembling various smaller "chiplets" instead of developing a monolithic, single-die IC. Each chiplet generally serves a specific function or a designated set of functions and can be manufactured independently. It would, therefore, be possible to design hardware security services as a dedicated and scalable *security chiplet*, and/or integrate specific security features to individual chiplets.

Despite the prospective benefits of a chiplet approach, *the cost overhead of implementing hardware security in a chiplet system should be systematically studied.* Specifically, it is important to understand when it is beneficial to have separate dedicated chiplets (for security, core, memory, I/O, etc.) as opposed to a monolithic system where security services are tightly integrated with other components. More specifically, the cost of bonding and extra packaging is an expense overhead that may make the chiplet systems more expensive at small scales. However, the *key insight* is that commercial enterprises are rarely interested in a single design/solution; they will almost certainly offer a diverse suite of products through a combination of binning and multiple design variants to meet a wide range of market demands [5]. This makes chiplet-based systems more attractive in many scenarios.

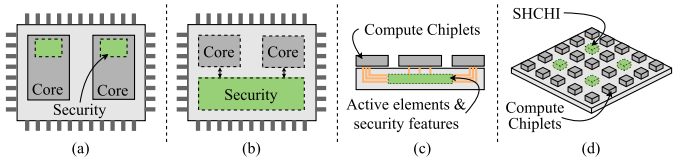

Fig. 1. Existing platforms consist of (a) monolithic SoCs where security features are embedded within the core itself, (b) monolithic SoCs with security implemented as a coprocessor, (c) chiplet-based systems with security embedded in an *active interposer*, and (d) heterogeneous integrated chiplet-based systems with security implemented within distributed utility chiplets. The last two models are SHCHI compatible platforms.

In this paper, we develop a framework for analyzing the cost implications of hardware security features (HWSFs) in the context of a chiplet system. Specifically, we consider three main design strategies: *(a)* a monolithic design where security is tightly integrated with other regular functionalities, *(b)* a distributed chiplet-based security design where the entire design is broken down into smaller chiplets but the security is still tightly integrated into each chiplet, and *(c)* a centralized design where all the security functionalities are implemented in a dedicated security helper chiplet(s) (SHCHI [1]). An overview of different variants is shown in Fig. 1.

We carefully analyze the cost and design challenges of achieving this capability and present our takeaways and insights. In summary, the main contributions of this paper are:

- A cost model to quantify the effective cost of various security solutions in a chiplet system, allowing for a more informed comparison to the current state of the art.
- A systematic evaluation of the advantages and disadvantages of different design strategies on cost and performance for next-generation heterogeneous chiplet systems.

II. Cost Analysis: Is SHCHI is Beneficial?

We formulate a model aimed at estimating the cost associated with implementing SHCHI. We first systemize the security features that commonly exist in modern systems and then develop a cost model for security.

Received 26 October 2024; revised 30 January 2025; accepted 3 February 2025. Date of publication 6 February 2025; date of current version 18 March 2025. This work was supported in part by NSF under Grant CNS-2303115 and Grant CNS-2338623. *(Corresponding author: Pooya Aghanoury.)*

Pooya Aghanoury and Nader Sehatbakhsh are with the School of Engineering, Department of Electrical and Computer Engineering, University of California, Los Angeles, Los Angeles, CA 90095 USA (e-mail: aghanoury@ucla.edu; nsehat@ucla.edu).

Santosh Ghosh is with Intel Labs, Hillsboro, OR 97124 USA (e-mail: santosh.ghosh@intel.com).

Digital Object Identifier 10.1109/LCA.2025.3539282

[1] Pronounced like she.

1556-6056 © 2025 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: UCLA Library. Downloaded on July 02,2025 at 18:19:28 UTC from IEEE Xplore. Restrictions apply.

A. Security Cost Modeling Systemization

There are various hardware security features (HWSF) in modern systems. Some HWSFs protect against software attacks at the architectural level, such as control flow integrity (CFI), information flow tracking (IFT), rowhammer mitigations (RHM), and Spectre and Meltdown mitigations [3], [6], [7]. Others are security-specific hardware modules such as cryptographic accelerators for AES and physical unclonable functions (PUFs) [8].

Our *key observation* is that the associated cost for implementing security can be modeled by categorizing HWSFs into three groups: *area dominant features, I/O dominant features*, and a third group with a moderate need for I/O and area, *hybrid features*. In the following, we explain the rationale for such categorization and describe each category.

Area Dominant Features: This class compromises hardware security solutions with minimal inputs but a significant amount of logic required for computation. Examples include *cryptographic co-processors/accelerators*. These accelerators are usually integrated on-chip and serve multiple cores, as their services are not frequently needed. Therefore, implementing these features is primarily sensitive to area considerations rather than I/O. Moving these accelerators off-chip could introduce additional latency to the cryptographic operations, although they can leverage existing networks and hide the delay through various techniques (e.g., pipelining).

I/O Dominant Features: Another class is security features that mainly rely on I/O for their enforcement. For instance, *hardware monitoring units* are a prime example. Hardware monitoring units are a broad category that encompasses multiple categories of monitoring features that track several hardware activities, such as data buses, performance counters, memory addresses, and more. Examples of defenses include protection against bus fault/probing attacks and/or memory tagging approaches. Typically, they require transferring some additional integrity tags in parallel with actual data in real-time resulting in additional I/Os.

Hybrid: A class in between are methods that need both I/O and logic for enforcing security. Depending on the complexity, either or both I/O and logic might be large. Two prime examples in this category are *(dynamic) information flow tracking (DIFT)* and *hardware malware detectors (HMD)*.

B. Security Cost Formulation

A plethora of research over the past several decades has provided models to estimate the cost of monolithic SoCs and chiplet-based systems [9], [10], [11]. However, to understand the advantage of using SHCHI for cost-effective security, we must first define a cost model for a chiplet-based system. We breakdown our analysis into two main parts: manufacturing and non-recurring engineering (NRE) costs.

Security Chiplet Manufacturing Cost: We observe that there are two possibilities for integrating security in a chiplet system: *distributed* and *centralized*. In the distributed format, security features are integrated into different chiplets (tightly coupled with regular compute) whereas in centralized, ALL security features are integrated into one chiplet (security chiplet). An overview of such designs is shown in Fig. 2.

For monolithic designs, the cost depends on the security area. We define the manufacturing cost of *security* within an SoC/monolithic design as follows:

$$C_{mono,security} = SAR \times \frac{C_{wafer}/N_{chip}}{Y_{chip}}, \tag{1}$$

where SAR is the proportion of area dedicated to security features to the area of the full system (security area ratio). N_{chip} is the maximum number of chips obtainable from a single wafer, and Y_{chip} is the respective yield (defined by the binomial yield model [11] given some chip area A).

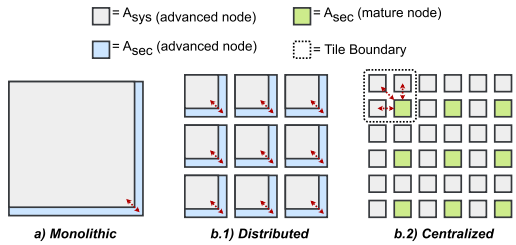

Fig. 2. Possible security integration strategies. Monolithic SoCs (a) and Distributed chiplet systems (b.1) must add security using their respective process node. While centralized chiplet system (b.2) employs tiling where clusters of regular chiplets share a Security chiplet that may use an older process node. A refers to the added area.

For chiplet-based approaches (distributed and/or centralized), two separate costs should be considered: *(i)* per chiplet cost, and *(ii)* integration cost (e.g., to create a 2.5D system). Specifically, the cost of a 2.5D system can be modeled as:

$$C_{2.5D} = C_{int} + \sum_{i=1}^{n} \frac{(C_i + C_{assembly_i})}{Y_{bond_i}^{pc}}, \tag{2}$$

where C_i is the cost of the ith chiplet (security and/or functional), Y_{bond_i} is the bonding cost for the ith chiplet, $C_{assembly}$ is per-chiplet assembly cost, C_{int} is the silicon interposer cost, and pc is the number of I/O pins per chiplet.

Based on Fig. 2, the per-chiplet cost (C_i) for a *distributed* approach is similar to that of $C_{mono,security}$. The only difference is SAR (i.e., the ratio between A_{sys} and A_{sec}) is different for chiplet-based systems (see the difference between Fig. 2(a) and (b.1)).

For a centralized approach, the cost for regular chiplets does not change compared to a non-secure version. However, adding extra centralized security chiplets will increase the total number of dies/chiplets in the system (n in (2)). Furthermore, the security chiplet cost will depend on *(i)* the security area of the security chiplet and *(ii)* its yield.

Specifically, for centralized security chiplets, pc can be modeled as: $SIR \times A_{chiplet,sec}$, where SIR is defined as the Security I/O Ratio (SIR). Our main observation is that the area and I/O parameters are relatively correlated, and SIR can be used to determine the amount of correlation between the two overheads. For I/O-dominant features (see Section II-A), even a small HWSF has large I/O_{sec}, and SIR is quite large (e.g., around 10). Alternatively, for area-dominant features (e.g., a crypto engine) SIR is relatively low (around 0.1) but $A_{chiplet,sec}$ is higher (and the hybrid HWSF has balanced parameters). Furthermore, note that centralized approaches could benefit from a higher yield because they can be manufactured in more mature technologies.

The assembly cost per chiplet, $C_{assembly}$, is also an important factor, particularly in small-scale systems. This cost is a function of various parameters, including the overhead of chiplet handling and placement on the interposer, which involves precise machine-based pick-and-place operations [10].

Lastly, the silicon interposer cost functions, C_{int} (see (2)), as a chip in its own right, adhering to the principles of yield and cost outlined in (1) (without the SAR term). Nevertheless, owing to its absence of active devices and its reliance solely on wiring, it can be produced using a more mature process, resulting in lower wafer costs and defect rates close to zero (i.e., $C_{int} << C_{mono}$) [12].

Chiplet Design Cost: Apart from manufacturing costs, non-recurring (design) engineering costs should also be considered. Specifically, C_{nre}

Authorized licensed use limited to: UCLA Library. Downloaded on July 02,2025 at 18:19:28 UTC from IEEE Xplore. Restrictions apply.

TABLE I
COST MODEL PARAMETERS

Param	Description
C_{nre}	Process dependent base NRE cost
C_{int}	Process dependent cost of a silicon interposer
C_{wafer}	Process dependent cost of a wafer
V_{chip}	Total chip volume for a particular design
u	Portion of unshared NRE costs between designs
r	Non-reuse factor, proportional to chiplet area
k	Number of design variants
Y_{bond}	Chiplet-interposer bonding yield.
Y_{yield}	Process dependent binomial yield for any chip
SAR	Proportion of area dedicated to security

for a given chip/chiplet can be expressed as:

$$C_{nre,tot} \approx \frac{C_{nre}}{V_{chip}} \times (ku + r(1-u)) . \quad (3)$$

The process-dependent base NRE cost, N_{nre}, is amortized over the chip volume, V_{chip}. Parameters k, u, and r are defined as the number of design variants, unshareable portion of NRE between designs, and non-reuse factor proportional to chip area, respectively. To further understand the above parameters, note that in chiplet systems, scalability is achieved through two main factors: the higher yield rate due to smaller-sized chiplets and the *shared* NRE costs. However, chiplet systems cost more to fabricate due to the additional components (e.g., interposer) and bonding costs. The cost advantage becomes apparent when the designer wishes to create multiple variants of a particular design (i.e., k in (3)).

While reusability exists in all three cases, it can be argued that the more specific a chiplet's functionality is, the more *reusable* it will be. For example, an FFT chiplet would be more reusable than an FFT module on a bigger chip [12] hence u in (3) decreases. Furthermore, the shareable portion is also impacted by the (non-)reusability, r, of the design (because the smaller the area, the more likely it is reusable) [12].

The *key insight* is that while chiplet systems incur initial NRE for each chiplet design (e.g., fab costs, masks), each subsequent *system* design would only incur some portion of NRE due to *reuse* and *sharability*. This contrasts with an SoC where each design incurs the full NRE cost. Furthermore, a similar argument can be made for centralized vs. distributed scenarios where the centralized approach can benefit most from reusability because the security and regular functionalities can be designed and implemented *independently*. As a result, the same security chiplet can be reused in different systems (i.e., with different regular chiplets), and/or the same regular chiplets can be reused with a different security chiplet.

III. COST EVALUATIONS

A. Chiplet Vs. SoC NRE Security Costs

To further illustrate NRE and average costs, we examine the total system cost (i.e., the sum of manufacturing and NRE costs) as the number of *design variations* (k) increase. The parameters used are explained in Table I and their respective values are defined based on previous works [10], [12]. In the SoC, we assume that the first design variant is 20 mm^2, and each subsequent variant is an additional 10 mm^2. While in the chiplet-based approach, the first variant is 20 mm^2, and each subsequent variant is an additional 10 mm^2 or two chiplets.

Fig. 3 shows that the total cost spent on developing all variants of a given design becomes more cost-effective for the chiplet-based system when the number of design variants exceeds six. The NRE cost of a

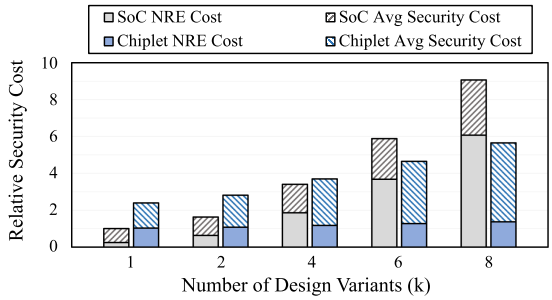

Fig. 3. Change in relative security cost as a function of the number of design variants, normalized to the single variant SoC ($u = 10\%, r = 1$ [12]).

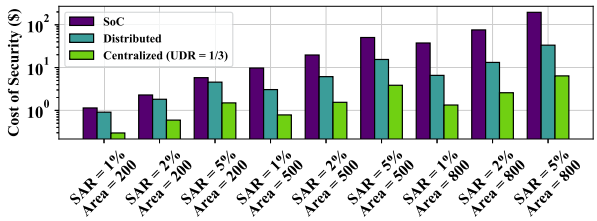

Fig. 4. Cost of security across various platform configurations.

system that utilizes a compute chiplet and a security chiplet incurs a near-constant NRE with only a slight increase due to packaging NRE-related costs. At larger scales, the NRE cost overhead associated with designing a security module in an SoC becomes more dominant over the design cost of an entire chiplet-based system. The diminishing cost of heterogeneous integration not only makes architectures like SHCHI economically viable but positions them as a necessity in the evolving landscape of architecture and hardware security [13].

B. SoC Vs. Distributed Vs. Centralized Cost Analysis

Fig. 4 illustrates nine distinct configurations comparing the cost of security area ratio (SAR) and overall system size across different platforms. Specifically, it uses a 7 nm process for the SoC and distributed platforms, and a 65 nm process for security chiplets in the centralized model. We consider a 1 to 3 security/utility to regular chiplet (UDR) ratio for the centralized model. Numbers are normalized with respect to the SoC cost in the first configuration, and we consider *hybrid* HWSFs. The graph elucidates the trends in security costs as system size expands, and highlights the cost differentials between platforms at each respective system size.

IV. PERFORMANCE ANALYSIS

To model performance as a function of security configuration, we examine the performance using the gem5 simulator on SPEC2017 benchmark suites. We consider four classes of HWSFs. Specifically, we define an HWSF *delay-sensitive* if it is tightly connected to the pipeline. On the other hand, if delays can be concealed through various methods such as speculation, the performance impact is reduced. We call this category *delay-insensitive*. Examples of the former are SpecCFI [3] and STT [14]. For the latter, examples are RHMD [1] and Cyclone [15]. In addition to sensitivity, the frequency of such events is crucial. Lastly, the impact of network latency should be considered. For example, HWSFs that are far away from SHCHI could experience longer delays due to more hopping. We consider this as part of the delay sensitivity.

Consequently, we model sensitive/non-sensitive (N/S) and high/low frequency (H/L) categories. For the baseline model, we consider an

Authorized licensed use limited to: UCLA Library. Downloaded on July 02,2025 at 18:19:28 UTC from IEEE Xplore. Restrictions apply.

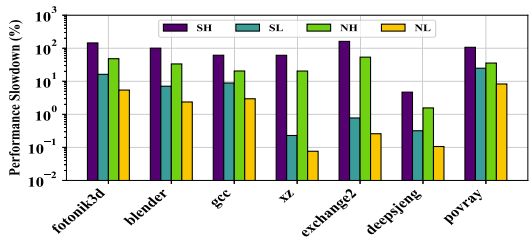

Fig. 5. Per-core performance overhead for various configurations (sensitive vs. non-sensitive and high frequency vs. low frequency.

TABLE II
RELATIVE COST-PERFORMANCE TRADE OFF

Integration Strategy	Security Feature Category					
	IO Dominant		Area Dominant		Hybrid	
	Perf.	Cost	Perf.	Cost	Perf.	Cost
SoC	✓	$$$$	✓	$$$$	✓	$$$$
Distributed	✓	$$$	✗	$$$	✗	$$$
Centralized	✗	$	✓	$	✗	$$
Hybrid	✓	$$	✓	$$	✓	$$

out-of-order system with a private L1 of 16 KB and 64 KB of shared L2 cache. For each class, we inject a delay for each relevant activity. Delays are induced by instrumenting the code and modifying the gem5 simulator to stall when these specific transactions are detected. To model these four variants, we designate the *return* instructions as the *low frequency* variant and the *direct conditional* instructions as the *high frequency* variant. Additionally, we assign 2 cycles as the non-sensitive (small) latency penalty and 8 cycles as the sensitive (large) latency penalty. The results of this model are illustrated in Fig. 5. It should be noted that several benchmarks were excluded from the presentation due to similar results. The observed results suggest that high-frequency events should stay on chip while low-frequency events are good candidates for SHCHI.

V. CONCLUSIONS AND MAIN TAKEAWAYS

Our findings reveal that while the integration of HWSFs into chiplets poses unique challenges, it offers unparalleled opportunities for customization (summarized in Table II).

The trade-offs between performance and cost depend on the type of HWSF and other circuit-level and microarchitectural considerations. These tradeoffs can be optimized by leveraging different design methodologies (e.g., hybrid).

REFERENCES

[1] K. N. Khasawneh, N. Abu-Ghazaleh, D. Ponomarev, and L. Yu, "RHMD: Evasion-resilient hardware malware detectors," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2017, pp. 315–327.
[2] A. Nazari et al., "EDDIE: EM-based detection of deviations in program execution," in *Proc. 44th Annu. Int. Symp. Comput. Architecture*, 2017, pp. 333–346.
[3] E. M. Koruyeh, S. Haji Amin Shirazi, K. N. Khasawneh, C. Song, and N. Abu-Ghazaleh, "SpecCFI: Mitigating spectre attacks using CFI informed speculation," in *Proc. IEEE Symp. Secur. Privacy*, 2020, pp. 39–53.
[4] L. Delshadtehrani et al., "PHMon: A programmable hardware monitor and its security use cases," in *Proc. 29th USENIX Secur. Symp.*, 2020, pp. 807–824.
[5] S. Naffziger et al., "Pioneering chiplet technology and design for the amd epyc TM and ryzenTM processor families: Industrial product," in *Proc. ACM/IEEE 48th Annu. Int. Symp. Comput. Architecture*, 2021, pp. 57–70.
[6] M. Qureshi et al., "Hydra: Enabling low-overhead mitigation of row-hammer at ultra-low thresholds via hybrid tracking," in *Proc. ACM/IEEE 48th Annu. Int. Symp. Comput. Architecture*, 2022, pp. 699–710.
[7] G. E. Suh et al., "Secure program execution via dynamic information flow tracking," *ACM Sigplan Notices*, vol. 39, no. 11, pp. 85–96, 2004.
[8] C. Herder, M. -D. Yu, F. Koushanfar, and S. Devadas, "Physical unclonable functions and applications: A tutorial," *Proc. IEEE*, vol. 102, no. 8, pp. 1126–1141, Aug. 2014.
[9] D. Stow, Y. Xie, T. Siddiqua, and G. H. Loh, "Cost-effective design of scalable high-performance systems using active and passive interposers," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2017, pp. 728–735.
[10] A. Graening et al., "Chiplets: How small is too small," in *Proc. 60th ACM/IEEE Des. Automat. Conf.*, 2023, pp. 1–6.
[11] C. H. Stapper, "Defect density distribution for LSI yield calculations," *IEEE Trans. Electron Devices*, vol. 20, no. 7, pp. 655–657, Jul. 1973.
[12] P. Ehrett, T. Austin, and V. Bertacco, "Chopin: Composing cost-effective custom chips with algorithmic chiplets," in *Proc. IEEE 39th Int. Conf. Comput. Des.*, 2021, pp. 395–399.
[13] Y. Hu et al., "Wafer-scale computing: Advancements, challenges, and future perspectives," *IEEE Circuits Syst. Mag.*, vol. 24, no. 1, pp. 52–81, Firstquarter 2024.
[14] J. Yu, M. Yan, A. Khyzha, A. Morrison, J. Torrellas, and C. W. Fletcher, "Speculative taint tracking (STT) a comprehensive protection for speculatively accessed data," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2019, pp. 954–968.
[15] A. Harris et al., "Cyclone: Detecting contention-based cache information leaks through cyclic interference," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2019, pp. 57–72.

Authorized licensed use limited to: UCLA Library. Downloaded on July 02,2025 at 18:19:28 UTC from IEEE Xplore. Restrictions apply.