2024 Design, Automation & Test in Europe Conference (DATE 2024)

Automated Hardware Security Countermeasure Integration inside High Level Synthesis

Amalia-Artemis Koufopoulou [(1)], Athanasios Papadimitriou [(1)(2)], Mihalis Psarakis [(1)] and David Hély[(3)]
[(1)] *Dept. of Informatics, University of Piraeus, Piraeus, Greece*
[(2)] *Dept. of Digital Systems, University of the Peloponnese, Greece*
[(3)] *Univ. Grenoble Alpes, Leti, F-26000 Grenoble, France*
amykouf@unipi.gr, a.papadimitriou@uop.gr, mpsarak@unipi.gr, david.hely@lcis.grenoble-inp.fr

***Abstract*— High-level Synthesis (HLS) methodology has revolutionized the development of complex hardware designs. It enables the rapid conversion of algorithmic descriptions of functionalities to highly optimized hardware equivalents. While modern HLS tools excel in addressing classic design constraints, such as area, latency and power requirements, they fall short regarding security considerations. Security's role is significantly emphasized in today's digital environment, given the existence of powerful hardware attacks, such as Fault Injection (FI) and Side-Channel Analysis (SCA) attacks. HLS methodology can theoretically facilitate the integration of security measures from the high level, yet its core mechanisms do not actively address the preservation or the improvement of security levels of any countermeasure described. Instead, it may sacrifice security enhancement entirely in circuits of high optimization goals. In this work, first, we propose the automatic countermeasure insertion in a way so that both HLS optimization efforts and the secure addition of the countermeasure are implemented effectively. Secondly, we modify the internal mechanisms of the HLS scheduling algorithm and operation chaining to reduce vulnerable points of the design. We demonstrate our methodology by performing fault injection experiments and comparing the results with a straightforward countermeasure integration technique in terms of hardware security and traditional design metrics.**

Keywords — High-level Synthesis (HLS), Security Countermeasures, Scheduling, Operation Chaining, Google XLS

I. Introduction

High-level Synthesis (HLS) has claimed a significant role among Electronic Design Automation (EDA) tools [1]. Designers can describe complex functionalities in a high-level language (HLL), which is easier to manipulate than register-transfer level (RTL) designs. HLS can significantly assist the integration of security mechanisms in hardware designs, considering the increasing demand for robustness when they operate in untrusted environments where powerful attacks threaten application integrity. Attacks based on Side Channel Analysis (SCA) [2] and fault injections (FI) campaigns [3] need to be considered during the development process so that appropriate mitigation techniques can be applied.

However, recent works on the evaluation of the security of HLS-generated designs [4] have shown that HLS completely disregards security. The HLS flow comprises the translation of the input HLL code into a target agnostic intermediate representation (IR), followed by a series of front-end optimization algorithms (i.e., elimination of redundant instructions). Then, the allocation, scheduling and binding algorithms (back-end) transform the optimized IR to RTL [5].

Excluding the IR-related transformations, any of the algorithms comprising the HLS flow may affect the security properties of a design [6]. For example, redundancy-based countermeasures, such as Double Modular Redundancy (DMR), can be entirely removed due to IR optimizations [7]. Similarly, the scheduling algorithm may parallelize operations that should be executed in different clock cycles to achieve latency minimization [6]. This parallelization step can either result in increased leakage exploited by SCA attacks or render the design more vulnerable to multiple-bit FIs.

This work aims to present a methodology for the successful automated integration of countermeasures against SCA and FI attacks within the HLS flow. The presented HLS flow automatically integrates a DMR countermeasure after front-end optimizations, thus avoiding code elimination. Additionally, it employs security-aware scheduling based on the Force-Directed Scheduling (FDS) algorithm [8], balancing the security cost with traditional area and performance goals (i.e. latency minimization, resource utilization or power usage). The proposed scheduling identifies equivalent (twin) operations and (twin) registers between the two DMR modules and attempts to separate them in the time domain. The minimization of twin operations and registers can reduce the effect of concurrent multiple-bit FI attacks and mitigate the combinational leakage due to parallel computations.

Our methodology is built on top of Google XLS HLS [9], an open-source HLS platform providing the necessary environment to explore the HLS algorithms. As a case study, our approach is applied to a computational SBox implementation [10]. The experimental results show that our methodology enhances the security properties of the cryptographic engine against FI (i.e., reduces the critical error rate more than 30%).

II. Proposed Security Aware HLS Scheduling

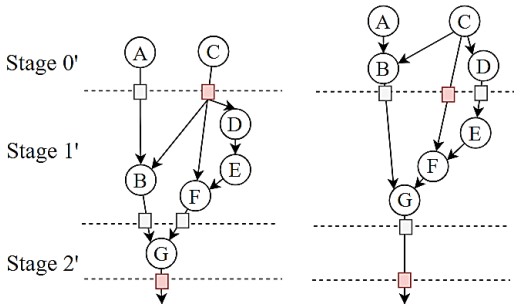
FDS algorithm [8] uses operation mobility (the distance between the earliest and the latest stage an operation can be scheduled) to calculate the operation's probability of being scheduled in a stage. Then, the total contribution of all the possible operations is calculated for each stage. Operations are then scheduled in stages with the least total contribution, with the contributions re-calculated after each placement. FDS achieves a uniform distribution of operations across the stages, which can benefit resource sharing, as well as reduce power consumption for the resulting design [11].

In our approach, we have modified the FDS algorithm to consider twin operation and register overlap (the modified algorithm is called SecureFDS). SecureFDS attempts to avoid scheduling any two duplicated operations of the DMR design in the same stage. By shifting the twin operations, twin registers are also affected, limiting the fault space that could thwart the DMR design. Given the non-overlapping constraint of the twin operations, we proceed by scheduling the first operation of one copy to the stage with the smallest total contribution within its mobility range, as per FDS. For its twin operation on the other copy, the scheduling probabilities will be re-calculated, excluding the stage where the operation of the first copy is placed. In order to guarantee that both DMR copies benefit equally from the uniform distribution, we alternate the application of FDS and its modified version across the two copies.

979-8-3503-4859-0/DATE24/© 2024 EDAA

After SecureFDS, we proceed with the secure operation chaining. Chained operations only store their results in registers when used in succeeding stages. By diversifying the DMR copies operations in the same stage, we also diversify the registers at the end of them. This way, we reduce the chance that the two copies will be concurrently affected in the same way. For our approach, we define a chain path (cp) parameter, i.e., the maximum number of dependent operations allowed in a stage. We explore all chaining combinations for the two modules for which, the first stage contains chain paths of length 1 to cp-1 (cp_0), while the following stages allow chain paths up to cp. We use the combination that results to the least amount of twin operation and register overlaps.

Fig. 1 shows the effects of different chaining schemes. If Fig. 1(a) scheme is applied to one DMR copy and Fig. 1(b) to the other, the number of twin registers (noted with red) would be 2. All the other registers are either eliminated or are not active at the same stage, thus a concurrent double FI attack would not affect them in the same manner. The different chaining schemes contribute further to reducing the overlapped operations, as their number is reduced to 4.



(a) Chained Scheduling ($cp = 3, cp_0 = 1$) *(b) Chained Scheduling ($cp = 3, cp_0 = 2$)*

Figure 1. Chainings for twin operations and register overlap

Finally, we modified the Google XLS flow to accept a custom scheduling file as an input. The custom scheduling file, as well as the DMR version of the optimized IR code are added back to Google XLS flow to generate the RTL design.

III. Evaluation

Our fault model assumes double transient faults, with one bit flip in each DMR replica. To emulate our exhaustive double-bit flip (DBF) model, we first performed a simulated exhaustive single-bit flip (SBF) campaign for each DMR copy, meaning an SBF was injected in all flip-flops (FFs) in every clock cycle of the execution. Then, we examined all the combinations of the SBFs injected in the two copies at the same clock cycle. We defined as Critical Errors the cases when the DBF causes both DMR copies to produce the same erroneous output, i.e., the DMR scheme fails to detect the error.

In order to demonstrate our methodology, we used an HLL algorithm performing the AES SubBytes function in a computational manner [10]. We present results comparing the use of a) FDS and the same chaining scheme on both DMR copies and b) SecureFDS with the diverse chaining schemes. Table I demonstrates the security levels of these two approaches against DBF attacks. The absolute numbers of critical errors are significantly reduced for all cases when SecureFDS is used. The improvement of Critical Error Rate (CER) metric ranges from 6.74% for cp=1 to 32.89% for cp= 8. Concerning the latency (max clock frequency) metric, we note a performance degradation when SecureFDS is applied, especially as the cp number increases. Similarly, for area utilization metrics (FF & LUT), we observe that the SecureFDS impose a small overhead in some cases. Both metrics highlight the loss in efficiency of SecureFDS compared to FDS.

Table I. Comparison of FDS and SecureFDS with different chaining schemes regarding security & traditional metrics

cp	Scheduling	Critical Errors	CER (%)	Max Clock Frequency (MHz)	FFs	LUTs
1	*FDS*	1371	0.0013	526	3652	638
1	*SecureFDS*	1287	0.0012	474	3534	575
3	*FDS*	427	0.0097	429	1300	292
3	*SecureFDS*	303	0.0067	339	1241	358
8	*FDS*	164	0.0703	341	514	204
8	*SecureFDS*	77	0.0472	283	541	223

IV. Conclusions

In this work, we present a methodology that automatically integrates a DMR countermeasure in the HLS flow and modifies the internal mechanisms appropriately so that the countermeasure's effect is not only retained but is even improved. The proposed scheduling approach considers the security principle of operation separation. This way, it reduces the overlap of twin operations and registers, i.e., elements concurrently used across the DMR copies. Our methodology significantly improves the security of the designs against FI attacks with minimal time and area overheads.

Acknowledgment

This research has been financed by the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 895937.

References

[1] Cong, Jason, et al. "High-level synthesis for FPGAs: From prototyping to deployment." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 30.4 (2011): 473-491.

[2] Kocher, Paul, Joshua Jaffe, and Benjamin Jun. "Differential power analysis." *Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19*. Springer Berlin Heidelberg, 1999.

[3] Bar-El, Hagai, et al. "The sorcerer's apprentice guide to fault attacks." *Proceedings of the IEEE* 94.2 (2006): 370-382.

[4] Koufopoulou, Amalia-Artemis, et al. "Security and Reliability Evaluation of Countermeasures implemented using High-Level Synthesis." *2022 IEEE 28th International Symposium on On-Line Testing and Robust System Design (IOLTS)*. IEEE, 2022.

[5] Shi, Shang, et al. "SecHLS: Enabling Security Awareness in High-Level Synthesis." *Proceedings of the 28th Asia and South Pacific Design Automation Conference*. 2023.

[6] Pundir, Nitin, et al. "Secure high-level synthesis: Challenges and solutions." *2021 22nd International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2021.

[7] dos Santos, André Flores, et al. "Applying TMR in hardware accelerators generated by high-level synthesis design flow for mitigating multiple bit upsets in SRAM-based FPGAs." *Applied Reconfigurable Computing: 13th International Symposium, ARC 2017, Delft, The Netherlands, April 3-7, 2017*.

[8] Paulin, Pierre G., and John P. Knight. "Force-directed scheduling for the behavioral synthesis of ASICs." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 8.6 (1989): 661-679.

[9] Google. "XLS: Accelerated HW Synthesis." GitHub, 2020, https://github.com/google/xls/. Accessed July 2023.

[10] Canright, David. "A very compact S-box for AES." *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, Berlin, Heidelberg, 2005.

[11] Katkoori, Srinivas, and Ranga Vemuri. "Scheduling for low power under resource and latency constraints." *2000 IEEE International Symposium on Circuits and Systems (ISCAS)*. Vol. 2. IEEE, 2000.