MemSecure: Enabling Orders of Magnitude Reduction in Embedded Memory Certification and Trust Effort

Kartik Mohanram [†] ***and Sundar Iyer*** [‡]

[†]Department of Electrical and Computer Engineering, University of Pittsburgh, PA, USA
[‡]Memoir Systems, Santa Clara, CA, USA
kmram@pitt.edu sundaes@memoir-systems.com

Abstract: *MemSecure is a general framework for the realization of hundreds of embedded memories in modern SoCs from a small set of fundamental macros. It lays the theoretical and practical foundation to simplify — by one or more orders of magnitude — the certification and trust problem for embedded memory.*

Keywords: Embedded Memory; Compiler; Security; Trust.

Summary

Security concerns have made Trusted Electronics an important and challenging field of research for SoC Designs [1]. Embedded Memory (a key component of SoCs [2]), dominates SoC design in the number of unique instances, the die-area consumed, and in its obvious importance, i.e., embedded memory stores valuable data that needs to be tamper-resistant and secure.

The sheer number of required configurations that embedded memories need to support imply that they are built using advanced physical memory compilers, which can generate any unique configuration [3–6]. The heavy use of compiled memories poses multiple challenges on the certification and trust fronts in practice. This is because standard design-for-hardware-trust solutions [7], which can be used to certify and trust circuits, find it difficult to handle the large number of different memory configurations and the customization effort that is required for each instance. This motivates solutions that can reduce the overall number of memories that need to be built through the identification of a reduced set of 'fundamental' memory macros, which can be used one or more times to realize all memory instances in the design.

This paper describes MemSecure, a fast, general method based upon the classical change-making algorithm for the identification of such fundamental memory macros. MemSecure enables a significant reduction in the number of fundamental macros at negligible area cost. MemSecure was evaluated on single-port memory macros, which are by far the most commonly used embedded memory in SoCs. Our results show that MemSecure can provide 4–8 orders of magnitude reduction in the size of the search space of a memory complier, and converge to fundamental sets that are 5.4–16× smaller than the original set for 2.1–4.7% wasted bits and comparable power and performance. Further, when MemSecure is combined with Algorithmic Memory Technology (a technique to mimic multi-port memories using single-port memories [8, 9]), it makes the approach truly universal and exhaustive. The fundamental memory macros are now capable of realizing single-port and multi-port memory configurations on any complex SoC, with over 110× reduction in the original set, representing two orders of magnitude simplification in the ability to certify and trust memory.

Background and Methods

Defense, military, and strategic security concerns have made Trusted Electronics an important and challenging field of research for SoC Designs [1]. This is especially true for Embedded Memory, one of the most critical and widely used components of modern SoCs. The number of on-die memory instances per SoC is increasing rapidly, and it is projected that embedded memories will occupy over 70% of the die-area in future SoCs [2].

Memory compilers are CAD tools that can quickly generate hundreds or even thousands of unique memories differing in depth and width, organization (banks, ports, etc.), density, performance, and power across SRAM and embedded DRAM technologies [3–6]. Unfortunately, this only makes the job of certification and trust of embedded memory harder. Collectively dubbed 'design-for-hardware-trust' [7], solutions that help prevent the insertion of Trojans, facilitate easier detection of Trojans, and provide effective IC authentication are difficult to automate on a large scale, simply due to the vast number of memory configurations that require intensive customization to achieve desired levels of certification and trust.

In this paper, we describe MemSecure, a methodology to systematically reduce the number of distinct memory macros that are used by a memory compiler to realize all the memory instances in a design. It is motivated by the premise that if all the memory instances in an SoC design can be realized from a reduced set of 'fundamental' blocks, the burden of certification and trust can be reduced significantly. To the best of our knowledge, this is the first attempt to propose a systematic scalable solution to this increasingly important problem in trusted electronics. An additional benefit is that by relying on a small number of unique memory blocks that can be intensively characterized, a corresponding increase in field reliability can also be achieved.

For a set of memory instances M, each instance $m \in M$ is given by the product $d \times w$ where d is its depth and w is its width. MemSecure seeks a reduced set M' (note that it is not necessary that $M' \subset M$) from which all $m \in M$ can be realized through two fundamental compiler operations: multiplexing and instantiation. Multiplexing is the operation of compiling larger memory depths using one or more macros of equal or smaller depth, whereas instantiation is the operation of compiling wider macros using one or more macros of equal or smaller width. Typically, compilers generate the m by interpolating across a range of pre-characterized memory macros, and adopt a greedy strategy (e.g., largest macro first) during compilation. More generally, memory compilation is a two-dimensional combinatorial problem within the general class of $\mathcal{NP}$-hard knapsack and bin-packing problems. When the depth

Table 1: Results for fundamental set minimization

Design Specifications				Fundamental Set		
Design	Macros	Depths	Widths	Fundamental Macros	Reduction	Waste
Industry 1	113	33	77	21	5.4×	2.1
Industry 2	181	58	89	29	6.2×	4.0
Industry 3	313	85	136	36	8.7×	4.7
Syn 1	305	75	66	31	9.8×	3.6
Syn 2	335	74	87	33	10.2×	4.5
Syn 3	483	83	181	39	11.2×	3.7
Syn 4	577	84	108	40	16.0×	3.6

and width dimensions are decoupled, it can be shown that optimal solutions in each dimension can be obtained through the classical change-making ($\mathcal{CM}$) algorithm, a knapsack-type problem.

Classical $\mathcal{CM}$ requires a set of denominations and provides perfect packings. Although this is useful when perfect packings are desirable, it can result in excessive fragmentation unless a large range of denominations (i.e., fundamental memory macros) are available. However, since our objective is to reduce the number of fundamental memory macros, we relax the need for exact realization of memories within MemSecure. For example, along the depth dimension, such a relaxation can yield packings of the form 63K = [32K,32K] where an exact packer would yield 63K = [7K,8K,16K,32K]. The core algorithms in MemSecure extend classical $\mathcal{CM}$ to explicitly tradeoff area, i.e., memory capacity for reduction in the number of fundamental memory macros required by the memory compiler. MemSecure also controls the extent of multiplexing and instantiation during the composition process to ensure that the critical path delay is not impacted by multiplexing during depth packing and that loading on the address lines is not increased due to memory fragmentation during width packing.

The core methods in MemSecure decouple the depth and width dimensions, i.e., MemSecure first processes the set $D = \{d \mid d \times w \in M\}$ to determine a set of fundamental depth macros D' that can be used to realize all depths (irrespective of width) in M. In the second phase, for each $d' \in D'$, MemSecure consolidates the unique widths across all depths from D and invokes a pass of the core packing algorithm. Thus, for every $d' \in D'$, MemSecure obtains a reduced set of widths $W'_{d'}$ by applying the $\mathcal{CM}$-based algorithm. The final set of fundamental macros M' is given by $\bigcup_{d' \in D'} \{d'\} \times W'_{d'}$.

Results

We validate the effectiveness of MemSecure on data from real-world designs, and also on synthetic data generated to model this real-world data from small, medium, and large designs. In particular, we apply MemSecure to identify fundamental sets for single-port embedded memory, which is by far the most commonly used memory in SoCs today. Next, we combine MemSecure and Algorithmic Memory Technology (a technique to realize 2-port, dual-port and multi-port memory configurations from single-port memories [8,9]) to show how embedded memories with any port configuration can be realized using fundamental macros. This ensures that our approach is truly universal and exhaustive, and is applicable across the entire gamut of embedded memory.

We present results on these datasets for our main objective, which prioritizes fundamental set minimization at the expense of wasted bits in Table 1. The first dataset comprises three cases based on sanitized industry designs from high volume and leading SoC vendors. In order to demonstrate that MemSecure is effective across a wide class of SoCs, statistical analysis of the first dataset and knowledge of the behavior of memory compilers was used to generate synthetic test sets with the freedom to test the method across a design space that we expect to encounter in practice.

The first four columns of the table give details about the test cases that were considered for the experiments. The first column specifies the source of the design (industry or synthetic). The second column enumerates the total number of unique memory macros that were used in the design. Note that each macro may have more than one instance, but that data is not reported here. The third and fourth columns report the number of unique memory depths and the number of unique memory widths in the design, respectively.

Columns 5–7 report results when muxing limit was set to 2/4/8 and instantiation limit was also set to 2/4/8. Note that 2/4/8 means that the first pass for each packing is with a limit of 2; if the penalties are excessive, a second pass is made with a limit of 4, with a final pass with a limit of 8 if optimization with 4 is not effective. These options ensure that MemSecure determines the fundamental macro sets without resulting in excessive overhead by way of muxing on the critical paths or instantiation to achieve larger widths. Column 5 reports the size of the fundamental set M'; column 6 reports the reduction in terms of the size of the original set M; column 7 reports the waste as a %age of the total memory when the fundamental set is used to realize all memory instances in that test case. As seen from the table of results, MemSecure successfully provides 5.4–16× reduction in the size of the fundamental set for less than 5% wasted bits in the worst case. Note also that such a fundamental set provides a 4–8 orders of magnitude reduction in the size of the search space of a modern memory complier, facilitating quicker convergence in memory compilation and optimization.

MemSecure was also combined with Algorithmic Memory Technology [8, 9]. Since Algorithmic Memory Technology can realize multi-port memories from single-port memories, MemSecure can certify and trust multi-port memories without needing to create fundamental memory macros for 2-port, dual port, and other multi-port physical memories. For a synthetic library (which had single-port, 2-port, dual-port and 4-port memories), the reduction was over 110×, which is roughly 2 orders of magnitude reduction with less than 5% wasted bits.

Conclusions

We have addressed a problem inherent to the certification and trust of embedded memory macros that was considered infeasible before, i.e., the proliferation of embedded memory configurations and the resulting challenges in coming up with techniques to certify and trust these embedded memories in modern SoCs.

Our proposed framework MemSecure demonstrates how hundreds of memory instances in a design can be realized from a much smaller set of 'fundamental' memory macros. MemSecure realizes these fundamental macros in reasonable execution time (tens of seconds on an average desktop computer) and leverages existing memory compilers to build all the memory instances based on the fundamental sets. MemSecure is applied to single port embedded memory (by far the most common memory in SoCs today), and when combined with Algorithmic Memory Technology, it is also capable of realizing multi-port memories. MemSecure is currently under integration into a memory compiler platform.

We believe our work lays a strong theoretical and practical foundation to simplify — by one or more orders of magnitude — the trusted electronics problem for embedded memory, and allows architects to focus on the certification and trust of a small manageable number of memory instances.

References

[1] J. Lieberman, "National Security Aspects of the Global Migration of the U.S. Semiconductor Industry," 2003.

[2] K. Darbinyan, G. Harutyunyan, S. Shoukourian, V. Vardanian, and Y. Zorian, "A robust solution for embedded memory test and repair," in *Proc. Asian Test Symposium*, pp. 461–462, 2011.

[3] R. E. Matick and S. E. Schuster, "Logic-based eDRAM: Origins and rationale for use," *IBM Journal of Research and Development*, vol. 49, no. 1, pp. 145–165, 2005.

[4] K. Zhang, *Embedded Memories for Nano-Scale VLSIs*. Series on Integrated Circuits and Systems, Springer, 2009.

[5] "ARM Embedded Memory IP: http://www.arm.com/products/physical-ip/embedded-memory-ip/index.php."

[6] "Synopsys DesignWare Embedded Memories and Logic Libraries: http://www.synopsys.com/IP/SRAMandLibraries/Pages/default.aspx."

[7] M. Tehranipoor and F. Koushanfar, "A survey of hardware Trojan taxonomy and detection," *IEEE Design and Test of Computers*, vol. 27, no. 1, pp. 10–25, 2010.

[8] "Memoir Systems: http://www.memoir-systems.com."

[9] "Breaking through the embedded memory bottleneck http://eetimes.com/design/memory-design/4391378/Breaking-through-the-embedded-memory-bottleneck–part-1."