ECE 5770 : Resilient Computer Systems Modeling On-Die Termination Power Overhead of Encrypted Data

Udit Gupta, Monica J. Lin
School of Electrical and Computer Engineering, Cornell University, Ithaca, NY
ug28@cornell.edu, mjl256@cornell.edu

Abstract

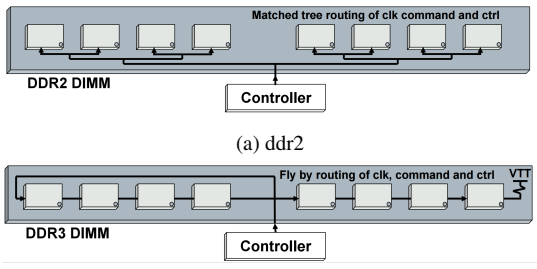
1. Introduction

Since the advent of the Internet, compute devices have become not only more prolific but also varied - encompassing not only high performance compute clusters but also general purpose computers and mobile systems. Moreover, the assumptions made about the security model of a system has changed drastically; users can no longer be trusted, and devices have the potential to interact with unknown and possibly untrustworthy hosts while communicating sensitive information. For example, the boom in smartphones has caused a secondary explosion in the mobile application industry, allowing users to log in to various trusted systems (e.g. bank accounts, medical records and shopping accounts) remotely.

However, the sudden increase in connectivity exposes users to adversaries that may attempt to leverage these interactions in order to obtain confidential information or disrupt the use of services or applications.

In response to these changes, there has been an increasing effort in developing trusted computing platforms augmented with specialized hardware modules that provide security features such as authentication and decryption/encryption. One such security feature that remains a focus in trusted computing research is protecting off-chip memory. Protecting off-chip memory includes maintaining confidentiality of private data. Challenges in designing memory protection mechanisms involves providing encryption primitives at a low cost (money, area, power and design complexity), high throughput and low latency without compromising security. Current work also focuses on memory protection schemes for general purpose and high performance computing systems. The domain of memory protection in the mobile and embedded computing domains is relatively less studied and poses interesting challenges. Mobile and embedded devices often operate under strict power and area constraints. Providing secure memory protection while following the prescribed power and area constraints as well as maintaining performance of the computing devices is an ongoing challenge.

1. Characterizing the power overhead of the memory system is still an open problem - especially in the context of encryption.
2. We use first order approximations to model the power overhead of
3. On die termination - ODT encrypting data on the memory system.
4. We verify that encrypted data has a significant power overhead [add numbers]
5. Outline the paper's sections



(a) ddr2

(b) ddr3

Figure 1: DDR DRAM Chip Topologies [2]

2. Problem Formulation

Figure 2 illustrates the general secure memory encryption architecture used in this study. The threat model assumes that the on-chip memory, `data-cache`, is secure and tamper-proof whereas the off-chip memory is not. The `encryption core` between the data-cache is supposed to provide cryptographic confidentiality properties that prevent unauthorized principals from reading sensitive information that user's wish to keep secret.

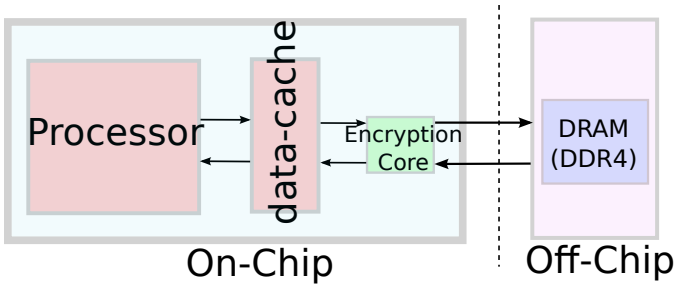


Figure 2: General Memory Encryption Architecture: All on-chip memory accesses, between the processor and data-cache, are performed in plaintext whereas all off-chip, between the data-cache and DRAM, are encrypted.

Typically, the encryption core seen in Figure 2 implements the `AES-CTR` mode scheme. Where previous studies have focused on the performance and area overhead of memory encryption, we wish to explore the impact of memory encryption on the power consumption of off-chip memory storage systems. Specifically we want use various computer architecture analysis tools to simulate memory access traces and model the power overhead of encryption on DDR4 memory technology. Our first order model characterizes the `ODT` power overhead when using `Data-Bus Inversion` for realisitic benchmarks targetting mobile systems using DDR4 memory technology.

3. Methodology

As mentioned earlier, DRAM power consumption is dominated by charging/discharging chip capacitances and signal reflections caused by link terminations. Once the capacitors have reached steady state, their power consumption is negligible in comparison to the system as a whole and thus can be neglected from our power models. Instead we focus on link termination and signal reflections. Specifically, DDR4 memory technology leverages `DBI` to reduce the signal reflections and power consumption. Intuitively, DBI aims to reduce the number of bit flips (DBI-AC) and binary signal probabilities (DBI-DC) to improve power consumption of read and write operations. For reducing bit flips, take for example the case where memory location M previously had the value 00000000 that was being overwritten by the value 11111111. Instead of flipping every bit in M, a DBI-AC enabled memory system would retain the previous state of 00000000 and set the DBI flag that signifies that all the bits are flipped. This would incur the power consumption of 1 bit flip, not the entire data byte. Since the power consumption of 0 and 1 is not the same, depending on memory technology constraints, DBI-DC aims to reduce the prevalence of the higher power consuming bit. In the case of DDR4, 0 consumes more power and thus DBI aims to reduce the number of 0's and increase the number of 1's [add citation]. Figure 3 illustrates the impact of DBI-DC on improving signal quality and consequently power consumption of read and write operations. The larger open data eye illustrates that there is less interference between the driver and reflected signals when using DBI-DC and the smaller V_{pp} suggests that the driver has to work less with DBI-DC enabled.

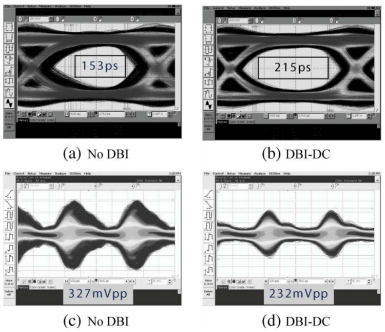


Figure 3: Using DBI-DC increases the open data eye and reduces the required V_{pp} for effectively driving the DRAM chip. Taken from [4]

The following sections outline our model, and experimental setup for measuring the ODT power overhead of memory encryption on DBI-enabled DDR4 memory technology.

3.1 Model

In order to model the power overhead of memory encryption on DBI-enabled memory technologies, we use the same model described by Hollis [4], shown below:

$$P_t = A \times P_{dc} + B \times P_{ac}$$

Where P_t is the ODT power consumption, P_{dc} is the intrinsic DC-power consumed and P_{ac} is the intrinsic AC-power consumed by bit flips. The ratios A and B determine the DC and AC factors of P_t respectively. DBI-DC aims to exploit program structure and reduce A whereas DBI-AC aims to reduce B. Together they can significantly decrease the ODT power consumption. As Hollis [4] describes - under the assumption that data written to memory or read from memory are completely random:

$$A = 0.5$$

$$B = 0.5$$

Intuitively, truly random data would have equal probability of each binary digit making $A = 0.5$. Similarly, truly random data would have equal probability of flipping or not flipping and thus, $B = 0.5$. In order to model ODT power consumption, we must determine the values of A and B for secure memory systems with encryption and insecure memory systems without encryption. The following sections outline our experimental setup for analyzing the DBI enabled power model described.

3.2 Experimental Setup

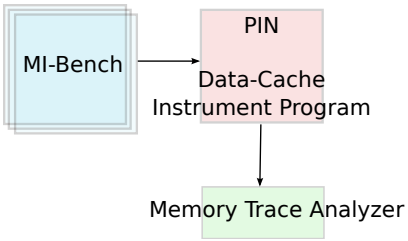


Figure 4: General system architecture for the experimental setup for analyzing the DBI enabled power model

Figure 4 illustrates the general experimental setup we used to implement the DBI model described earlier. First, we need a set of representative benchmarks to run through to generate realistic memory traces that can be used to analyze the effect of memory encryption on power consumption. We chose MiBench [3] for our benchmark suite. MiBench consists of applications representative of embedded and mobile workloads and provides easy to use build binary executables which integrates well with the `PIN` architectural simulator we used. We chose not to use the `SPEC2006` benchmark as the applications targeted workloads handled by general purpose processors. We used 28 applications in the MiBench suite (the following 14 applications on big and small datasets):

1. `basicmath`
2. `bitcount`
3. `qsort`
4. `susan`
5. `jpeg`
6. `dijkstra`
7. `patricia`
8. `stringsearch`
9. `blowfish`
10. `sha`
11. `adpcm`
12. `CRC32`
13. `FFT`
14. `gsm`

The chosen set of MiBench applications spans all six classes provided by the benchmark: `automotive`, `consumer`, `network`, `office`, `security`, and `telecomm`.

3.2.1 Architectural Simulator

In order to analyze the impact of memory encryption on DBI-enabled DDR4 memory systems, we need to generate memory traces from the MiBench applications. Traditional simulator such as `Gem5` [1] and `ESESC` [5] did not meet our constraints. Gem5 and ESESC for the purposes of this study required significant overhead in terms of establishing an environment and integrating the simulators with MiBench applications to generate usable memory traces. Instead we chose use `Pin` [6] - a dynamic binary instrumentation tool developed by Intel as a lean, easy to use computer architectural simulator. Since Pin performs run-time program instrumentation analysis on compiled binary files, integrating Pin with MiBench required little effort. Once we generated the specific `Pintool` to generate memory traces, we could simply run the Pin simulator providing the Pintool and compiled Mibench application binaries as inputs.

Leveraging the IA-32 instruction set architecture framework provided by Pin, we created a `Pintool`, to output the memory trace of arbitrary compiled binaries. That is, we used the `dcache` and `safecopy` examples provided to create a single Pintool that provided a trace of all first-level cache misses. On every miss, the Pintool would output whether the operation was a `load` or `store` operation, the address and the value being read or written to memory. Unlike Gem5 and ESESC, Pin does not provide a cycle-by-cycle timing trace for when specific architectural events occur. Though this may be efficient for system-wide power analysis, our model is not parametrized on time but only the specific bit values of data units being read and written. In our experience, Pin worked well for generating such a coarse, event based memory trace.

For the purposes of our study we only simulated the data-cache with : 32KB total size, 8-way associativity, and 128B cache line size. The Pintool we generated allows the user to input a defined size, associativity and line size. The specific parameters chosen were used in previous Pin exercises as the default configuration and seemed reasonable for our study.

3.2.2 Memory Trace Analysis

Once the memory trace of data-cache misses was generated, the traces needed to analyzed to model the power consumption. To the best of our knowledge Pin itself does not offer off-chip DDR memory power consumption tools so we chose to study other simulators that could easily integrate with the Pin memory traces. We focused on using `DRAMSim` [7] however ran into many integration issues. Though DRAMSim takes as input a memory trace, it produced power traces that suggested that for each operation the power consumed was `nan`. In addition, the memory trace that DRAMSim used needed timing information which we could not accurately produce using Pin. Looking at the source code, we could not figure out an easy way to output the power consumed by each operation and decided instead to write our own `Python` based analysis script.

Our Python script reads the memory trace generated by Pin and tracks the sequential load and store misses. For each miss, it counts the total number of 0's and bit-flips assuming DBI is enabled. In order to account for DBI we use the following rules:

DBI-DC If the number of 0's in a given byte is greater than 4, then flip all the bits and set the DBI-DC flag. Using this method, the maximum number of 0's in any given data byte is 4.

DBI-AC If the number of bit-flips between the previous and new data value is greater than half the word length (32-bits in our system), flip all the bits and set the DBI-AC

flag. Using this method, the maximum number of bit flips for a given word is 16. For store operations, the previous state is determined by tracking all written values in memory previously. If the store operation is the first to write to the specific memory location then we assume the previous state in memory was a string of all 0's. For load operations, we keep track of the previous load operation and compare the new read value to the previous load operation's read value. This assumes that there is a single read bus from the DDR4 to the processor on which we can characterize the DBI-AC factor.

At the end of the memory trace simulation, the Python script calculates the DBI-DC and DBI-AC ratios as follows:

$$DBI - DCRatio = \frac{\texttt{Number of 0's}}{\texttt{Total number of bits}}$$

$$DBI - ACRatio = \frac{\texttt{Number of Bit Flips}}{\texttt{Total number possible of bits}}$$

Finally, we wrote `bash` scripts to fully integrate the simulation flow from the Pin data-cache simulation to the python script and generating graphs for the produced DBI-DC and DBI-AC ratios.

4. Evaulation

1. Looking at A, B ratios for loads and stores.
2. Put graphs and analyze them.
3. Intuition for decrease (store is lower - mostly misses)

5. Group Dynamics

6. Related Work

7. Conclusion

References

[1] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, Aug 2011.

[2] J. Burnett. DDR3 Design Considerations for PCB Applications. Jul 2009.

[3] R. M. Guthaus, S. J. Ringenberg, D. Ernst, M. T. Austin, T. Mudge, and B. R. Brown. MiBench: A free, commercially representative embedded benchmark suite. *Int'l Symp. on Workload Characterization (IISWC)*, 2001.

[4] T. M. Hollis. Data Bus Inversion in High-Speed Memory Applications. *IEEE Transactions on Circuits and Systems*, 2009.

[5] E. K. Ardestani and J. Renau. ESESC: A Fast Multicore Simulator Using Time-Based Sampling. In *International Symposium on High Performance Computer Architecture*, HPCA'19, 2013.

[6] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood. Pin: Building customized program analysis tools with dynamic instrumentation. *SIGPLAN Not.*, 40(6):190–200, Jun 2005.

[7] D. Wang, B. Ganesh, N. Tuaycharoen, K. Baynes, A. Jaleel, and B. Jacob. DRAMSim: A memory-system simulator. 2005.