Hardware Assistance for Trustworthy Systems through 3-D Integration

Jonathan Valamehr[†], Mohit Tiwari[‡], and Timothy Sherwood[‡]
[†]Department of Electrical and Computer Engineering
[‡]Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106
{valamehr@ece,tiwari@cs,sherwood@cs}.ucsb.edu

Ryan Kastner
Dept. of Computer Science and Engineering
Univ. of California, San Diego
La Jolla, CA 92093
kastner@cs.ucsd.edu

Ted Huffmire, Cynthia Irvine, and Timothy Levin
Dept. of Computer Science
Naval Postgraduate School
Monterey, CA 93943
{tdhuffmi,irvine,levin}@nps.edu

Abstract

Hardware resources are abundant; state-of-the-art processors have over one billion transistors. Yet for a variety of reasons, specialized hardware functions for high assurance processing are seldom (i.e., a couple of features per vendor over twenty years) integrated into these commodity processors, despite a small flurry of late (e.g., ARM TrustZone, Intel VT-x/VT-d and AMD-V/AMD-Vi, Intel TXT and AMD SVM, and Intel AES-NI). Furthermore, as chips increase in complexity, trustworthy processing of sensitive information can become increasingly difficult to achieve due to extensive on-chip resource sharing and the lack of corresponding protection mechanisms. In this paper, we introduce a method to enhance the security of commodity integrated circuits, using minor modifications, in conjunction with a separate integrated circuit that can provide monitoring, access control, and other useful security functions. We introduce a new architecture using a separate control plane, stacked using 3-D integration, that allows for the function and economics of specialized security mechanisms, not available from a co-processor alone, to be integrated with the underlying commodity computing hardware. We first describe a general methodology to modify the host computation plane by attaching an optional control plane using 3-D integration. In a developed example we show how this approach can increase system trustworthiness, through mitigating the cache-based side channel problem by routing signals from the computation plane through a cache monitor in the 3-D control plane. We show that the overhead of our example application, in terms of area, delay and performance impact, is negligible.

ACSAC '10 Dec. 6-10, 2010, Austin, Texas USA

1. INTRODUCTION

The development effort required to build a system is directly proportional to the cost of its failure; hence critical systems used in space shuttles and banks undergo much more rigorous development cycles than systems for home users. *High assurance* systems, which are designed to withstand attacks by professional, well-funded adversaries, require a tremendous investment of time, effort, and money by their small user base. In comparison to commodity systems, these systems generally lag far behind in performance and programmability. Unfortunately, for commodity processors, security threats are often not considered at the rapidly changing ISA [8] or micro-architecture levels. Clearly, a method that allows commodity parts to be retrofitted with protection mechanisms *without increasing the cost* for ordinary users and *without decreasing the performance* of the commodity processor will offer a significant advantage for high assurance system development.

Economics of Hardware Trust: The economics of trustworthy system development puts designers under constraints not faced by low assurance, commodity systems. For example, the expense of special-purpose hardware can make it costlier to provide both high performance and strong security. Even when hardware vendors incorporate security enhancements, integrating these mechanisms into a complex system design may present many practical and theoretical problems, driving up the costs and driving out the release schedule. In addition to the fact that such system devel-

opment costs per unit are very high, users requiring such functionality make up a small portion of the market. Sophisticated security mechanisms at the hardware level are typically targeted at a relatively small market sector and add unacceptable costs to commodity products.

Performance Ramifications: The design cycle of trustworthy systems also places constraints on the performance that can be realized in the final version of these systems. Due to the high non-recurring engineering (NRE) cost of manufacturing custom hardware and the small amortization base of low volume products, manufacturers are often forced to choose less costly alternatives, such as an older, cheaper process (e.g., 0.5um vs. 45nm).

As a result of these economic factors, designers of trustworthy systems requiring high performance need some way to incorporate commercial hardware components without compromising security. To address this challenge, a method of bridging the gap between cutting-edge technology and trustworthy systems is of paramount necessity.

3-D Integration for High Assurance: The primary goal of this paper is to introduce a new method by which security functionality can be added to a processor as a foundry-level configuration option. Specifically, we propose a new and modular way to add security mechanisms to current and next-generation processors through the use of 3-D integration. We advocate consolidating these security mechanisms into a physical overlay, *literally a separate plane* of circuitry stacked on top of a commodity integrated circuit. The security mechanisms that reside in this overlay can then be connected to the underlying chip with a variety of interconnect technologies, yet can be completely omitted without change to the commodity chip's function and without affecting its cost. Using 3-D hardware to alleviate this problem offers many advantages over other hardware solutions as well as software solutions. These advantages are fully explored in Section 2.

Contributions: In this paper, we show that an active layer[1], which we call a *3-D control plane*, specifically dedicated to security, has the potential to implement a variety of security functions in a cost-effective and computationally efficient way. Specifically, this paper makes the following contributions:

- We are the first to develop a method of using 3-D integration for trustworthy system development, and propose to combine an independently fabricated 3-D control plane containing arbitrary security functions (such as micro-architectural protection mechanisms) along with a commodity integrated circuit, which we refer to as the computation plane.
- Security functions can be broadly classified as either active or passive monitors, depending upon whether the 3-D control plane modifies signals on the computation plane. We describe precise circuit-level primitives required to build both active and passive monitors such that signals on the computation plane can be arbitrarily tapped, disabled, re-routed, or even overridden. We also outline how the 3-D control plane can be integrated in a purely optional and minimally intrusive manner with very minor modification to the commodity computation plane.
- We demonstrate our circuit-level primitives using an active monitor that implements a well-known micro-architectural protection mechanism: a cache monitor that can prevent access-driven cache side channel attacks.
- Finally, we validate the functionality of our circuit-level primitives using SPICE simulations, and build a synthesizable prototype of our 3-D cache monitor to evaluate the area-delay cost of its inclusion. We also quantify the impact of our cache protection mechanism on the performance of SPEC benchmark programs, through detailed timing simulations on an out-of-order CPU simulator.

Before describing the circuit-level modifications required of the computation plane, we begin with a discussion of 3-D integration and the opportunities it presents for trustworthy system design.

2. MOTIVATION FOR 3-D SECURITY

In this section, we provide a short background on 3-D integration and present our motivation for using 3-D hardware to address the concerns raised in Section 1. Since 3-D integration is an existing technology already used in industry [24, 26], our work does not discuss the feasibility of 3-D integration but rather focuses on the security ramifications of a 3-D control plane.

2.1 3-D Integration

While the details of how we use this technology are more fully described in Section 3, the main idea is that two pieces of silicon are fused together to form a single chip. The two active layers of the silicon (the commodity computation plane and 3-D control plane) are connected through inter-die vias[2] (micron-width wires that are chemically "drilled-and-filled" between the layers) which run vertically between them. This ability to interconnect multiple active layers enables the addition of an optional die that specifically implements security functions to a commodity processor die. This 3-D control plane would have access to the security-dependent signals of the system. Such a system could be sold to customers requiring application-specific security policy enforcement, information flow control, or other security-specific support. Commodity systems, on the other hand, are unlikely to include this additional, more costly functionality that only benefits a small number of customers.

Attaching multiple layers of silicon together in 3-D stacks is a relatively new, yet already marketed technology [26], which is being explored by most of the major microprocessor manufacturers [6]. As opposed to most current 2-D circuits, which use only one active layer for computation, 3-D circuits contain multiple active layers, or *planes*, which are then connected using techniques such as inter-die vias (or "posts"). Several 3-D interconnect technologies are currently being evaluated in industry as a means of stacking multiple chips together. Some potential applications include the

[1]The active layer is the silicon layer where transistors reside, and metal layers are fabricated above that connect the transistors together. We define a "plane" as the combination of the silicon and metal layers that compose a typical 2-D integrated circuit.

[2]Vias are physical connections between two wires on different metal layers.

Security Architecture	Power	Bandwidth	Delay
Security Functions On-chip	Low power consumption, with the only addition being the power used by security logic and interconnect	Bus width is limited due to contending traffic and component congestion throughout the chip (1-16 bytes) running at core clock speed (>2 GHz)	On-chip delay is dictated by the length of interconnect, which is often very large between components
Security Functions on a Co-Processor	In addition to powering another chip, driving long off-chip bus wires consumes large amounts of power	Low data bus widths due to I/O pin availability (1-8 bytes) running at external clock speed (~ 400 MHz)	Very large delay between off-chip co-processor and CPU (>200 cycles)
Security Functions on a 3-D Control Plane	3-D security only slightly increases power consumption, and can use less power than on-chip due to exploitation of locality of security modules	3-D allows bus widths to be increased significantly (up to 128 bytes) running at core clock speed (>2 GHz)	3-D exhibits low delay due to the short length of inter-die vias, as well as the locality that can be exploited to shorten critical paths

Figure 1: This table compares other hardware options for security against a 3-D control plane and shows the advantages and disadvantages in terms of power, bandwidth, and delay [11, 20].

stacking of DRAM or bigger caches directly onto the processor die to alleviate memory pressure [17] and designing stacked chips of multiple processors [2].

Toshiba has applied 3-D integration to a CMOS image sensor camera module for mobile phones, which they call a Chip Scale Camera Module (CSCM), achieving a significant reduction in size while satisfying high-speed I/O requirements [24]. The Toshiba work demonstrates that cost savings are possible with 3-D integration because passive components, which provide load matching between the chip and the camera, can be integrated into the chip. This makes the passive components cheaper, smaller, and faster than board-level components; therefore, savings can be realized in power, resistance, and capacitance, as driving lines between layers consumes much less power than between chips. Furthermore, multiple layers, each optimized for its particular function, can be combined into a single stack.

Large microprocessor manufacturers are unlikely to integrate support for highly specialized security mechanisms because the market for such features represents such a small portion of their total customer base. This is an example of Gresham's Law: if a manufacturer incurs the cost of security mechanisms deemed unnecessary by the general commodity market, a competing, less costly product without such mechanisms will dominate. By fabricating the optional 3-D control plane with functions that are complementary to (but separate from) those of the main processor, stacked interconnect offers the potential to add security mechanisms to a small subset of devices without impacting the overall cost of the commodity processor.

Just to be clear, we are advocating the development of a processor which is *always fabricated with special connections built in for joining it with a control plane.* The difference between the system sold for the cost-sensitive consumer market and the one that is sold to the security-sensitive customer is only whether a specialized security device is actually stacked on top of the standard integrated circuit, utilizing the special connections. Additional benefits to this approach are that security mechanisms implemented in hardware are faster than software-only approaches, and the security mechanisms can be specialized for particular sets of applications, systems, and customers.

2.2 3-D vs. Other Hardware Solutions

This section discusses the advantages of using 3-D integration over other hardware methods such as on-chip and co-processor implementation of security functions. In general, implementing security functions in software is less costly than in hardware, but software implementations have worse performance and are more susceptible to tampering. Implementing security functions in hardware is more expensive, but the result has better performance and is more resilient to manipulation.

Why not On-Chip?: Implementing security features on-chip creates many issues and discrepancies. It would force all users of the chip uninterested in system trustworthiness to incur the possible negative effects of the added security logic. As discussed previously, an unacceptable consequence of on-chip security is the increase in cost for all consumers. In addition, on-chip security functions have the potential of decreasing the overall performance of the chip, as security modules may need long interconnect wires to data and control lines spanning the whole chip area; this can be mitigated by the exploitation of locality in the 3-D layer as well as short interconnect through inter-die vias as explained in Figure 1. The large majority of microprocessor consumers are chiefly concerned with the performance of the chip, and on-chip security could provide advantages to competing chip manufacturers who do not incorporate these security features. Because of market pressures, chip manufacturers are reluctant to pursue such a course. With 3-D security, the small percentage of consumers who need the added security logic have the option of including it in their systems, while consumers who do not need this extra logic can omit it.

Why not use a Co-processor?: A co-processor solution, much like 3-D security, allows the consumer to have the option of including additional security logic. However, unlike 3-D security, an off-chip co-processor can not safely access internal micro-architectural control signals without possibly making them susceptible to outside tampering. This makes 3-D security much more attractive and feasible, as any resource or control signal can be accessed and modified by the 3-D control plane. Also, co-processor solutions suf-

fer from the utilization of slow, power-hungry off-chip buses. These off-chip buses operate at much slower frequencies than can be realized with a 3-D solution (Figure 1), and they can introduce large delays in processor speed. In addition, off-chip buses have to interface with the main processor through the main processor's I/O pins, and they are limited in size based on available pins. This equates to smaller bus widths (Figure 1), which can further hinder performance. Choosing which pins to interface between the processor and the co-processor also creates inflexible co-processor designs, because we are limited to accessing or modifying those pins, whereas with a 3-D solution we can create any number of different co-processor designs and access any internal signal. Aside from performance, a co-processor solution also entails increased power usage, as driving long off-chip buses requires much more power than driving short inter-die vias to a 3-D control plane. A 3-D security scheme does not fall victim to any of these issues.

Disadvantages of 3-D Security: 3-D security holds much promise as a solution; however, it is not without tradeoffs. Chips fabricated using 3-D integration need greater thermal management, and, without additional cooling, will run at higher temperatures due to the proximity of components [11]. While this is a known issue, it is not insurmountable and can be addressed with more expensive cooling solutions. Another disadvantage of 3-D chips is their expected manufacturing yield, as the functionality of the complete chip is dependent on the individual yield of each of the two dies.This can create lower overall yield than the individual dies. However, the cost of this lower yield will not be incurred by most consumers, as the decrease in yield only applies to the systems that need the 3-D control plane attached.

This section has compared 3-D security with other software and hardware solutions for trustworthy systems. The 3-D control plane can include different types of security monitors. In the next section, we will discuss both of these types of monitors, and follow with our novel circuit architecture to allow the use of an optional 3-D control plane.

3. 3-D SECURITY ARCHITECTURE

The 3-D control plane can include several security functions on one die, implemented as either passive or active monitors. While passive monitoring in 3-D for system profiling has been explored previously [12], a novel contribution of this work is providing active monitoring in a 3-D control plane. In the following section we explain the uses of these two types of monitors, and describe a novel circuit-level architecture that allows us to make the functions of these monitors available as a fabrication option in an overlay.

3.1 Passive and Active Monitors

Passive Monitors: One potential use of the 3-D control plane is to act as a passive monitor, simply accessing and analyzing data from the computation plane. For instance, we may wish to monitor accesses to a particular region of memory or audit the use of a particular set of instructions. To monitor these events, we must understand when such events are occurring, which necessitates *tapping* some of the wires from the processor. This requires posts and vias to the instruction register and memory wires, which gives us direct access to the currently executing instruction.

Passive monitoring is reasonably straightforward to implement in 3-D technology, as it just requires a set of vias to the top of the computation plane, and then a post from there to the 3-D control plane. Figure 2 shows such a post.

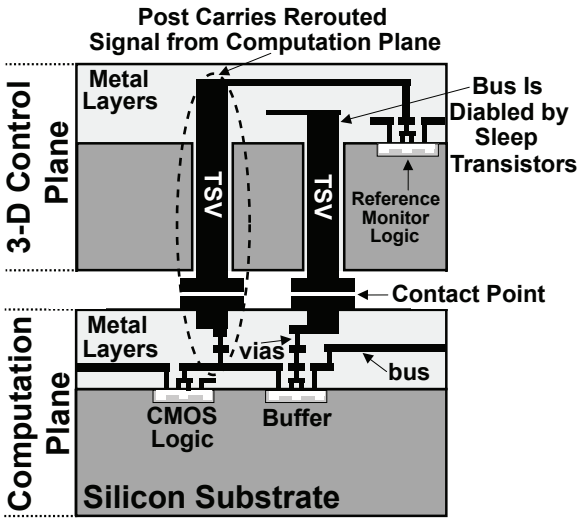

Figure 2: This figure shows the low level architecture for a method to route data/control lines on the computation plane through the 3-D control plane. This can be performed to isolate resources in the computation plane by disabling a bus, for example. The computation plane and the 3-D control plane are connected by inter-die vias or through-silicon vias (TSVs). Posts are required to tap the required signals needed by the security logic, and sleep transistors are used to either reroute, override, or disable lines on the computation plane. Using these primitives, we can build mechanisms to monitor the computation plane.

The area overhead of this passive style monitoring in a 3-D layer was analyzed by Mysore et al. [12] in the context of hardware support for analyzing the processor in real time for debugging and performance profiling, which has high throughput requirements and is very slow to implement in software. Their conclusion was that, even with very pessimistic assumptions about the technology, there would be less than a 2% increase in the total area on the computation plane and that there would be no noticeable delay added. The small amount of area overhead is due to the need to save space for the vias across all of the layers of metal.

Active Monitors: Whereas passive monitoring allows for auditing, anomaly detection, and the identification of suspicious activities, systems enforcing security policies often require strong guarantees about restrictions to overall system behavior. A novel contribution of our work is the employment of active monitors; an active monitor enables control of information flow between cores, the arbitration of communication, and the partitioning of resources.

The key ability needed to support such functionality is to *reroute* signals to the 3-D control plane and then *override* them with potentially modified signals. With this technology and minor modification of the computation plane, we

can force all inter-core communication, memory accesses, and shared signals to travel to the 3-D control plane, where they are subject to both examination and control. For instance, we can ensure that confidential data being sent between two cores (which are traditionally forced to traverse a shared bus) is not leaked to a third party with access to that bus.

We have developed a method to modify signals on the computation plane that is accomplished in two parts. The first part is to ensure that the monitor has unfettered access to all the signals (tapping), which is, in essence, the same as the passive monitoring scenario described above. The second part is to selectively disable those links, essentially turning off portions of the computation plane (e.g., a bus), or overriding them to inject different values. The difficulty is that we must remove a capability (the connection between two components) only by adding a 3-D control plane (which cannot physically cut or impede that wire). The computation plane must be fully functional without an attached 3-D control plane, yet it needs to be constructed so that by adding circuitry, the targeted capability can be completely disabled. To accomplish this, components in the computation plane must be modified to support active monitoring.

3.2 Circuit-level Modifications

This section introduces the circuit level modifications we will make in order for the 3-D control plane to perform its intended function and for the computation plane to be able to execute in its absence. These primitives are illustrated in Figure 4.

Sleep Transistors: A novel and alternative method for disabling links is to physically impede the connection itself. While this sounds intrusive, we are the first to leverage an existing circuit technique called *power gating* [18] for this application. Support for power gating is added through the addition of *sleep transistors* placed between a circuit's logic and its power/ground connections. The sleep transistors act as switches, effectively removing the power supply from the circuit. The circuit is awake when the transistors are activated by a specific signal, which provides power to the circuit, allowing it to function normally. Alternatively, the sleep transistors can be given the opposite input and turned off, thus disconnecting the power to the circuit, temporarily removing all functionality, and effectively putting the circuit to sleep.

Sleep transistors are traditionally used to temporarily disable unused portions of an integrated circuit, saving power by preventing leakage current [19]; however, their use is also beneficial for providing the isolation an active monitor requires. With only a small amount of added hardware (two transistors and two resistors, shown in Figure 3) and posts for connectivity to the 3-D control plane, we can selectively turn off portions of the computation plane to force adherence to any specific security policy enforced in the control layer. Finally, many modern chips already employ power gating. This reduces the amount of additional hardware necessary to apply our security primitives, since only posts to the 3-D control plane to carry the control signal are required.

In addition to selectively removing power from some components on-chip, sleep transistors may be used to perform several key functions on data and control lines required by active monitors. Sleep transistors can be placed on any link that may need to be disabled or controlled. They can be

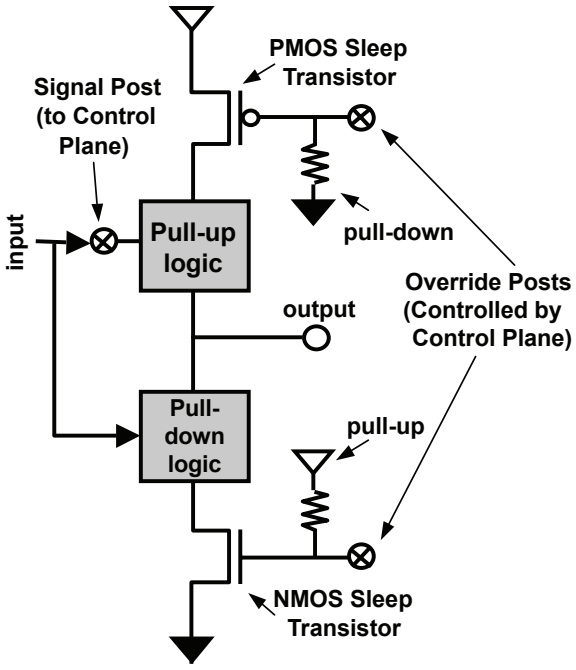

Figure 3: A circuit diagram of sleep transistors in the computation plane being used to remove power from a circuit.

managed by the 3-D control plane by simply providing a post that connects to their gate input. The following functions all use only one or two transistors per line and present a new set of options for trustworthy system development.

Tapping: *Tapping* can be used to send the requested signals to the 3-D control plane without interrupting their original path. As shown in Figure 4a, we use a transistor and apply the correct voltage to the gate of the transistor to create the additional path of the signal to the 3-D control plane. This is particularly useful when we are performing analysis (e.g., dynamic information flow tracking) on the flow of information on the computation plane without affecting its original functionality (Figure 6). *Tapping* can also be used when security logic on the 3-D control plane is dependent on some data in the computation plane, without the need to change their values in the system. In our 3-D cache eviction monitor (Section 3.3) we use *tapping* to access the address of a load or a store instruction to determine whether a cache eviction is allowed without interfering with the normal flow of the address through the bus.

Re-routing: *Re-routing* (Figure 4b) uses two transistors per line to send the requested signals to the 3-D control plane and block their transmission to the originally intended path. A pull-up resistor is attached to the gate of the transistor that is *disabling* the line, to force a connection when the 3-D control plane is not attached. *Re-routing* can be used in situations where we want to create new buses between resources on-chip.

Another use of *re-routing* is using a signal for a different purpose than was originally intended. Once on the 3-D control plane, the signal can be analyzed and combined with other data from the 3-D control or computation planes, or simply stored for later use. This can then be coupled with *overriding* (Figure 4c) to change control or data outputs on

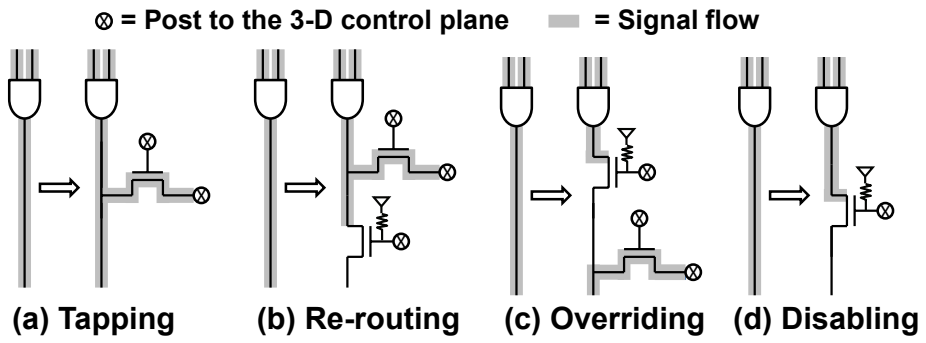

Figure 4: This figure shows the four different kinds of circuit level modifications that can be made and their respective diagrams. The sample base circuit is an AND gate and is found to the left of each circuit modification. *Tapping* requires only one transistor to optionally propagate the signal to the 3-D control plane, while *re-routing* and *overriding* need transistors with pull-up resistors to ensure their continued function for systems omitting the 3-D control plane. *Disabling* uses a transistor and a pull-up resistor to uphold the connection in the absence of the 3-D control plane, while giving the 3-D control plane the option of disconnecting the line for systems utilizing it.

the computation plane based on new logic in the 3-D control plane (Figure 7).

Overriding: *Overriding* (Figure 4c) allows us to block the intended value of a signal and modify it to a desired value for the security layer's function (Figure 6 and Figure 7). *Overriding* uses two transistors and a pull-up resistor much like *re-routing*. For some security applications, critical control signals need to be changed in order to adhere to a security policy that is being enforced by the 3-D control plane. In our 3-D cache eviction monitor (Section 3.3), we use *overriding* to change the value of a cache's write-enable signal (see Figure 8), allowing us to inject a value to allow or deny the eviction of a specific cache line.

Disabling: *Disabling* (Figure 4d) allows us to completely stop the flow of data on a common bus or a specific signal line. Uses of disabling include the ability to isolate a specific resource from unintended accesses, or enforcement of policies that require tight guarantees on the integrity of data on a shared bus. Many bus protocols work on a *mutual trust* system, where access to the bus is controlled by the devices that are connected, not by a trusted arbiter. In situations such as this, it is important to preserve trustworthy execution and the confidentiality of data during a sensitive computation. *Disabling* can be used to forcibly block access to a bus to ensure secure transactions without the possibility of unintended access (Figure 5).

3.2.1 Spice Simulation Results:

To verify the correctness of our circuit-level modifications, we developed Spice circuit models for each of the circuits in Figure 4 and used Spice simulations to read the voltage values at certain nodes for each circuit. Two experiments were performed, with input voltages at the transistor terminals corresponding to the 1) absence of the 3-D control plane and corresponding to the 2) presence of the 3-D control plane. NMOS transistors from 45nm predictive technology models [1] were used to characterize the sleep transistors, but PMOS transistors can also be used. Regardless of which transistor type we use, we need to buffer the signal after it has traveled through the transistor to ensure a strong signal propagation. During the experiment where the 3-D control plane is omitted, the transistor gates are not powered, and the pull-up resistors successfully power the transistor's gate and create a short, allowing the signal to pass normally. When the transistor gates are powered, we can successfully control the circuit and perform the function for each respective circuit. These experiments verify our ability to create functional systems with the option to add a modular 3-D control plane.

3.3 Theoretical 3-D Applications

Isolation: One potential application of our circuit-level primitives is the active isolation of resources in a system. For example, in multi-core processors there are shared data and address buses that rely on a mutually trusting shared bus protocol, where each core is responsible for its own arbitration. This is problematical for the security of bus traffic on a system running code of varying trust levels on each core. Figure 5 outlines this situation and how we can use *Disabling* to disconnect a core from the bus for any given amount of time, creating a Time Division Multiple Access (TDMA) protocol between the cores and the shared resources of interest.

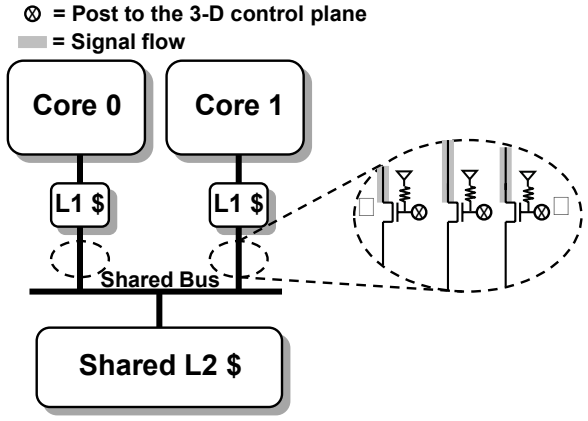

Figure 5: A multi-core processor with two cores that we wish to isolate. This is achieved using *Disabling* to block the connections to the bus for the core that is not currently allowed to use the bus.

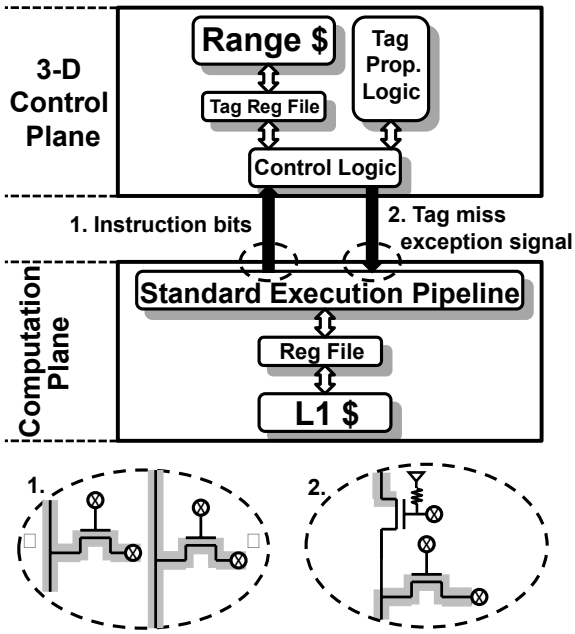

Figure 6: A 3-D system monitor tracking data flow on the computation plane. This is achieved using *Tapping*(1) to read signals that we want to track and *Overriding*(2) to raise an exception signal.

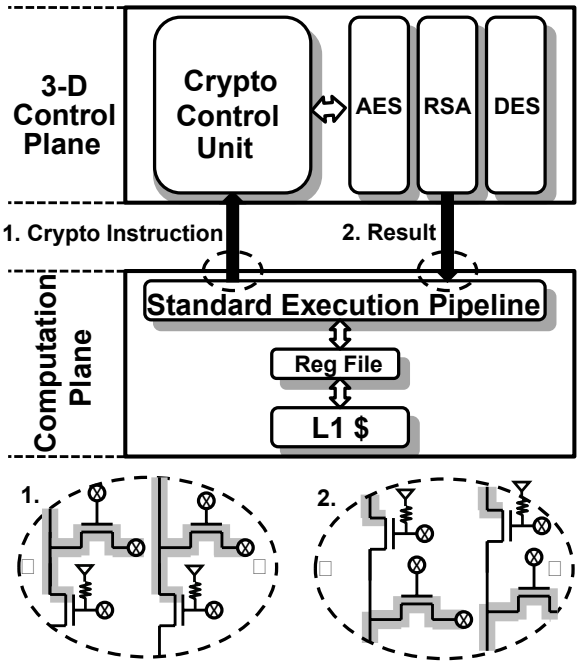

Figure 7: A 3-D cryptographic engine used to perform secure cryptography in the 3-D control plane, using *Re-routing*(1) to block the instruction execution on the computation plane and to send the instruction to the 3-D control plane to be executed. The result can be placed back in the execution pipeline using *Overriding*(2).

System Analysis and Monitoring: It is often useful to monitor the activity of the computation plane for auditing, intrusion detection, or post-mortem analysis. Information flow tracking in the 3-D control plane, for example, attempts to identify, track, mitigate, and deter the execution of malicious code. The basic premise of dataflow tracking is the storage of *metadata* in the form of tags associated with each individual address in memory. A dataflow tracking architecture with a small cache [21] that compresses memory addresses with matching metadata tags can be utilized in the 3-D control plane (Figure 6), to raise an exception in the event that malicious execution on the computation plane is detected. For such a monitor, we can use *Tapping* to read signals of interest on the computation plane and use *Overriding* to optionally modify an exception signal without tampering with normal use.

Secure Alternate Service: Another potential application is augmenting the functionality of the computation plane with additional hardware for security computations. For systems requiring high-bandwidth cryptographic functionality, we can implement a cryptographic engine on the 3-D control plane that can accept cryptographic instructions being executed on the computation plane, performing the operation immediately before sending the result back to the execution pipeline. This is achieved by using *Re-routing* to extract the cryptographic instructions from the standard execution pipeline, execute the instruction, and use *Overriding* to inject the result into the pipeline as if it were part of the normal instruction execution flow. While cryptographic hardware has been included in microprocessors [8], 3-D security allows the addition of any cryptographic algorithm or implementation to be included in the system as a foundry-level option. Essentially, 3-D security introduces flexibility in the system hardware, allowing any number of cryptographic cores to be optionally added to the processor.

The techniques described in this section provide powerful tools for implementing active monitors in the 3-D control plane, thereby allowing the addition of security-critical functionality. If used appropriately, this can eliminate certain types of side channels by mediating the use of a shared resource. In the following section we present the architecture of an active cache eviction monitor that we have implemented for the 3-D control plane using the previously discussed circuitry.

3.4 Architecture of a 3-D Cache Monitor

This section presents the custom architecture shown in Figure 8, implemented in the 3-D control plane, for eliminating access-driven cache side channel attacks. Concurrent processing platforms present several security issues; although these architectures provide increased performance through instruction-level parallelism, their methods of resource sharing leave them vulnerable to side channel attacks. One side channel attack [16] uses a simultaneous multithreading processor's shared memory hierarchy, exploiting the process-to-process interference arising from the cache eviction policy to covertly transfer information. As a result, an attacker thread may be able to extract information from a victim thread, such as a cryptographic key. This threat was demonstrated by Percival [16], where an implementation of the RSA encryption standard was attacked using the cache eviction protocol and used to observe, in small chunks, the total cryptographic key. This was achieved by having a ma-

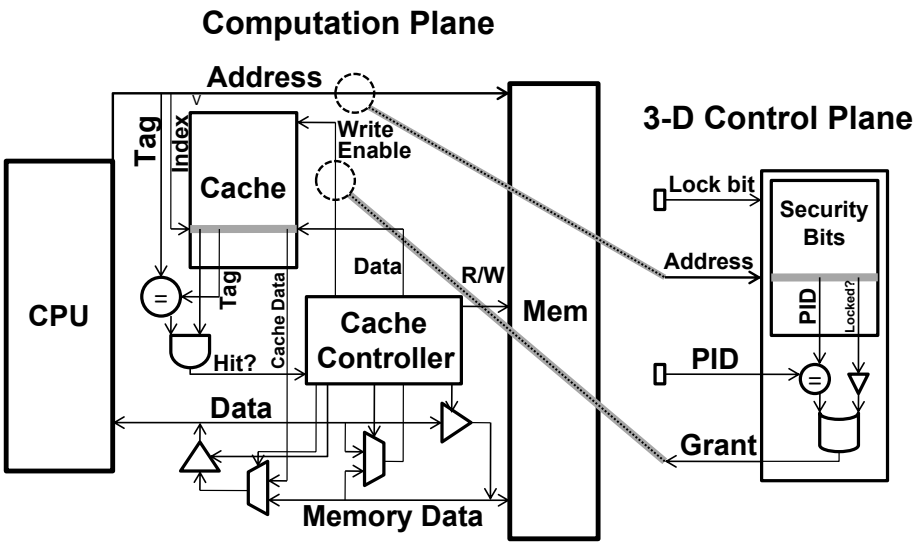

Figure 8: The architecture of a CPU/cache memory hierarchy and our 3-D cache eviction monitor working in concert. The address of the corresponding load/store is tapped to be sent to the 3-D control plane, and the cache write-enable signal is overridden in the case of a locked cache line eviction. The Lock bit as well as the Process ID (PID) are also provided to the 3-D control plane. We discuss options on how to access this information in Section 4. Once the cache monitor receives the load/store address, the Lock bit, and the PID, it can determine whether a cache eviction can be granted based on whether the cache line is locked or whether the PID matches, and issue the appropriate *override* signal on the cache write-enable signal.

licious thread consume sufficient memory so that when the victim thread executed, the spy thread's cache lines would be evicted. Thus by measuring subsequent access times for its cached items, the spy thread can observe which of its cache lines had been evicted by the victim. Once the spy thread knows these cache lines, it can infer parts of the cryptographic key due to the nature of the table look-ups performed during the encryption. Slowly but surely, the whole key can be compromised with a relatively low margin of error.

Our method to prevent these attacks is based on a previously proposed hardware solution [23]. In our application of this scheme, the 3-D control plane maintains a cache protection structure that indicates, for each cache line, whether it is protected, and if so, for which process. When a different process loads or stores data related to a protected cache line, no eviction will occur, and the data is not cached unless an alternate line is available in the cache protocol being used. Figure 9 shows a flowchart describing this new protocol, while Figure 10 provides a high-level overview of how the cache and the 3-D control plane will interact. Specifically, the cache protection structure contains memory elements on the 3-D control plane to store *security bits*, which hold the permissions of a process to evict shared cache entries of other processes. With this in place, when instructions proceed to load or store data, these security bits are first checked to determine whether to grant a cache eviction that might otherwise have occurred without policy oversight. As mentioned previously, when the 3-D control plane is not attached to the processor, the cache functions as normal. However, when the 3-D control plane is added, we can utilize the above strategy to avoid undesirable cache evictions. This is performed with an updated version of the *load* and *store* instructions. These instructions, named *secure_load* and *secure_store*, change the security bits in the 3-D control plane to reflect the process that currently occupies the line. Effectively, *secure_load* and *secure_store* modify the necessary bits to ensure that once a cache line is occupied by a process that needs cache eviction control, it cannot be evicted by any other process. This will control a simultaneous multithreading processor's shared memory and eliminate any threat of an access-driven side channel attack.

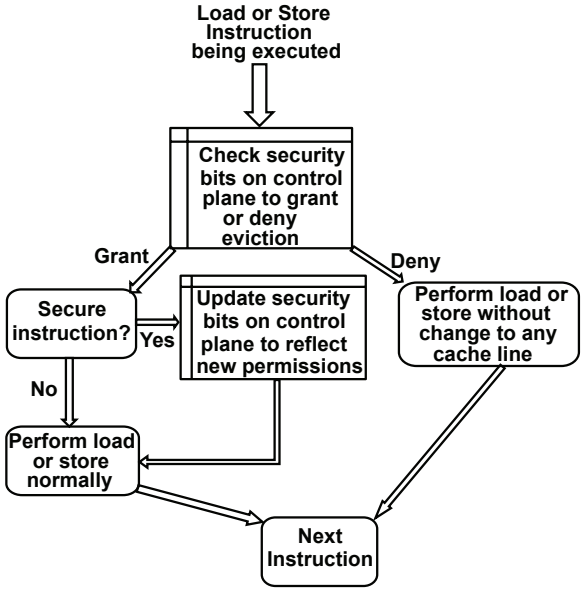

Figure 9: This flow chart describes how loads and stores are executed when the 3-D control plane is in place.

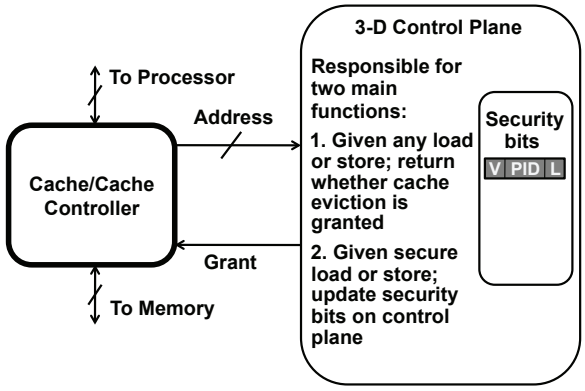

Figure 10: A high-level logical overview of how the cache and the 3-D control plane interact in our cache monitor, as well as the 3-D control plane's responsibilities when active.

As a proof of concept, we have developed a synthesizeable version of our security mechanism in Verilog. We designed our security mechanism as a separate module that is interfaced with a simple cache that we also implemented as a hardware design. Our design uses a straightforward 4-way set associative cache. For every load or store instruction, the cache controller first checks the 3-D control plane module to determine whether the related cache line is protected from evictions. The security bits on the 3-D control plane hold a valid bit, a process ID, and a lock bit for each cache line. During the loads and stores, these security bits are checked in the 3-D control plane, and a *grant* signal is generated if the cache line is open to eviction. While every load and store will be forced to check the security bits before proceeding, these security bits can only be manipulated by using *secure_load* and *secure_store*.

We synthesized both modules and have verified that the design is functional, easily scaled, and can be implemented with low overhead. This will be discussed in further detail in the following sections where we analyze performance metrics, overhead for a modern processor, and feasibility.

4. EXPERIMENTAL RESULTS

This section outlines our synthesis results, and discusses the effect of including the 3-D cache eviction monitor, both in terms of critical path and cache performance. We find that the 3-D cache eviction monitor does not increase the critical path of the circuit, and we observe that this type of cache-line locking produces very little performance degradation for many programs. We also discuss integration options and feasibility for the 3-D control plane on a sample commodity processor.

4.1 Performance and Analysis

Synthesis Results: In this section, we analyze the performance and area overhead of the 3-D cache eviction monitor. We use Altera Quartus to synthesize our design and extract specific timing and area information (Figure 11). To provide a clear picture of the overhead and performance effects of our design, we gathered timing and area information for both the cache/cache controller alone, as well as the cache/cache controller being interfaced with the 3-D cache eviction monitor module. The synthesis was performed for a Stratix II device, with the compiler set to optimize for performance. The standalone cache was able to run at approximately 151MHz; when we include our 3-D cache eviction monitor, the maximum frequency remains at 151MHz. The 3-D cache eviction monitor synthesized by itself has a maximum frequency of 217MHz. These maximum frequencies indicate that the critical path in the circuit including the 3-D cache eviction monitor resides in the underlying cache/cache controller, resulting in no change in cycle time for the circuit with the addition of the 3-D cache eviction monitor.

Design	Max Frequency	Area (LUTs)
Cache/Cache controller	~151MHz	468
3-D cache eviction monitor	~217MHz	291
Cache/Cache controller with 3-D monitor attached	~151MHz	749

Figure 11: The synthesis results produced by Quartus for the cache and cache controller, as well as the 3-D cache eviction monitor.

The above performance metrics do not take into account the delay of the vertical posts between the computation plane and the 3-D control plane. Loi et al. [11] characterized the worst-case delay of a 3-D bus that travels from one corner of a chip to the opposite corner on a 3-D layer above, and they found this delay to be about .29ns. Even with the addition of this bus delay to the 3-D cache eviction monitor's critical path, the new critical path is still less than that of the cache/cache controller, further confirming that the addition of the 3-D cache eviction monitor will have no effect on the performance of the cache subsystem.

Performance Evaluation: We evaluated the performance impact of locking specific cache lines with our 3-D cache eviction monitor. We used PTLsim [25], a cycle-accurate x86 simulator, to execute the SPEC2000 benchmark suite. The experiments we developed outline two scenarios:

- 1) Running each benchmark with a 32KB 4-way set associative cache, representing a 32KB L1 cache with no cache line locking. This is a best-case performance bound because running the benchmark and the AES program together will be slower than running the AES program by itself.
- 2) Running each benchmark on a 32KB 4-way set associative cache, with one of the ways locked, effectively resulting in a 24KB 3-way set associative cache. This is a worst-case performance bound because the AES program is smaller than an entire way of the cache (8192 bytes).

We modeled our cryptographic process after the AES algorithm, which can occupy up to 4640 bytes with an enlarged T-Box implementation [5]. With this in mind, 8192 bytes is

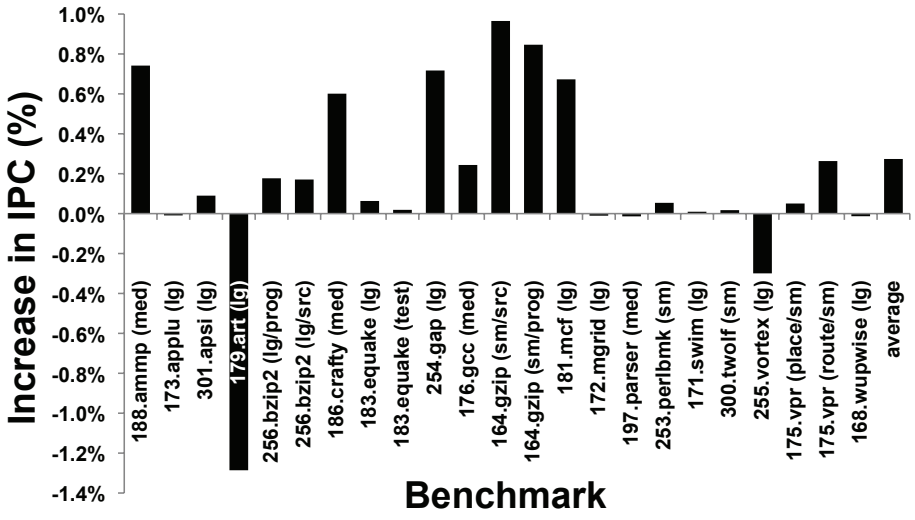

Figure 12: The results of our cache experiment using SPEC2000 benchmarks that were executed in PTLsim. We use two different sizes of cache to calculate a bound on the performance impact of locking a cryptography program like AES to one way of the cache. The average degradation in IPC between the two different cache sizes is 0.2%, indicating that this form of cache line locking has a small impact on performance for many different types of programs.

more than enough to store all of the necessary information (state vectors, round keys, and look-up tables) for AES.

Results for this experiment can be found in Figure 12. The average degradation in IPC is 0.2%, indicating that this form of cache line locking has a small impact on performance for many different types of programs. We were not able to build the binaries for mesa, galgel, facerec, or fma3d. In addition, we encountered technical difficulties with lucas, eon, and sixtrack.

4.2 Discussion and Integration Options

When integrating our security scheme for cache management with a processor, several factors must be considered. Implementing security functionality requires the following capabilities: access to the process ID of a thread during its execution, access to the address bus, and a method of discerning between normal and secure loads and stores. These are the high-level requirements of the 3-D control plane; some vertical posts are also needed to propagate this information to the 3-D control plane.

For our 3-D cache eviction monitor to function, we need to know the process ID of the thread performing the current load or store function. One option we have explored is accessing the process ID register that some architectures have, such as the ARM926EJ-S [10]. Accessing this register through the vertical posts will give the 3-D control plane direct access to the current process ID, allowing the control plane to compare it to the security bits.

We also need to know when loads and stores are being executed. One option is *tapping* the instruction bus, allowing us to monitor the execution of loads and stores and subsequently apply our security functions to those instructions. During the execution of loads and stores, the control plane will follow the protocol outlined in Figure 9.

Finally, the 3-D control plane must know whether each load and store operation is secure or not, so that the system can determine whether the security bits in the 3-D control plane need to be updated. One way to supply this information is to modify the instruction set to include two special instructions, *secure_load* and *secure_store*. This would create separate instructions of which the 3-D control plane is aware in order to distinguish between normal load/store and secure load/store operations. Another option is to add a register to the computation plane that reflects whether the current instruction is secure or not. The operating system can control this bit based on whether the instruction is secure or not, and the control plane could read this register. Both options are feasible and have no negative implications on the rest of the system.

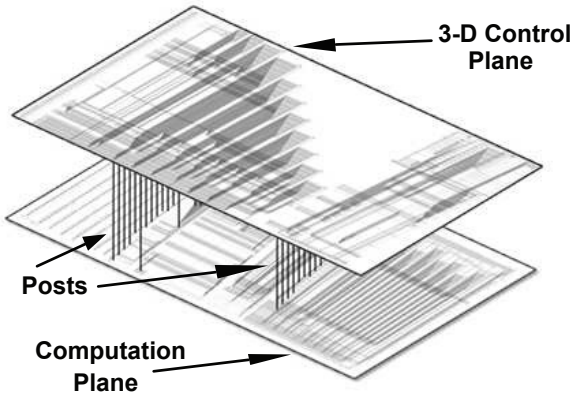

Figure 13: This figure is a visualization of the physical circuit-level diagram of our 3-D cache monitor. Gate-level diagrams were compiled in Quartus after synthesis of the modules.

Delivery of the previously mentioned required information to the 3-D control plane will be through the vertical posts. A general idea of the number of posts the 3-D control plane needs on a given system is the sum of the number of bits of: the *address size*, the *process ID size*, possibly one post for the secure register, and a *grant* bit post. For the ARM926EJ-S, this results in under 100 vias, which equates to about the silicon space for 50 bits of memory; this is a small and reasonable number of vertical posts to implement a strong security measure.

5. RELATED WORK

In this section, we discuss other work associated with cache side channel problems. We also discuss related work on the use of 3-D technology for security and communication as applied to CMP architectures.

On-chip and board-level resource sharing between cores is often used to enhance CMP performance. However, contention for those resources at the microarchitectural level can provide the basis for *side-channel cryptanalysis* attacks and other covert timing channels. Code and data caches, as well as the branch prediction unit, are some of the shared resources that can be exploited in these attacks [9, 4, 3]. In these cases, one process's use of the resource perturbs the response time of the next process that accesses it, in a predictable manner. Single-core computers with simultaneous multithreading, and SMP systems with cache coherency mechanisms, can have similar problems.

One approach to prevent resource contention in a concurrent execution model is to utilize separate physical caches for each core, or provide separate virtual caches within the physical cache (if virtual cache support is available in hardware) [15, 23]. Various forms of cache disablement are possible, including turning it off, turning it off for certain cores or processes, or turning off the eviction and filling of the cache through use of the processor *no-fill* mode. The latter can be used to create *sensitive sections* [13] of code that could not interfere with the cache behavior observable by other cores or processors – assuming that the code is not interruptible or that the previous processor mode is restored on interrupt, as otherwise, other processes might sense the change to the state of the processor (i.e., to *no-fill*), creating another covert channel [5].

Specific cryptographic attacks can be defeated or minimized by lowering the bandwidth of the cache channel, such as through nondeterministic ordering of access to cache [14] which makes detailed cache-use profiling difficult; and nondeterministic cache placement [22, 15] or nondeterministic polyinstantiation [7] of cache entries, [23] which, while the specific cause of the interference may be masked, still allows detection of cache misses caused by another process. The 3-D approach has the advantage of being able to implement many of these schemes for resolving cache contention, while doing it in an isolated environment, without modification to the processor ISA.

6. CONCLUSIONS

3-D integration offers the ability to decouple the development of security mechanisms from the economics of commodity computing hardware. We described the technology to enable passive and active monitoring of the computation plane by adding a minimal amount of hardware. Passive monitoring requires vias and posts, while active monitoring uses sleep transistors to perform several novel functions on the computation plane. Using these techniques, we described a number of broad strategies to enhance the security of the computation plane with a control plane. To provide quantitative measurements of the impacts of the control plane, we considered cache side channels, developing a complete hardware description for a cache with a control plane that eliminates eviction-based side channels. This work provides a pathway for the high-assurance community to utilize high-performance hardware while shortening development cycles for trustworthy systems.

Acknowledgements

The authors would like to thank the anonymous reviewers for their insightful comments. This research was funded in part by National Science Foundation Grant CNS-0910734.

7. REFERENCES

[1] Arizona State University Predictive Technology Models, Predictive Technology Models for 45nm Processes, Available at. `http://www.eas.asu.edu/~ptm/`.

[2] N. G. A. Akturk and G. Metze. Self-Consistent Modeling of Heating and MOSFET Performance in 3-D Integrated Circuits. *IEEE Transactions on Electron Devices*, 52(11):2395–2403, 2005.

[3] O. Acıiçmez. Yet another microarchitectural attack: Exploiting I-cache. In *Proceedings of the First Computer Security Architecture Workshop (CSAW)*, Fairfax, VA, November 2007.

[4] O. Acıiçmez, J. Seifert, and C. Koc. Micro-architectural cryptanalysis. *IEEE Security and Privacy Magazine*, 5(4), July-August 2007.

[5] D. J. Bernstein. Cache-timing attacks on AES. `http://cr.yp.to/antiforgery/cachetiming-20050414.pdf`, Apr. 2005. Revised version of earlier 2004-11 version.

[6] B. Black, M. Annavaram, N. Brekelbaum, J. DeVale, L. Jiang, G. H. Loh, D. McCauley, P. Morrow, D. W. Nelson, D. Pantuso, P. Reed, J. Rupley, S. Shankar, J. Shen, and C. Webb. Die Stacking (3D) Microarchitecture. *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 469–479, December 2006.

[7] D. E. Denning and T. F. Lunt. A multilevel relational data model. In *Proc. IEEE Symposium on Security and Privacy*, pages 220–234, 1987.

[8] S. Gueron. White paper: Advanced encryption standard (AES) instructions set, Intel corporation, July 2008.

[9] J. Kelsey, B. Schneier, C. Hall, and D. Wagner. Side channel cryptanalysis of product ciphers. *Journal of Computer Security*, 8(2–3):141–158, 2000.

[10] A. Limited. ARM926EJ-S technical reference manual, 2001-2008.

[11] G. L. Loi, B. Agrawal, N. Srivastava, S.-C. Lin, T. Sherwood, and K. Banerjee. A Thermally-Aware Performance Analysis of Vertically Integrated (3-D) Processor-Memory Hierarchy. In *Proceedings of the 43nd Design Automation Conference (DAC)*, June 2006.

[12] S. Mysore, B. Agrawal, S. Lin, N. Srivastava, K. Banerjee, and T. Sherwood. Introspective 3-D chips. In *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, San Jose, CA, October 2006.

[13] D. A. Osvik, A. Shamir, and E. Tromer. Cache attacks and countermeasures: the case of AES: (extended version). Technical report, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science,, Rehovot 76100, Israel, Oct. 2005.

[14] D. Page. Theoretical use of cache memory as a cryptanalytic side-channel. Technical Report CSTR-02-003, Department of Computer Science, University of Bristol, June 2002.

[15] D. Page. Partitioned cache architecture as a side channel defence mechanism. In *Cryptography ePrint Archive, Report 2005/280*, August 2005.

[16] C. Percival. Cache missing for fun and profit. In *Proceedings of BSDCan 2005*, Ottowa, Canada, May 2005.

[17] K. Puttaswamy and G. H. Loh. Implementing Caches in a 3D Technology for High Performance Processors. In *IEEE International Conference on Computer Design (ICCD) 2006*, pages 525–532, October 2005.

[18] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand. Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits. *Proceedings of the IEEE*, 91(2), February 2003.

[19] K. Shi and D. Howard. Sleep transistor design and implementation simple concepts yet challenges to be optimum. *IEEE VLSI-DAT Taiwan*, 2006.

[20] H. Sun, J. Liu, R. S. Anigundi, N. Zheng, J.-Q. Lu, K. Rose, and T. Zhang. 3D DRAM design and application to 3D multicore systems. *Design and Test of Computers, IEEE*, 26(5), September 2009.

[21] M. Tiwari, B. Agrawal, S. Mysore, J. K. Valamehr, and T. Sherwood. A small cache of large ranges: Hardware methods for efficiently searching, storing, and updating big dataflow tags. In *Proceedings of the International Symposium on Microarchitecture (Micro)*, Lake Como, Italy, November 2008.

[22] Topham and Gonzalez. Randomized cache placement for eliminating conflicts. *IEEETC: IEEE Transactions on Computers*, 48, 1999.

[23] Z. Wang and R. Lee. New cache designs for thwarting cache-based side channel attacks. In *Proceedings of the 34th International Symposium on Computer Architecture*, San Diego, CA, June 2007.

[24] H. Yoshikawa, A. Kawasaki, T. Iizuka, Y. Nishimura, K. Tanida, K. Akiyama, M. Sekiguchi, M. Matsuo, S. Fukuchi, and K. Takahashi. Chip scale camera module (CSCM) using through-silicon-via (TSV). In *Proceedings of the International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, February 2009.

[25] M. T. Yourst. PTLsim: A cycle accurate full system x86-64 microarchitectural simulator. In *Performance Analysis of Systems & Software, 2007. ISPASS 2007. IEEE International Symposium on*, pages 23–34, 2007.

[26] I. Ziptronix. 3D integration for mixed signal applications, 2002.