2012-12-01

A 3D Split Manufacturing Approach to Trustworthy System Development

Valamehr, Jonathan

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/37460

NPS-CS-12-004

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

A 3D Split Manufacturing Approach to Trustworthy System Development

by

Jonathan Valamehr, Timothy Sherwood,
Ryan Kastner, David Marangoni-Simonsen,
Ted Huffmire, Cynthia Irvine, and Timothy Levin

1 December 2012

Approved for public release; distribution is unlimited.

Prepared for: National Science Foundation

This page left intentionally blank

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

Daniel T. Oliver
President

Leonard A. Ferrari
Executive Vice President and Provost

The report entitled "*A 3D Split Manufacturing Approach to Trustworthy System Development*" was prepared for and funded by National Science Foundation, 4201 Wilson Boulevard, Arlington, VA 22230.

Further distribution of all or part of this report is authorized.

This report was prepared by:

Jonathan Valamehr
Ph.D. Candidate, UCSB

Timothy Sherwood
Associate Professor, USCB

Ryan Kastner
Professor, UCSB

David Marangoni-Simonsen
Student, Harvey Mudd College

Ted Huffmire
Assistant Professor, NPS

Cynthia Irvine
Professor, NPS

Timothy Levin
Research Associate Professor, NPS

Reviewed by:

Peter Denning
Chairman
Department of Computer Science

Released by:

Jeffrey D. Paduan
Dean of Research

This page left intentionally blank

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)* 01-12-2012	**2. REPORT TYPE** Research	**3. DATES COVERED** *(From - To)* 03/01/2010 - 11/30/2012

4. TITLE AND SUBTITLE
A 3D Split Manufacturing Approach to Trustworthy System Development

5a. CONTRACT NUMBER

5b. GRANT NUMBER
CNS-0910734, CNS-0910389, and CNS-0910581

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Jonathan Valamehr, Timothy Sherwood, Ryan Kastner, David Marangoni-Simonsen, Ted Huffmire, Cynthia Irvine, and Timothy Levin

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Center for Information Systems Security Studies and Research (NPS CISR)
1411 Cunningham Road, Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER
NPS-CS-12-004

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Science Foundation, 4201 Wilson Blvd., Arlington, VA 22230

10. SPONSOR/MONITOR'S ACRONYM(S)
NSF

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
Not applicable

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Securing the supply chain of integrated circuits is of the utmost importance to computer security. In addition to counterfeit microelectronics, the theft or malicious modification of designs in the foundry can result in catastrophic damage to critical systems and large projects. In this Technical Report, we describe a 3D architecture that splits a design into two separate tiers: one tier that contains critical security functions is manufactured in a trusted foundry; another tier is manufactured in an unsecured foundry. We argue that a split manufacturing approach to hardware trust based on 3D integration is viable and provides several advantages over other approaches.

15. SUBJECT TERMS
Hardware-oriented security and trust, 3D integration, trustworthy system development, policy enforcement, cryptographic hardware, embedded systems security, malicious hardware, trusted foundries

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**	UU		Ted Huffmire
					19b. TELEPHONE NUMBER *(Include area code)*
Unclassified	Unclassified	Unclassified		40	831-656-7601

This page left intentionally blank

NPS-CS-12-004

3Dsec
3D security

3Dsec Technical Report

A 3D Split Manufacturing Approach to Trustworthy System Development

Jonathan Valamehr, Timothy Sherwood,

Ryan Kastner, David Marangoni-Simonsen,

Ted Huffmire, Cynthia Irvine, and Ted Huffmire

1 December 2012

This page left intentionally blank

This material is based upon work supported by the National Science Foundation under Grants No. CNS-0910734, CNS-0910389, and CNS-0910581. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation

Author Affiliations

University of California, Santa Barbara:

Jonathan Valamehr
Department of Electrical and Computer Engineering
University of California, Santa Barbara
Santa Barbara, CA 93106

Timothy Sherwood
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106

University of California, San Diego:

Ryan Kastner
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093

Harvey Mudd College:

David Marangoni-Simonsen
Department of Electrical Engineering
Harvey Mudd College
Claremont, CA 91711

Naval Postgraduate School:

Ted Huffmire, Cynthia Irvine, and Timothy Levin
Center for Information Systems Security Studies and Research
Naval Postgraduate School
Monterey, CA 93943

This page left intentionally blank

A 3D Split Manufacturing Approach to Trustworthy System Development

Jonathan Valamehr
Department of Electrical and Computer Engineering
University of California, Santa Barbara
Santa Barbara, CA 93106
Email: valamehr@ece.ucsb.edu

Timothy Sherwood
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106
Email: sherwood@cs.ucsb.edu

Ryan Kastner
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093
Email: kastner@cs.ucsd.edu

David Marangoni-Simonsen
Department of Electrical Engineering
Harvey Mudd College
Claremont, CA 91711
Email: dmarangonisimonsen@gmail.com

Ted Huffmire, Cynthia Irvine, and Timothy Levin
Department of Computer Science
Naval Postgraduate School
Monterey, CA 93943
Email: tdhuffmi@nps.edu; irvine@nps.edu; levin@nps.edu

Abstract

Securing the supply chain of integrated circuits is of the utmost importance to computer security. In addition to counterfeit microelectronics, the theft or malicious modification of designs in the foundry can result in catastrophic damage to critical systems and large projects. In this Technical Report, we describe a 3D architecture that splits a design into two separate tiers: one tier that contains critical security functions is manufactured in a trusted foundry; another tier is manufactured in an unsecured foundry. We argue that a split manufacturing approach to hardware trust based on 3D integration is viable and provides several advantages over other approaches.

Index Terms

Hardware-Oriented Security and Trust; 3D Integration; Trustworthy System Development; Policy Enforcement; Cryptographic Hardware; Embedded Systems Security; Malicious Hardware; Trusted Foundries

The views expressed this this Technical Report do not represent the official policy of the United States Government, the Department of Defense, or the National Science Foundation.

A shorter version of this Technical Report was published as a Transactions Brief in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*.

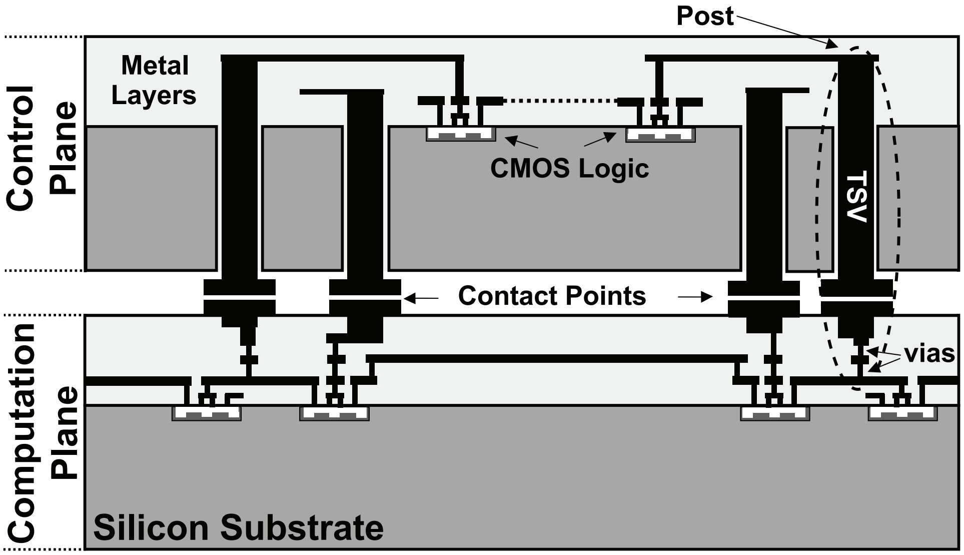


Fig. 1. Two-tier 3D integrated circuit in which the control plane (above) and computation plane (below) are joined using a face-to-back bonding process. Face-to-face bonding is also possible.

I. Introduction

Security is an essential design goal in computer architecture, which must be addressed throughout the entire lifecycle of a system. The process of designing hardware requires trusting intellectual property (IP) cores and computer-aided design (CAD) tools developed by third parties, as well as the fabrication and packaging of the final system. Sensitive IP is vulnerable to theft and modification during tape-out, even if a perfect design free of security flaws is sent to the foundry (a trusted foundry does not solve the problem of flawed designs).

Supply chain trust is a global issue. Many countries are concerned about the theft and malicious modification of sensitive designs of chips used in mission-critical systems. In response to this trend, governments have established trusted foundry programs. A trusted foundry is certified as a trusted environment able to produce leading-edge integrated circuits from trusted sources. While the availability of trusted foundries is beneficial, lower-priority projects may not have access to the trusted foundry, and the cost per unit may be higher.

We propose the use of 3D integration to allow designers of trustworthy systems to leverage the capabilities of commodity foundries. In our approach, a design is split into two tiers: the *computation plane* and the *control plane*. The computation plane houses a high-performance processor and is manufactured in an unsecured foundry. The control plane contains critical security functions and is manufactured in a trusted foundry. The control plane is optional, and including it is a foundry-level configuration choice. This tier is a separate plane of circuitry stacked on top of the computation plane, and the two tiers are joined with vertical interconnect. The decision to include or omit the control plane does not affect the function, performance, or cost of the computation plane. In addition to the benefits of split manufacturing, our technique provides a financial solution for system builders who wish to add security features to cutting-edge processors.

Contributions: In this Technical Report, we show that a control plane, a custom die dedicated to security, has the potential to implement a variety of security functions in a cost-effective and computationally efficient way when joined to a computation plane using 3D integration. Our approach uses circuit-level primitives for accessing signals on the computation plane so that they can be tapped, disabled, rerouted, or overridden, to integrate with the control plane in a purely optional and minimally intrusive manner. We also extend our preliminary work [29] to incorporate: (1) refinements to the circuit-level primitives that support our approach; (2) new 3D systems we have designed to evaluate our approach; and (3) an argument comparing split manufacturing based on 3D integration to other approaches to split manufacturing.

II. Motivation for 3D Security

With 3D integration, two integrated circuits are fused together to form a single chip, as shown in Figure 1. The two dies are connected with through-silicon vias (TSVs) or face-to-face vias, depending on whether a face-to-back or face-to-face bonding process is used. The ability to connect multiple dies allows an optional die dedicated to security functions (the control plane) to be joined with a commodity processor die (the computation plane). The control plane has direct access to the internal signals of the computation plane, benefiting customers requiring application-specific security policy enforcement, information flow control, or other security-specific support.

While 3D integration is an emerging technology, many 3D systems have successfully overcome the initial challenges of 3D technology, including thermal, testing, and yield. For example, Toshiba has applied 3D integration to a CMOS image sensor camera module for mobile phones, achieving a 55% reduction in volume and a 36% reduction in footprint while satisfying high-speed I/O requirements for video, with the required data rate per pin as high as 130MB/s for VGA at 30fps and 650MB/s for 3.2Mpixel at 15fps [31]. Kim et al. have joined a tier containing 64 CPU cores running at 277MHz with a tier containing 256K of SRAM; their system achieves a 63.8GB/s memory bandwidth [12]. Loh et al. provide detailed analysis of the advantages of 3D integration for implementing a cache, with their analysis showing significant reductions in access latency and energy per access; they also devise a 3D floor plan for the Intel Pentium 4, removing critical paths and improving performance and power by 15% [15]. Loh et al. also show that 3D integration can improve clock frequency by 10.3% for the Alpha 21364, and that 3D integration can expose instruction-level parallelism for the Alpha 21264, improving performance by 9.7% [15]. Loh et al. also show that a 3D version of a dynamic non-uniform cache architecture reduces average L2 access time by 50%; they also show that 3D stacking can allow the cache size to increase, reducing average memory access latency by 13% and reducing off-chip bandwidth by 3x [15]. Using the Intel Core 2 Duo as a baseline, Black et al. show that a 3D stacked DRAM cache can reduce the cycles per memory access by 13% on average and as much as 55% while reducing off-chip bandwidth and power by 66% [5]. Loh proposes optimizations to 3D DRAM that result in 1.75x speedup over prior 3D-DRAM approaches, and Loh also proposes a L2 miss handling architecture that achieves an extra 17.8% performance improvement [14]. Puttaswamy and Loh show that a 3D-partitioned cache can reduce latency by 21.5%, reduce energy consumption by 30.9%, and increase IPC by 12% [22].

Several approaches to split manufacturing are possible. First (option 0), a CPU can be fabricated in an unsecured foundry, and software implementing security functions can be loaded for execution onto the CPU in a secure facility. In general, implementing security functions in software is less costly than in hardware, but software implementations have worse performance (and greater power consumption) and are more susceptible to tampering. Next (option 1), both the processor and hardware security functions reside on the same 2D chip, which is manufactured in a trusted foundry (this is a base case that does not allow for split manufacturing). Next (option 2), a coprocessor implementing security functions can be manufactured in a trusted foundry, and it resides on the same circuit board as a main processor that is manufactured in an unsecured foundry. Finally (option 3), one tier is made in a trusted foundry, the other tier is made in an untrusted foundry, and the two are joined in a trusted facility using 3D integration.

Based on the area, power, and performance figures from the literature presented above, we extrapolate that option 2 (using a separate processor and coprocessor connected at the circuit board level) will consume more power than option 1 (implementing everything on the same processor), and option 3 (using 3D) will consume the least power, in general. Of course, individual designs may vary (some may not benefit from 3D integration), but in general, option 3 will also have the least delay and greatest bandwidth, followed by option 1 and then option 2, which has the greatest delay and the least bandwidth due to the use of slow, power-hungry off-chip buses.

A variant of option 1 was recently proposed by the US Intelligence Advanced Research Projects Agency (IARPA), in which an unsecured foundry (called the front-end-of-line, or FEOL) manufactures a layer of transistor devices and then sends the unfinished wafer to a trusted foundry (called the back-end-of-line, or BEOL), which adds metal layers that connect the devices to form useful circuits [10]. The interface between the FEOL and BEOL circuits in the IARPA program is different from the interface between the tiers in option 3, which allows one tier to monitor, disable, reroute, and override signals in another tier. Also, while the transition between the FEOL and BEOL foundries is an open research challenge for the IARPA program, joining separately made dies is already proven technology with 3D integration.

A variant of option 0 uses reconfigurable hardware: an FPGA is made in an unsecured foundry, and a design is loaded onto the FPGA in a trusted facility. While an FPGA may achieve higher throughput than a CPU, the design is protected by bit-stream encryption, which can be thwarted by side-channel attacks on the bit-stream decryption mechanism. Anti-fuse FPGAs can help mitigate this problem, but it is a write-once technology, unlike SRAM-based FPGAs.

We consider the trust issues of split manufacturing in [8] and establish that our threat model includes unintentional hardware design flaws and malicious software in the computation plane; the threats of hardware Trojans (i.e., malicious inclusions), physical tampering/probing, simple/differential power analysis, and compromising RF/acoustic/photonic emanations are outside the scope of our work. A first-order concern is whether the output of the unsecured foundry needs to be independently trustworthy for the joined system to provide certain trustworthy functions. If so, it would seem to obviate the purpose of the effort. For option 3, we found that the independence of a control plane from interference by the computation plane (through active corruption of the processing or passively, via withholding of services) is a primary requirement for trustworthy behavior of the control plane [8]. However, the control plane can often choose whether to establish dependencies on the computation plane. The detection of malicious inclusions on the computation plane is another security feature that can be hosted on the control plane, although this technology is very immature. We believe it is not yet possible to add a layer of hardware to a computation plane that is riddled with malicious inclusions - effectively bearing an unknown degree of resemblance to its design - in the hopes that the composition of the two layers will be highly trustworthy. Nevertheless, provided that the requirements of self-protection and dependency layering are met for the control plane, it is possible to offer an alternate service to the computation plane, to actively override the computation plane for enforcement of policies, and to passively monitor the computation plane with high integrity [8]. Note that while option 3 allows a control plane to access the internal signals of a computation plane, this is not possible via the circuit board level coprocessor interface associated with option 2. Therefore, while a coprocessor can provide an alternate service such as encryption to a main processor, actively overriding or passively monitoring internal signals is impossible with option 2.

While the threats of malicious hardware, physical tampering, power analysis, and compromising emissions are outside the scope of this work (we are assuming that the 3D IC lacks countermeasures against these threats), we do not believe that 3D integration necessarily increases the risk of a physical probing attack on sensitive signals carried by the inter-die vias. Probing for the purposes of testing is much harder for 3D than for 2D, due to the difficulty of probing an individual TSV and the risk of breaking a TSV. Furthermore, the difficulty of chemically removing the package of a 3DIC and separating the bonded layers is significantly greater than the challenge of probing a circuit board connecting a processor and coprocessor (option 2). For example, a 3D cryptographic coprocessor's tiers are tightly bonded and have no exposed shared buses or I/O pins, and the inter-die vias are enclosed in a package and only accessible by removing the package and separating the tight bond between the tiers [28]. However, Chemical-Mechanical Planarization (CMP) (i.e., "sanding") of a tier is available to professional attackers, and future work is needed to develop secure protocols for the inter-die interface of option 3.

Finally, comparing option 1 and option 3, we note that 3D integration offers the potential to offload logic to the control plane. We argue that many security applications can benefit from this capability. For example, Tiwari et al. have developed an information flow tracking method that increases area by 70% over the base processor's area [26]. Additional area allows for the implementation of additional cipher implementations [28], as well as real-time monitoring and processing of programs in execution [17].

III. 3D Security Architecture

The control plane can include several security functions on one die, implemented as either passive or active monitors. A passive monitor accesses and analyzes data from the computation plane, e.g., memory accesses or instructions. Monitoring these events requires *tapping* some of the wires in the processor. 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 these types of behavior. A novel contribution of our work is the employment of active monitors, e.g., to control information flow between cores, to arbitrate communication, and to partition resources.

The key ability needed to support such functionality is to *reroute* signals to the control plane and then *override* them with potentially modified signals. With this technology, we can force all communication, memory accesses, and shared signals to travel to the 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 on-chip 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 when the control plane is connected. The first part is to ensure that the monitor has unfettered access to the signal

	1 Generic TSV Receptacle	128 TSV Receptacles	5-Stage MIPS Processor
Area (Library Area Units)	84.1	10764.8	240,000

TABLE I
AREA OF GENERIC TSV RECEPTACLE IN 90NM TECHNOLOGY NODE

(tapping), which is the same as the passive monitoring scenario described above. The second part is to disable the signal, preventing it from propagating (e.g., via a bus). The difficulty is that we must remove a capability (the connection between two components on the computation plane) only by adding a control plane. The computation plane must be fully functional without an attached control plane, yet it needs to be constructed so that by connecting circuitry, the targeted capability can be achieved. To accomplish this, components in the computation plane must be modified to support active monitoring.

Our preliminary work [29] introduces the circuit-level modifications needed for the control plane to perform its intended function and for the computation plane to function in its absence. The primitives each provide an environment for receiving one or two inter-die vias. We refer to this computation plane environment as a TSV receptacle or socket.

Tapping: can be used to pull specific signals to the control plane without interrupting their original path. This is particularly useful when performing analysis (e.g., dynamic information flow tracking) of the flow of information on the computation plane without affecting its original functionality.

Disabling: allows us to completely stop the flow of data on 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.

Overriding: allows us to block the intended value of a signal and modify it to a value determined by the security layer. 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 control plane.

Re-routing: combines tapping and disabling to send signals to the control plane and block their transmission to the original path. Re-routing can be used in situations where we want to create new "controlled" buses between resources on the computation plane. Re-routing also allows the use of a signal for a different purpose than originally intended. Once on the control plane, the signal can be analyzed and combined with other data from the control or computation planes, or simply stored for later use. This can then be coupled with overriding to change control or data outputs on the computation plane based on new logic in the control plane.

Diode: Diodes allow information to flow in only one direction (note that one-way communication can be enforced using various electrical techniques besides a diode, such as a buffer) [9]. Such an arrangement can enforce a policy requiring that information flow from a low confidentiality component to a high confidentiality component but not vice versa.

Generic TSV Receptacle: can be used to support multiple control plane applications with the same computation plane, e.g., when different control planes are used or when the applications on a given control plane are reconfigured [9]. A given TSV receptacle can be used for different purposes from application to application. A generic TSV receptacle provides design flexibility at the cost of additional circuitry and posts. In order to explore the area ramifications of incorporating these Generic TSV Receptacles on the computation plane, we synthesized one Generic TSV Receptacle and compared it to a simple five-stage pipelined MIPS processor. While it is infeasible to build sockets for every signal, good candidates include registers, control signals, and shared buses; engineers must strike a balance between the generality of the interface and its performance and cost.

The Generic TSV Receptacle and MIPS processor were written in Verilog and synthesized using Synopsys Design Compiler in 90nm technology. The area of the Generic TSV Receptacle, shown in Table II, is 84.1 AU. This is a very small percentage of the area of the full processor. Even if we needed 128 Generic TSV Receptacles, the additional area added to the MIPS processor is about 4.5%, which is relatively small. This percentage would be even less when adding Generic TSV Receptacles to a large, modern commodity processor that includes structures such as caches, advanced branch predictors, and Floating Point Units.

Assured Generic TSV Receptacle: The Generic TSV Receptacle could include diodes to assure that the information

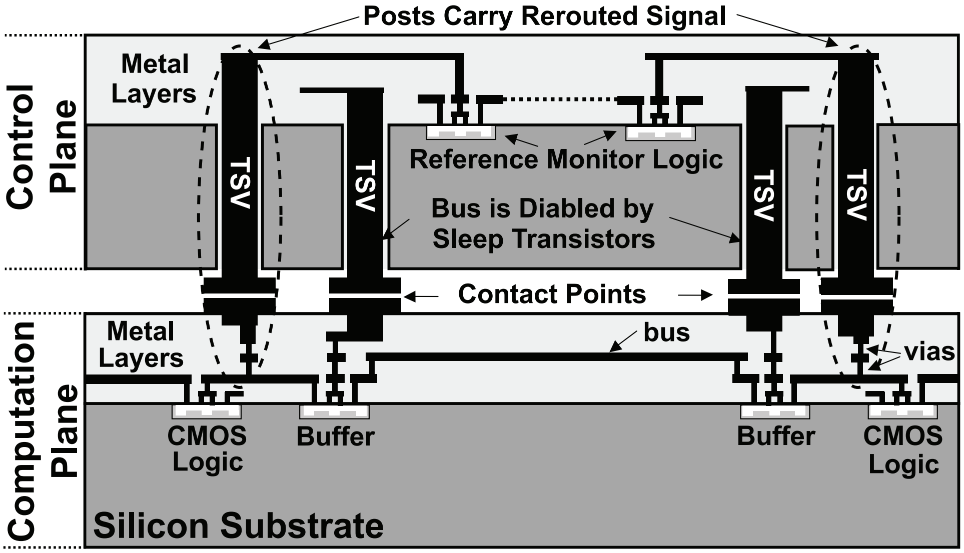


Fig. 2. This figure shows the low level architecture for a method to route data/control lines on the computation plane through the control plane. This can be performed to isolate resources in the computation plane by disabling a bus, for example. The computation plane and the control plane are connected 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.

flows of each post are precisely controlled, e.g., to prevent back flow from the computation plane.

A. Passive and Active Monitors

1) Passive Monitors: One potential use of the 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. Tapping 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 3D technology, as it just requires a set of vias from the audited circuit to the top of the computation plane, and then a post from there to the control plane. Figure 2 shows such a post. The area overhead of this passive style monitoring in a 3D layer was analyzed by Mysore et al. [17] 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.

2) 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 these types of behavior. A novel contribution of our work is the employment of active monitors, e.g., to control information flow between cores, to arbitrate communication, and to partition resources.

The key ability needed to support such functionality is to *reroute* signals to the control plane and then *override* them with potentially modified signals. With this technology, we can force all communication, memory accesses, and shared signals to travel to the 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 on-chip 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 the signal (tapping), which is, in essence, the same as the passive monitoring scenario described above. The second part is to selectively disable the signal,

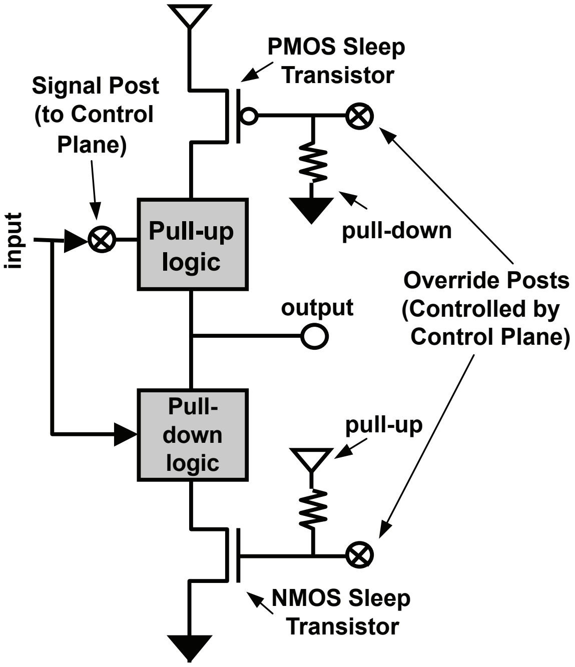


Fig. 3. A circuit diagram of sleep transistors in the computation plane being used to remove power from a circuit.

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 control plane (which cannot physically cut or impede that wire). The computation plane must be fully functional without an attached 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.

B. Circuit-level Modifications

This section introduces the circuit level modifications we will make in order for the 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. The primitives each provide an environment for receiving one or two posts. We refer to this computation plane environment as a *TSV receptacle/socket*.

1) Sleep Transistors: We use an existing technique called *power gating* [23] to disable links by physically impeding their connections. 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 "gating" (i.e., 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 [24]; 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 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 for certain components of the die. This reduces the amount of additional hardware necessary to apply our security primitives to such a component, since only posts to the control plane to carry the control signal would be 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

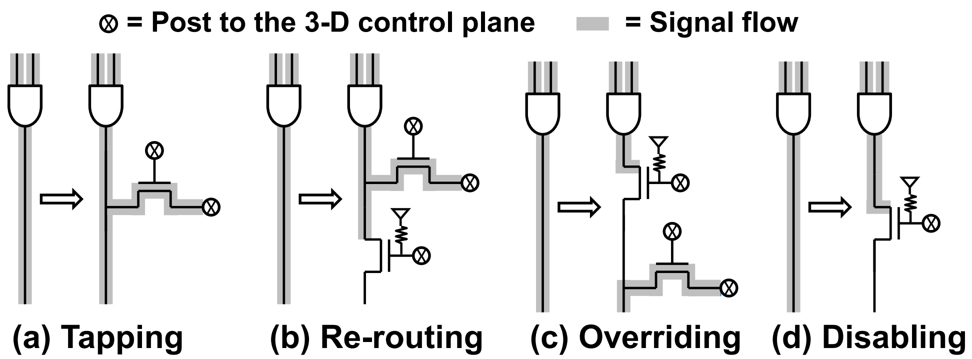


Fig. 4. This figure shows the four different kinds of circuit level modifications that can be made and their respective diagrams. The sample base circuit is an AND gate and is found to the left of each circuit modification. *Tapping* requires only one transistor to optionally propagate the signal to the control plane, while *disabling*, *re-routing*, and *overriding* need transistors with pull-up resistors to ensure their continued function for systems omitting the control plane.

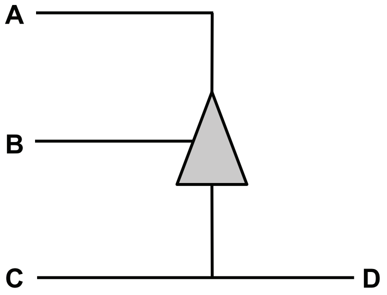


Fig. 5. Schematic of implementation of the tapping primitive. C is the computation plane signal being tapped. B is the signal from the control plane for turning the tap on or off. A is the tapped signal to the control plane. The gray triangle is a non-restoring tri-state buffer, which directly passes the input to the output but does not allow noise to propagate through the circuit. We verified the correct behavior of this primitive in simulation with Spice and ModelSim.

on any link that may need to be disabled or controlled. They can be managed by the 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.

2) Tapping: Tapping can be used to send the requested signals to the 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 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 14). *Tapping* can also be used when security logic on the control plane is dependent on some data in the computation plane, without the need to change the value of the data. In our 3D cache eviction monitor (Section V-A) 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. Figure 5 shows the schematic of an implementation of the tapping primitive.

3) 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 13). Figure 6 shows the schematic of an implementation of the disable primitive.

4) 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 14 and Figure 15). *Overriding* uses two transistors and a

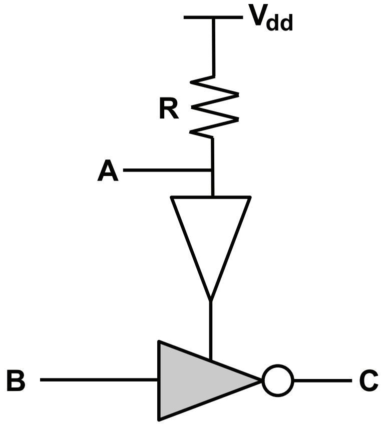


Fig. 6. Schematic of implementation of the disable primitive. B is the computation plane signal being disabled. A is the control plane signal for commanding the disablement. The white triangle is a buffer, which helps in driving loads from weak inputs and prevents back-driving (when an output can propagate through to an input). The gray triangle with a circle is a restoring tri-state buffer, which does not propagate noise but inverts the input. The purpose of inverting B is to prevent C from driving B. We verified the correct behavior of this primitive in simulation with Spice and ModelSim.

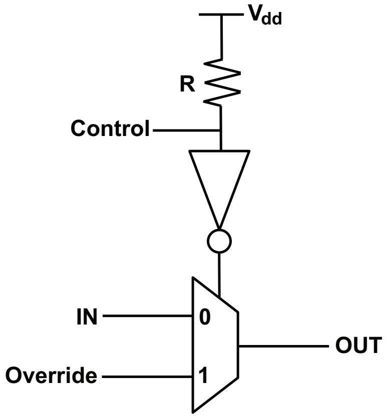


Fig. 7. Schematic of implementation of the override primitive. The white triangle with a circle is a NOT gate, which also serves as a buffer that requires relatively few transistors. The trapezoid is a multiplexer. $Control$ is a signal from the control plane commanding the override. IN is a computation plane signal to be overridden. $Override$ is another signal from the control plane. The value of $Control$ determines whether OUT takes the value of IN or $Override$. We verified the correct behavior of this primitive in simulation with Spice and ModelSim.

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 control plane. In our 3D cache eviction monitor (Section V-A), we use *overriding* to change the value of a cache's write-enable signal (see Figure 23), allowing us to inject a value to allow or deny the eviction of a specific cache line. Figure 7 shows the schematic of an implementation of the override primitive.

5) Re-routing: Re-routing (Figure 4b) uses two transistors per line to send the requested signals to the 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 control plane is not attached. *Re-routing* can be used in situations where we want to create new "controlled" buses between resources on the computation plane.

Another use of *re-routing* is using a signal for a different purpose than was originally intended. Once on the control plane, the signal can be analyzed and combined with other data from the 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 the computation plane based on new logic in the control plane (Figure 15). Figure 8 shows the schematic of an implementation of the reroute primitive.

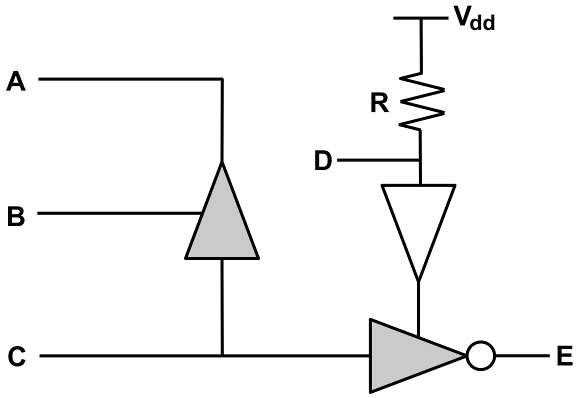


Fig. 8. Schematic of implementation of reroute primitive. This combines tapping and disabling. B is a signal from the control plane commanding the tapping of signal C in the computation plane. A is the tapped signal to the control plane. D is a signal from the control plane commanding the disablement. The gray triangle is a non-restoring tri-state buffer, which directly passes the input to the output but allows noise to propagate through the circuit. The gray triangle with a circle is a restoring tri-state buffer. The white triangle is a buffer, which helps in driving loads from weak inputs and in preventing back-driving, when an output can propagate through to an input. We verified the correct behavior of this primitive in simulation with Spice and ModelSim.

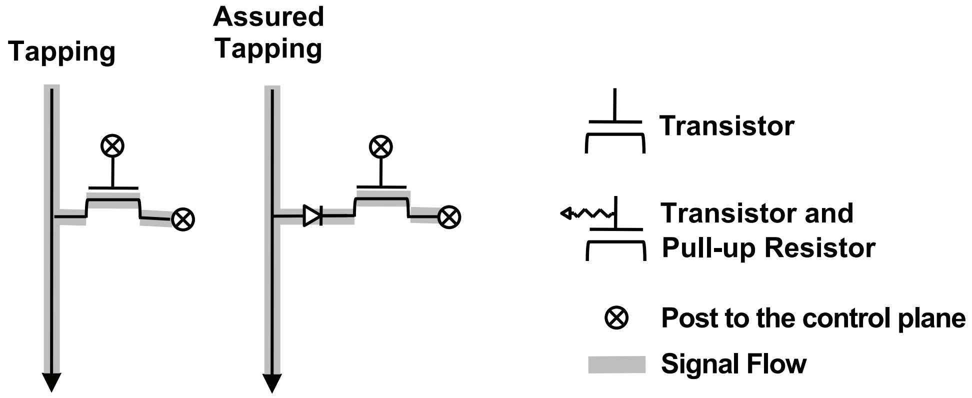


Fig. 9. Primitive TSV functions and corresponding selectively refined functions. The diode is placed between the native (computation plane) circuit and the TSV receptacle, such that the diode controls the flow regardless of how the TSV receptacle is used by the control plane.

6) Diode: A diode, as shown in Figure 9, allows information to flow in only one direction from one component to another[1] [9]. Such an arrangement can enforce a policy requiring that information can flow from a low confidentiality component to a high confidentiality component[2] but not vice versa. In other words, in a system whose resources have been partitioned into policy equivalence classes, a diode can be applied to a primitive to enforce the one-way flow of information between these classes. Diodes provide a granularity of enforcement related to the granularity of components that they connect. Diodes can be static or programmable, and they can ensure that vertical posts do not violate the inter-plane policy, e.g., by placing diodes between a post and the circuitry of a plane. We expect, however, that this hard-wired enforcement can have unforeseen effects, e.g., on the performance of bidirectional communication protocols, and that *programmable diodes* would provide flexibility in the designs supported.

7) Generic TSV Receptacle: A general-purpose TSV receptacle [9] can be used to support multiple control plane applications with the same computation plane, e.g., when different control planes are used or when the applications on a given control plane are reconfigured. These generic TSV receptacles are used to anchor posts on the computation plane to minimize the number of TSV receptacle types that the processor manufacturer must produce and allows a given TSV receptacle to be used for different purposes from application to application. Supporting these features requires identifying standard locations for posts on the computation plane such that generality is balanced against

[1] Note that one-way communication can be enforced using other electrical techniques besides a diode

[2] Given a lattice of confidentiality markings, “high” markings are those that are closer to the top (the universal upper bound), and “low” markings are those that are closer to the bottom (the universal lower bound).

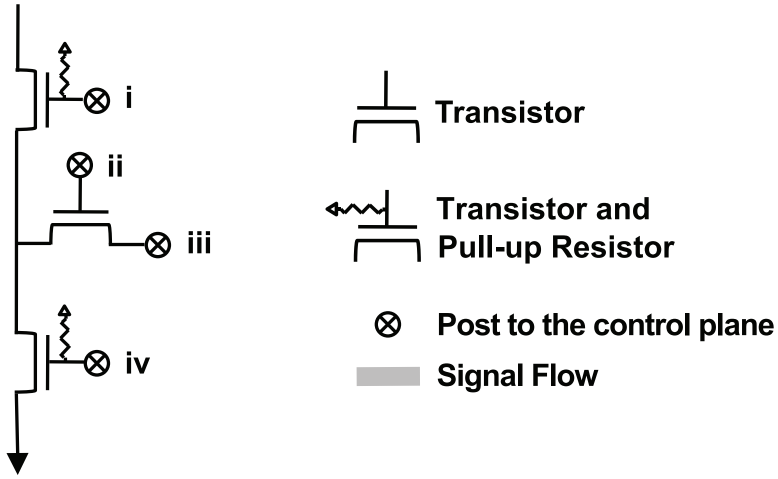


Fig. 10. Generic TSV Receptacle. A general-purpose TSV receptacle supports any of the basic circuit-level primitives (disabling, tapping, rerouting, and overriding). Signals from the control plane determine which primitive is active at a given point in time. Thus, the application on each different control plane could use a given TSV receptacle in a different way: one application might tap a TSV receptacle, and another might override it. The control posts are i, ii, and iv, and the data post is iii.

	1 Generic TSV Receptacle	**128 TSV Receptacles**	**5-Stage MIPS Processor**
Area (Library Area Units)	84.1	10764.8	240,000

TABLE II
AREA OF GENERIC TSV RECEPTACLE IN 90NM TECHNOLOGY NODE

the hardware resources (e.g., posts) required (see future work).

Figure 10 shows a general-purpose TSV receptacle that supports any of the five basic circuit-level primitives. To make use of a TSV receptacle, signals on the control plane determine which primitive is active at a given point in time. Thus, different 3D applications can use the same TSV receptacle in a different way. Only one configuration of the Generic TSV Receptacle is required even though, for example, one application might read from a TSV receptacle, and another might override circuits with it.

A generic TSV receptacle provides design flexibility at the cost of additional circuitry and posts. The Generic TSV Receptacle accepts four posts (three control and one data) implementing four primitive features internally: one tapping or insertion feature and two disabling features. These are combined to implement disabling (i), tapping (ii and iii), rerouting (ii, iii, and iv), overriding (i, ii, and iii), and native (none of the posts), which has no effect. The Generic TSV Receptacle could include *diodes* to assure that the information flows of each post are precisely controlled, in which case the diode for post iii could be programmable to support reading or writing. Figure 11 shows a schematic diagram of Generic TSV Receptacle implementation. Figure 12 shows a schematic diagram of improved Generic TSV Receptacle implementation.

With this newly developed Generic TSV Receptacle, designers of computation planes may have a generic set of posts that they believe will be of use to high assurance system developers, e.g., for use with control lines and bus interfaces. In order to explore the area ramifications of incorporating these Generic TSV Receptacles on the computation plane for all consumers, we synthesized one Generic TSV Receptacle and compared it to a simple five-stage pipelined MIPS processor. The Generic TSV Receptacle and MIPS processor were written in Verilog and synthesized using Synpsys Design Compiler in 90nm technology. Note that the MIPS processor in this experiment is different from the MIPS processor in Section IV-B.

The area of the Generic TSV Receptacle, shown in Table II, is 84.1 AU. This is a very small percentage of the area of the full processor. Even if we needed 128 Generic TSV Receptacles, the additional area added to the MIPS processor is about 4.5%, which is relatively small. This percentage would be even less when adding Generic TSV Receptacles to a large, modern commodity processor that includes structures such as caches, advanced branch predictors, and Floating Point Units.

To verify the correctness of our circuit-level modifications, we developed Spice circuit models for each of the

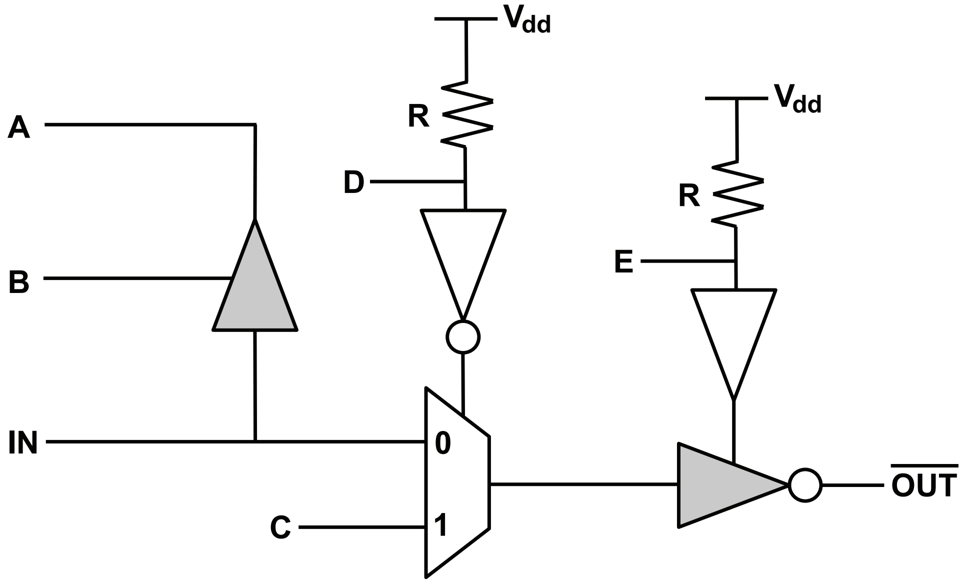


Fig. 11. Schematic diagram of Generic TSV Receptacle implementation, which combines tapping, overriding, and disabling to implement any of the four circuit-level primitives. On the left is the tapping portion of the circuit: the gray triangle connected to A, B, and IN is a non-restoring tri-state buffer, which serves as a switch that B activates. If B is high, then IN flows through to A; however, if B is low, then no signal flows through since the buffer is off. In the middle is the override portion of the circuit: D controls the multiplexer. When D is low, the NOT gate inverts D, and the multiplexer chooses C as an output. The pull-up resistor ensures that the system will output IN even if D is disconnected (i.e., in the absence of the control plane). On the right is the disable portion of the circuit. When E is high, the restoring tri-state buffer (the gray triangle with the circle) is on, and the circuit outputs OUT through the multiplexer. If E is low, the tri-state buffer is off, and the circuit outputs an indeterminate value. This circuit is not assured because of the non-restoring buffer in the tapping portion circuit; a malicious user could drive A and B high, which would create a short, causing unpredictable behavior past the multiplexer. The white triangle is a buffer, and the white triangle with a circle is a NOT gate. We verified the correct behavior of this primitive in simulation with Spice and ModelSim.

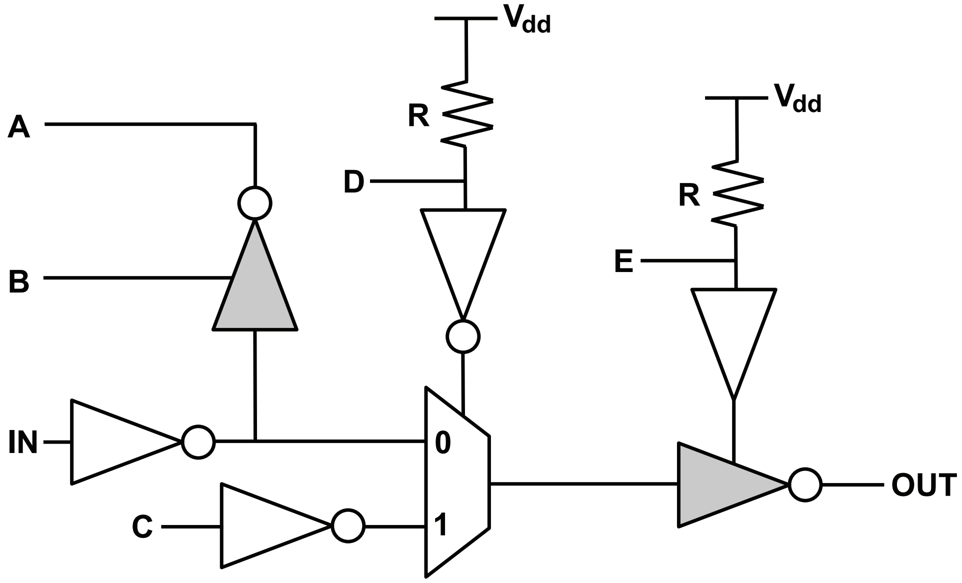


Fig. 12. Schematic diagram of improved Generic TSV Receptacle implementation. The restoring tri-state buffer (depicted as a gray triangle with a circle) in the tapping portion of the circuit prevents a malicious user from ever inserting a signal through the tapping portion of the circuit. The NOT gates (white triangles with circles) also ensure correct operation in the absence of the control plane. While in Figure 11, the output was inverted, the NOT gates in this figure invert C and IN, restoring the output to its correct value of C or IN. The white triangle without a circle is a buffer. We verified the correct behavior of this primitive in simulation with Spice and ModelSim.

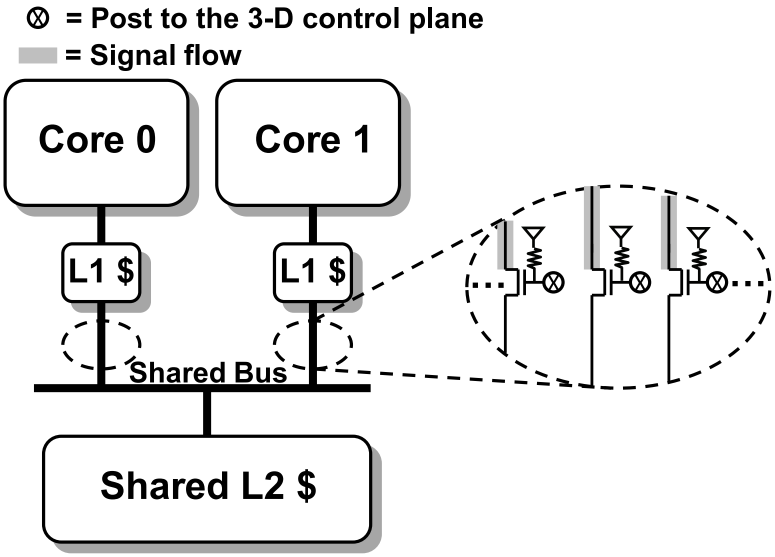


Fig. 13. A multi-core processor with two cores that we wish to isolate. This is achieved using *Disabling* to block each of the bit-level connections to the bus for the core that is not currently allowed to use the bus.

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 control plane and corresponding to the 2) presence of the 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 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 control plane.

IV. 3D Applications

This section describes general categories of 3D security applications and specific examples of a 3D information flow tracking system and a 3D cryptographic coprocessor.

A. Categories of 3D Applications

1) Isolation and Protection: Isolation of active resources is one potential application of our circuit-level primitives. 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 13 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.

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 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 [25] that compresses memory addresses with matching metadata tags can be utilized in the control plane (Figure 14), 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.

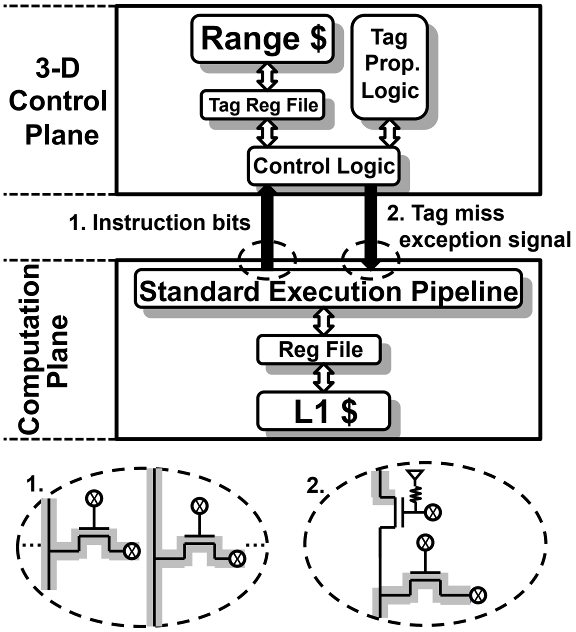


Fig. 14. A 3D system monitor tracking data flow on the computation plane. This is achieved by *Tapping*(1) the instructions that we want to track and *Overriding*(2) to raise an exception signal.

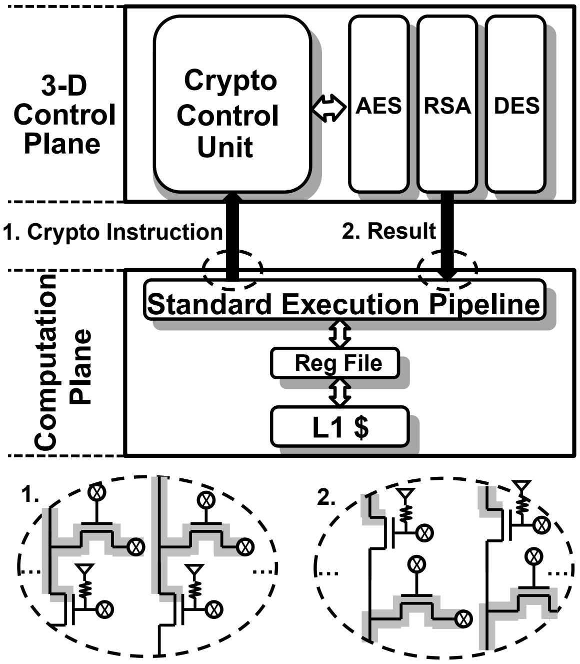


Fig. 15. A 3D cryptographic engine used to perform secure cryptography in the control plane, using *Re-routing*(1) to block the instruction execution on the computation plane and to send the instruction to the control plane to be executed. The result can be placed back in the execution pipeline using *Overriding*(2).

3) 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 control plane [28] that can accept cryptographic instructions being executed on the computation plane, performing the operation immediately before sending the result back to the execution pipeline, as shown in Figure 15. 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 [7], 3D security allows the addition of any cryptographic algorithm or implementation to be included in the system as a foundry-level option. Essentially, 3D security introduces flexibility in the system hardware, allowing any number of cryptographic functions to be optionally added to the processor.

B. MIPS Multi-Cycle Processor

For the following examples, we use a MIPS multi-cycle processor designed in Verilog as the computation plane circuit. The available commands for the processor are:

```
// Jumps to the instruction
// address at [location]
j [location]

// Branches to [location]
// if $a==$b ($a, $b are registers)
beq $a $b [location]

// Adds [imm] to $s and stores
//the value in $d
addi $d $s [imm]

// Operates on $s1 and $s2
// and stores in $d, where
// the operation can be
// add, and, or, sld, or sub
R-type $d $s1 $s2

// Loads memory word 0xaddr
// shifted by $s into $d
lw $d 0xaddr($s)

// Stores register word $d shifted by $s
// into memory address 0xaddr
sw $d 0xaddr($s)
```

Figure 16 shows a diagram of the data path of the MIPS CPU used for the following example applications. Figure 17 shows a state machine of this controller.

C. Security Levels

We use the MIPS CPU described in Section IV-B to construct a system that assigns a two-bit tag, representing one of four possible security levels (e.g., TS, S, C, and U), to every address in memory. The tags are stored in a dedicated region of memory. We attach another circuit, shown in Figure 19, that acts as a regulator, operating in parallel with the MIPS CPU. DIP switches on the Xilinx Virtex-V FPGA development board are used to set the security level of a process that is executing a small program consisting of approximately ten MIPS instructions. The regulator will prevent the process from executing instructions with a higher security level. For example, if a

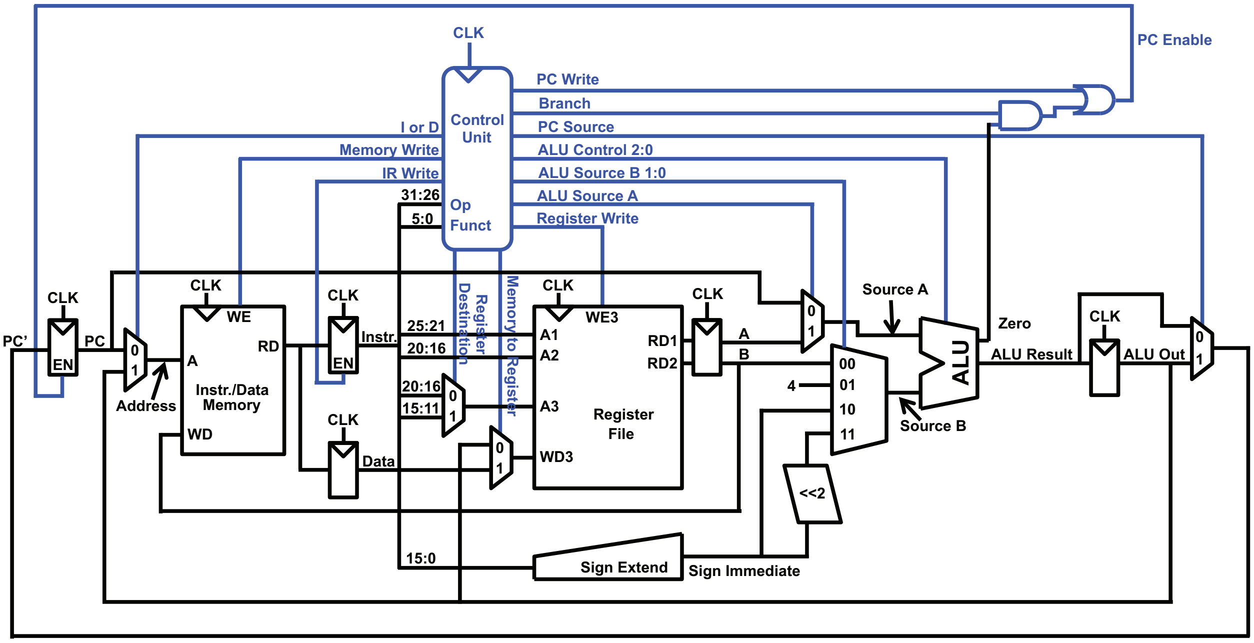


Fig. 16. Diagram of the data path of the MIPS CPU used for example applications *except for the cache eviction monitor example described in Section V*. The gray wires represent the portion of the circuit that controls the MIPS data path, shown in black.

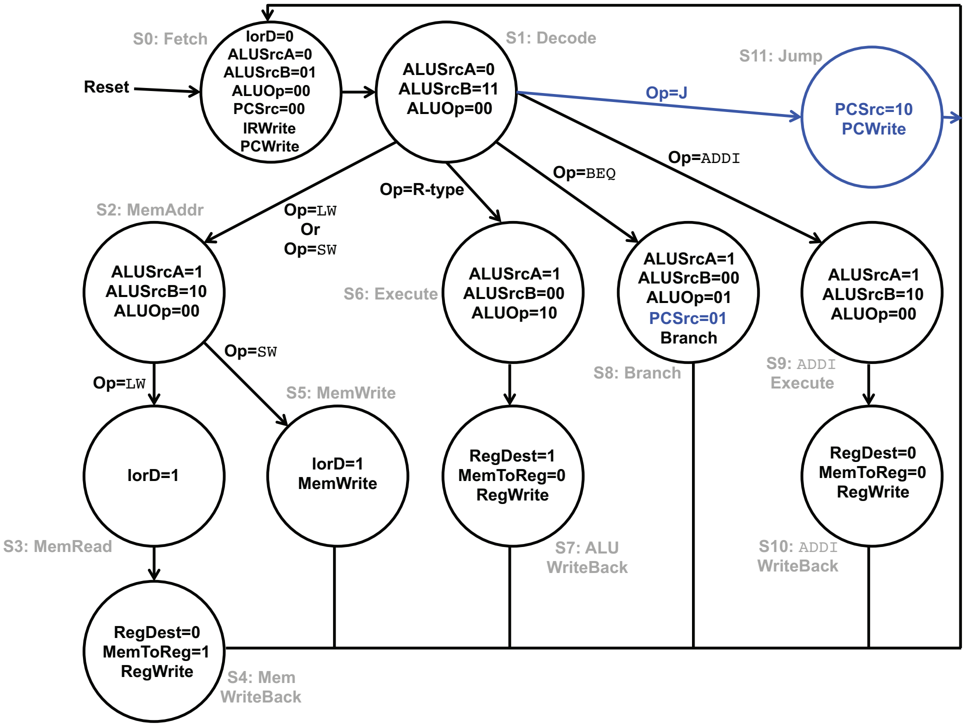


Fig. 17. State machine of the MIPS CPU in Figure 16.

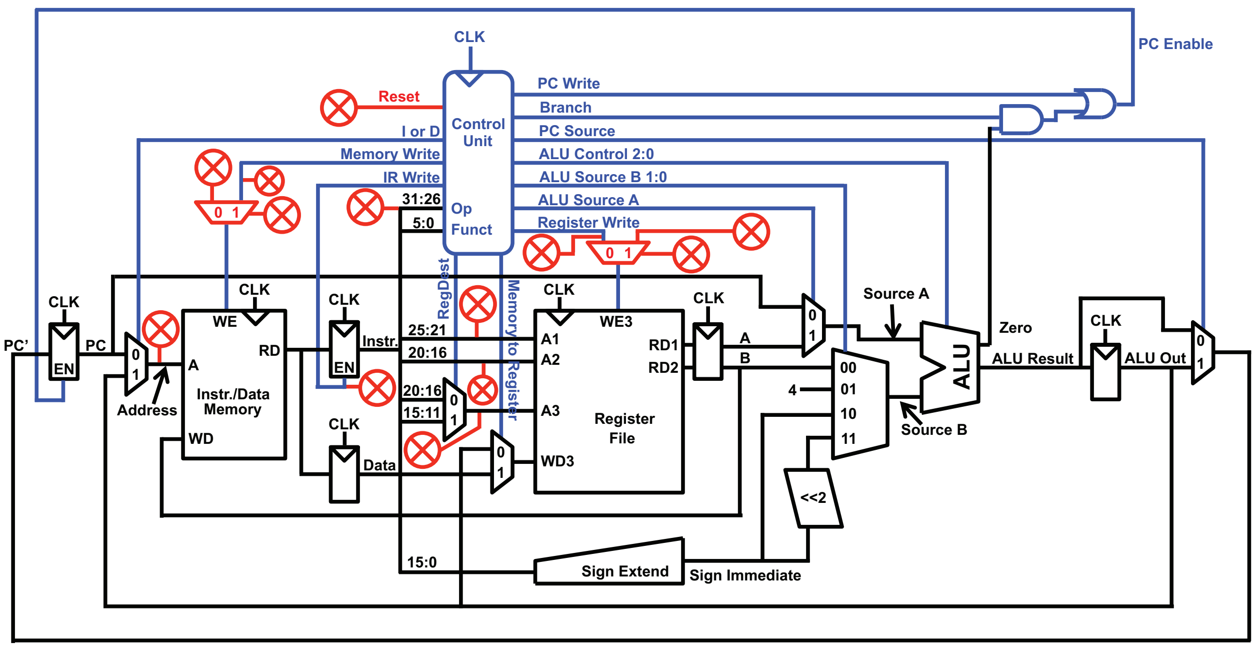


Fig. 18. MIPS CPU with posts added (circles) for attaching the regulator circuit in the control plane.

process at a lower level branches to an instruction with a higher label, the regulator allows the branch but skips over all instructions at the higher level until the program counter reaches an instruction with a label that is equal to or less than the level of the process. I.e., the MIPS CPU must wait until the next available instruction that is at or below its level.

The regulator accomplishes its policy enforcement duties by performing *shadow* computations on the tags in parallel with the actual computation being performed on the MIPS CPU. The regulator has the same control circuitry as the MIPS CPU. In a 3D implementation, vertical posts would send relevant signals (e.g., memory address and instruction "taps") to the regulator circuit. Note that if a process with a higher security label branches to an instruction with a lower label, the instruction label is changed to the higher level, and classification creep may occur, i.e., the tags throughout the system will slowly elevate to that higher level.

The regulator contains a memory bank housing the same addresses as the MIPS CPU. This memory bank contains a tag for each address in the MIPS CPU's memory. While the MIPS CPU is running, the regulator determines (1) whether the current instruction is at or below the current process's security level; (2) whether a memory read/write attempts to access an address whose label is above its level; and (3) whether a process is editing data, in which case the processor will set the label of all of the modified data to the level of the currently executing program. For this application, 3D integration offers high bandwidth, low latency, and the ability to use the entire control plane's area for the regulator circuitry. Figure 18 shows the MIPS CPU with posts added (depicted as circles) for attaching the regulator circuit in the control plane. Figure 19 shows the regulator circuit. Figure 20 shows the state machine for the regulator circuit.

D. 3D Cryptographic Coprocessor

We use the MIPS CPU described in Section IV-B to construct a system that uses a cryptographic coprocessor in the control plane to provide a secure alternate service for providing memory encryption and decryption to the CPU in the computation plane [28]. This system encrypts incoming data writes and decrypts outgoing data reads. It intercepts writes to memory and overrides them with encrypted versions of the data. The cryptographic core monitors data writes and reads, checking the write address against a predetermined list to determine whether it is an instruction. If it is not an instruction, the system encrypts the data into memory. Also, the output data is only decrypted if the incoming read address corresponds to data. The cryptographic coprocessor is a stripped-down AES core from the Open Cores website. It was developed by Rudolf Usselmann from ASICS.ws (Samoa). Figure 22 shows the 3D cryptographic coprocessor. This is attached to the computation plane in Figure 21, which shows

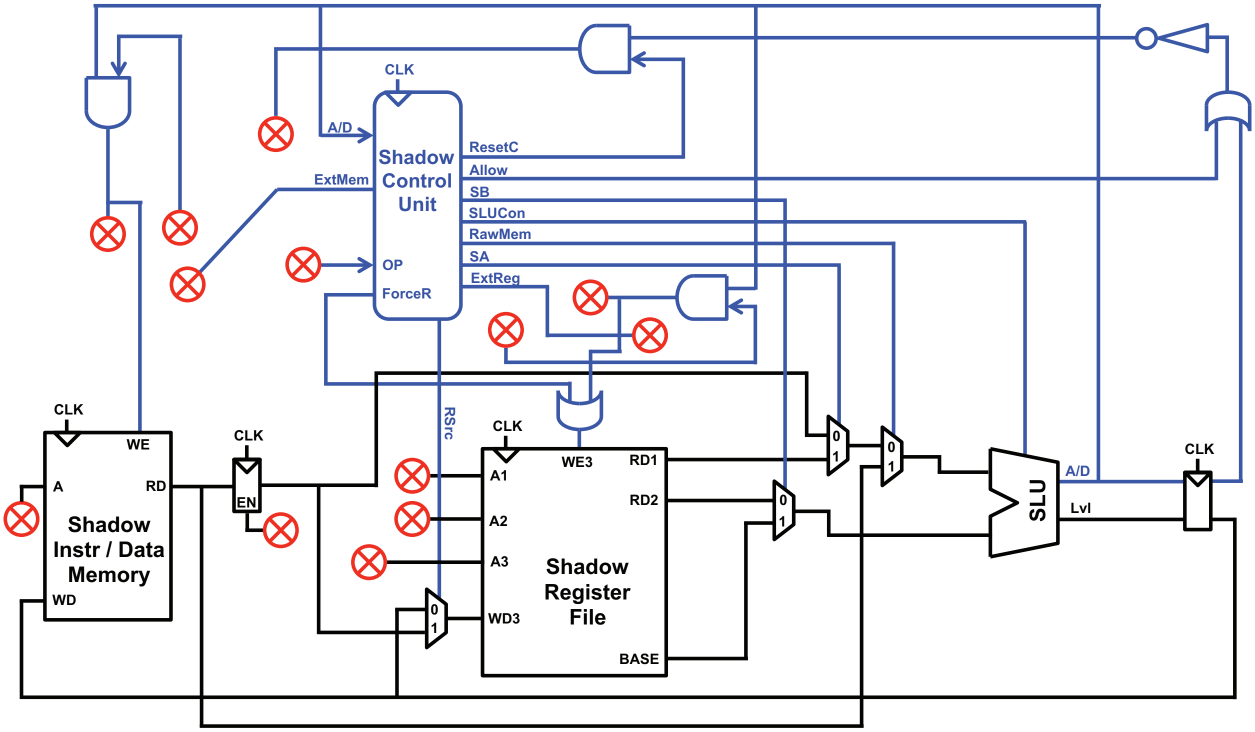


Fig. 19. Diagram of the regulator circuit. The gray wires represent control, and circles represent posts to the MIPS CPU in the computation plane.

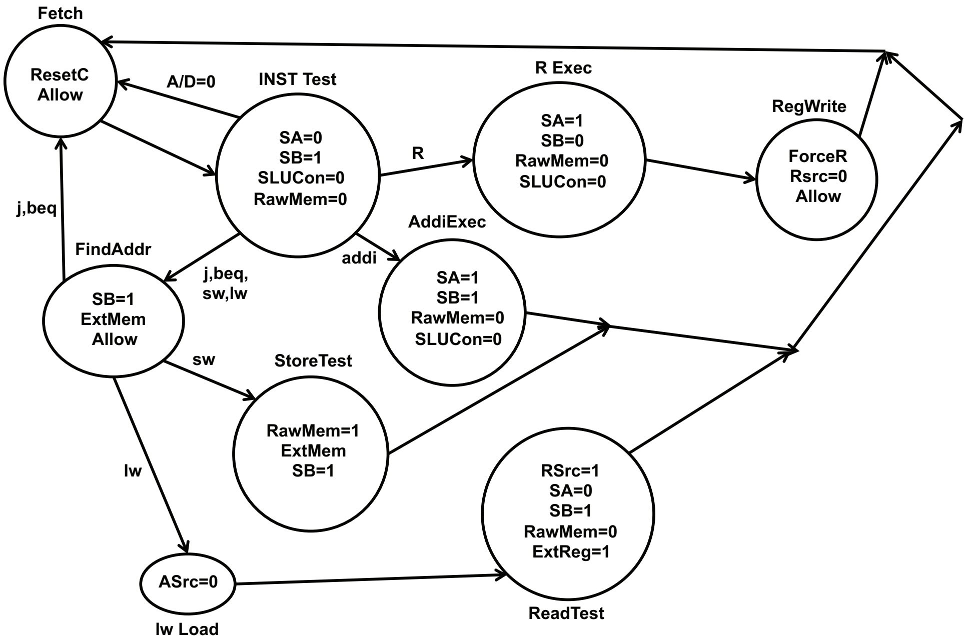


Fig. 20. State machine for the regulator circuit in Figure 19.

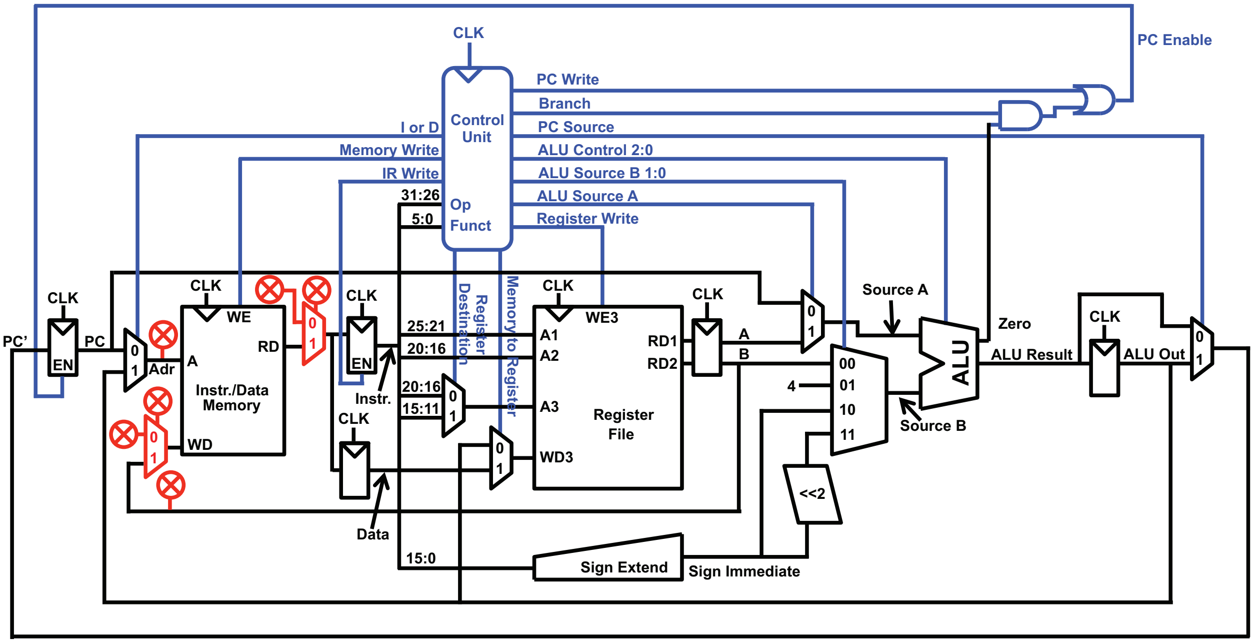


Fig. 21. MIPS CPU modified with posts for optional attachment of a 3D cryptographic coprocessor.

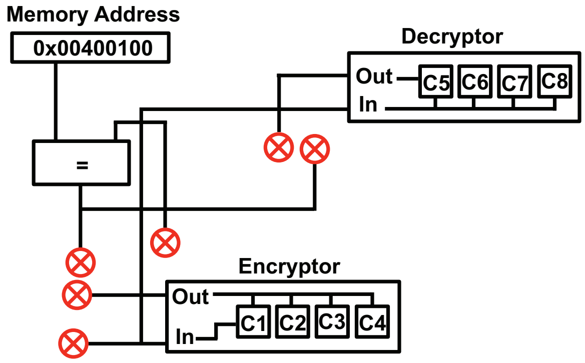


Fig. 22. Cryptographic coprocessor.

the MIPS CPU modified with posts for attaching the crypto coprocessor. We loaded the combined circuit onto a Virtex-V FPGA development board and ran a simple software program on the circuit to demonstrate the system.

V. A 3D Cache Monitor

A. Architecture of a 3D Cache Monitor

This section presents the custom architecture shown in Figure 23, implemented in the 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 [21] 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. An example of this threat was demonstrated by Percival [21], 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 malicious 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

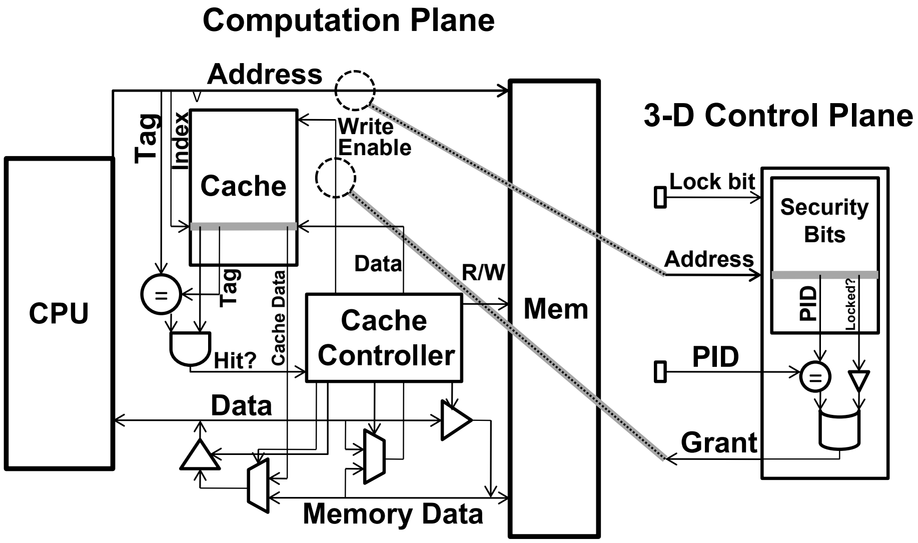


Fig. 23. The architecture of a CPU/cache memory hierarchy and our 3D cache eviction monitor working in concert. The address of the corresponding load/store is tapped to be sent to the control plane, and the cache write-enable signal is overridden in the case of a locked cache line eviction. The Lock bit as well as the Process ID (PID) are also provided to the control plane. We discuss options on how to access this information in Section V. 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.

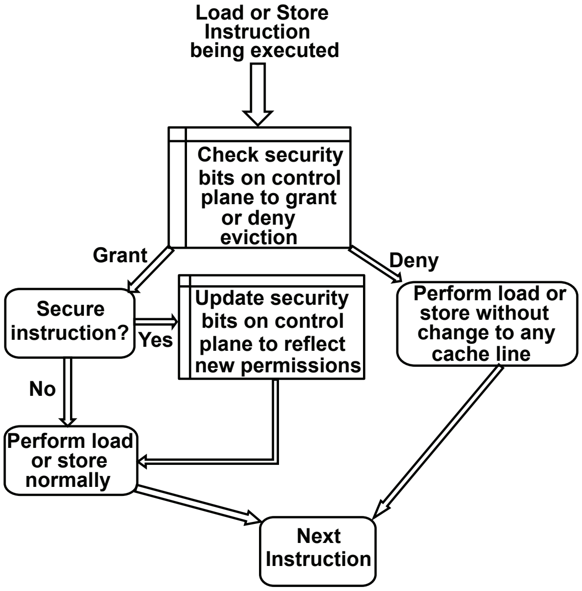

Load or Store Instruction being executed
Check security bits on control plane to grant or deny eviction
Grant
Deny
Secure instruction?
Yes
Update security bits on control plane to reflect new permissions
Perform load or store without change to any cache line
No
Perform load or store normally
Next Instruction

Fig. 24. This flow chart describes how loads and stores are executed when the control plane is in place.

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 derived with a relatively low margin of error.

Our method to prevent these attacks is based on a previously proposed hardware solution [30]. In our application of this scheme, the 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 24 shows a flowchart describing this new protocol, while Figure 25 provides a high-level overview of how the cache and the control plane will interact. Specifically, the cache protection structure contains memory

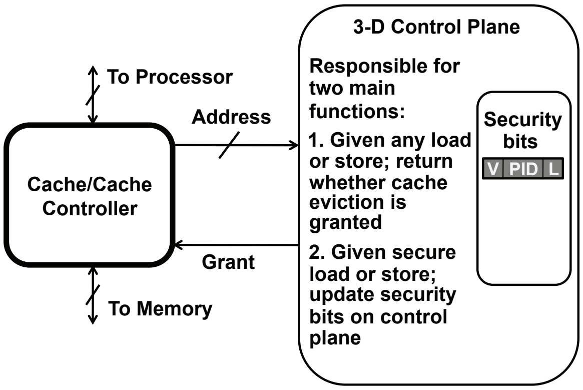

3-D Control Plane
Responsible for two main functions:
1. Given any load or store; return whether cache eviction is granted
2. Given secure load or store; update security bits on control plane
Security bits
V
PID
L
To Processor
Address
Cache/Cache Controller
Grant
To Memory

Fig. 25. A high-level logical overview of how the cache and the control plane interact in our cache monitor, as well as the control plane's responsibilities when active.

elements on the 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 control plane is not attached to the processor, the cache functions as normal. However, when the 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 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.

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 control plane module to determine whether the related cache line is protected from evictions. The security bits on the 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 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. In addition to outlining our synthesis results, we discuss the effect of including the 3D cache eviction monitor, both in terms of critical path and cache performance. We find that the 3D 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 control plane on a sample commodity processor.

B. Performance and Analysis

1) Synthesis Results: In this section, we analyze the performance and area overhead of the 3D cache eviction monitor. We use Altera Quartus to synthesize our design and extract specific timing and area information (Table III). 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 3D 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 3D cache eviction monitor, the maximum frequency remains at 151MHz. The 3D cache eviction monitor synthesized

Design	Max Frequency	Area (LUTs)
Cache/Cache Controller	∼ 151 MHz	468
3-D Cache Eviction Monitor	∼ 217 MHz	291
Cache/Cache Controller With 3D Monitor Attached	∼ 151 MHz	749

TABLE III
THE SYNTHESIS RESULTS PRODUCED BY QUARTUS FOR THE CACHE AND CACHE CONTROLLER, AS WELL AS THE 3D CACHE EVICTION MONITOR.

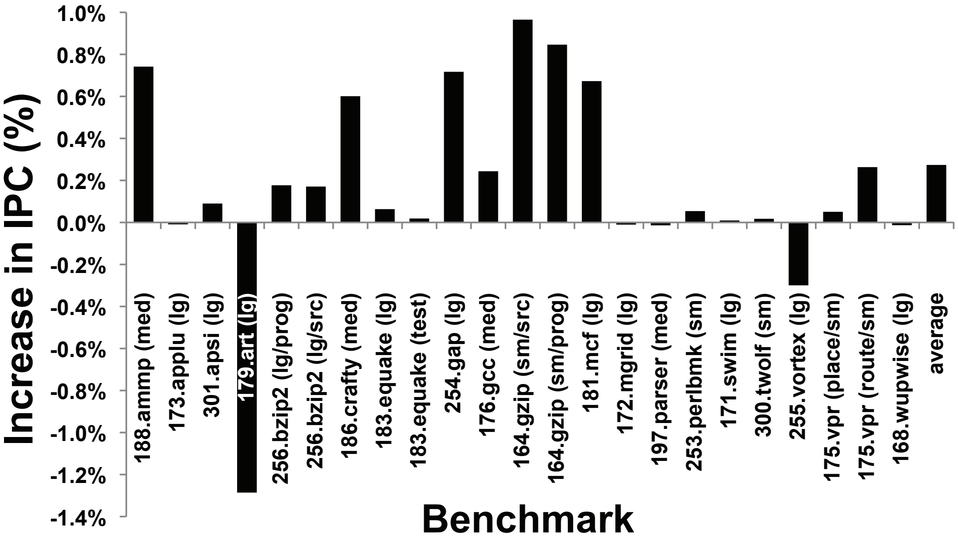


Fig. 26. The results of our cache experiment using SPEC2000 benchmarks that were executed in PTLsim. We use two different sizes of cache to calculate a bound on the performance impact of locking a cryptography program like AES to one way of the cache. The average degradation in IPC between the two different 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.

by itself has a maximum frequency of 217MHz. These maximum frequencies indicate that the critical path in the circuit including the 3D 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 3D cache eviction monitor.

The above performance metrics do not take into account the delay of the vertical posts between the computation plane and the control plane. Loi et al. [16] characterized the worst-case delay of a 3D bus that travels from one corner of a chip to the opposite corner on a 3D layer above, and they found this delay to be about .29ns. Even with the addition of this worst-case bus delay to the 3D 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 3D cache eviction monitor will have no effect on the performance of the cache subsystem.

2) Performance Evaluation: We evaluated the performance impact of locking specific cache lines with our 3D cache eviction monitor. We used PTLsim [32], 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 [4]. With this in mind, 8192 bytes is more than enough to store all of the necessary information (state vectors, round keys, and look-up tables) for AES.

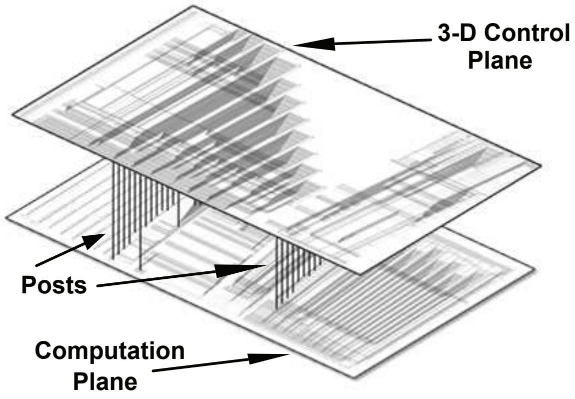


Fig. 27. This figure is a visualization of the physical circuit-level diagram of our 3D cache monitor. Gate-level diagrams were compiled in Quartus after synthesis of the modules.

Results for this experiment can be found in Figure 26. 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. Testing of other programs in the benchmark suite (mesa, galgel, facerec, fma3d, lucas, eon, and sixtrack) is planned for future work.

C. *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 control plane; some vertical posts are also needed to propagate this information to the control plane.

For our 3D 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 [13]. Accessing this register through the vertical posts will give the 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 24.

Finally, the 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 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 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.

Delivery of the previously mentioned required information to the control plane will be through the vertical posts, shown in Figure 27. A general idea of the number of posts the 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.

VI. Related Work

In this section, we discuss other work associated with cache side channel problems. We also discuss related work on the use of 3D technology for security and communication as applied to chip multi-processor (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 [11], [3], [2]. 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) [20], [30]. 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* [18] 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 [4].

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 [19] which makes detailed cache-use profiling difficult; and nondeterministic cache placement [27], [20] or nondeterministic polyinstantiation [6] of cache entries, [30] which, while the specific cause of the interference may be masked, still allows detection of cache misses caused by another process. The 3D 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.

VII. Conclusions

3D integration is a promising technology for developing trustworthy systems, as it 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 also 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 while having minimal impact on performance. We have also developed designs for multilevel security (MLS) and 3D cryptographic coprocessing. This work provides a pathway for the high-assurance community to utilize high-performance hardware while shortening development cycles for trustworthy systems. Future work will involve additional refinement to our designs, with the goal of transforming our RTL designs (e.g., Verilog that can be prototyped on an FPGA) to a format such as GDSII that is ready for fabrication in silicon as a 3D IC.

Acknowledgements

This research was funded by National Science Foundation Grants CNS-0910734, CNS-0910389, and CNS-0910581.

References

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

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

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

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

[5] Bryan Black, Murali Annavaram, Ned Brekelbaum, John DeVale, Lei Jiang, Gabriel H. Loh, Don McCaule, Pat Morrow, Donald W. Nelson, Daniel Pantuso, Paul Reed, Jeff Rupley, Sadasivan Shankar, John Shen, and Clair Webb. Die stacking (3D) microarchitecture. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Orlando, FL, December 2006.

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

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

[8] Ted Huffmire, Timothy Levin, Michael Bilzor, Cynthia E. Irvine, Jonathan Valamehr, Mohit Tiwari, Timothy Sherwood, and Ryan Kastner. Hardware trust implications of 3-D integration. In *Proceedings of the 5th Workshop on Embedded Systems Security (WESS)*, Scottsdale, AZ, October 2010.

[9] Ted Huffmire, Timothy Levin, Cynthia Irvine, Ryan Kastner, and Timothy Sherwood. 3-D extensions for trustworthy systems. In *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA)*, Las Vegas, NV, July 2011.

[10] Intelligence Advanced Research Projects Agency (IARPA). Trusted integrated chips (TIC) program broad agency announcement 11-09, October 2011.

[11] John Kelsey, Bruce Schneier, Chris Hall, and David Wagner. Side channel cryptanalysis of product ciphers. *Journal of Computer Security*, 8(2–3):141–158, 2000.

[12] Dae Hyun Kim, Krit Athikulwongse, Michael B. Healy, Mohammad Hossain, Moongon Jung, Ilya Khorosh, Gokul Kumar, Young-Joon Lee, Dean Lewis, Tzu-Wei Lin, Chang Liu, Shreepad Panth, Mohit Pathak, Minzhen Ren, Guanhao Shen, Taigon Song, Dong Hyuk Woo, Xin Zhao, Joungho Kim, Ho Choi, Gabriel Loh, Hsien-Hsin Lee, and Sung Kyu Lim. 3D-MAPS: 3D massively parallel processor with stacked memory. In *IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, February 2012.

[13] ARM Limited. ARM926EJ-S technical reference manual, 2001-2008.

[14] Gabriel H. Loh. 3-D stacked memory architectures for multi-core processors. In *International Symposium on Computer Architecture (ISCA)*, Beijing, China, June 2008.

[15] Gabriel H. Loh, Yuan Xie, and Bryan Black. Processor design in 3D die-stacking technologies. *IEEE Micro*, 27(3), May-June 2007.

[16] Gian Luca Loi, Banit Agrawal, Navin Srivastava, Sheng-Chih Lin, Timothy Sherwood, and Kaustav 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.

[17] S. Mysore, B. Agrawal, S.C. 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.

[18] Dag Arne Osvik, Adi Shamir, and Eran 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, October 2005.

[19] 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.

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

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

[22] Kiran Puttaswamy and Gabriel H. Loh. Implementing caches in a 3D technology for high performance processors. In *IEEE International Conference on Computer Design (ICCD)*, San Jose, CA, October 2006.

[23] 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.

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

[25] Mohit Tiwari, Banit Agrawal, Shashidhar Mysore, Jonathan K. Valamehr, and Timothy 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.

[26] Mohit Tiwari, Hassan Wassel, Bita Mazloom, Shashidhar Mysore, Frederic Chong, and Timothy Sherwood. Complete information flow tracking from the gates up. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, San Jose, CA, October 2006.

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

[28] Jonathan Valamehr, Ted Huffmire, Cynthia Irvine, Ryan Kastner, Cetin Kaya Koc, Timothy Levin, and Timothy Sherwood. A qualitative security analysis of a new class of 3-D integrated crypto co-processors. *Festschrift Jean-Jacques Quisquarter, Lecture Notes in Computer Science*, 6805, March 2012.

[29] Jonathan Valamehr, Mohit Tiwari, Timothy Sherwood, Ryan Kastner, Ted Huffmire, Cynthia Irvine, and Timothy Levin. Hardware assistance for trustworthy systems through 3-D integration. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, Austin, TX, December 2010.

[30] 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.

[31] Hiroshi Yoshikawa, Atsuko Kawasaki, Tomoaki Iizuka, Yasushi Nishimura, Kazumasa Tanida, Kazutaka Akiyama, Masahiro Sekiguchi, Mie Matsuo, Satoru Fukuchi, and Katsutomu 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.

[32] 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.

Jonathan Valamehr is a PhD student of Electrical and Computer Engineering at the University of California, Santa Barbara in Santa Barbara, California. He is working on several projects that blend computer architecture and security, including the use of 3D integration to provide high-speed security functions, such as mitigating side channel attacks and very high-throughput and secure cryptography. He is a student member of the IEEE and the ACM.

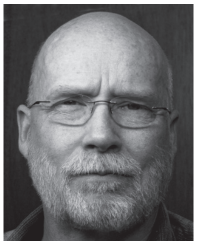
Timothy Sherwood is an Associate Professor of Computer Science at the University of California, Santa Barbara in Santa Barbara, California. His research interests include computer architecture, specifically in the development of novel high-throughput methods by which systems can be constructed, monitored, and analyzed. He has a Ph.D. in Computer Science and Engineering from the University of California, San Diego. He is a member of the IEEE and the ACM.

Ryan Kastner is a Professor of Computer Science and Engineering at the University of California, San Diego in San Diego, California. His research interests focus on many aspects of embedded computing systems, including reconfigurable architectures, digital-signal processing, and security. He has a Ph.D. in Computer Science from the University of California, Los Angeles. He is a member of the IEEE and the ACM.

David Marangoni-Simonsen is an undergraduate student of Electrical Engineering at Harvey Mudd College in Claremont, California. He expects to graduate in May of 2013. He enjoys do-it-yourself projects, including everything from audio visualizers to electro-muscular incapacitating devices. David has worked as an intern at the Naval Postgraduate School for two summers. He is a student member of the IEEE.

Ted Huffmire is an Assistant Professor of Computer Science at the Naval Postgraduate School in Monterey, California. His research spans both computer security and computer architecture, focusing on hardware-oriented security and the development of policy enforcement mechanisms for application-specific devices. He has a Ph.D. in computer science from the University of California, Santa Barbara. He is a member of the IEEE and the ACM.

Cynthia Irvine is the Chair of the Cyber Academic Group, Director of the Center for Information Systems Security Studies and Research (CISR), and Professor of Computer Science at the Naval Postgraduate School in Monterey, California. Her research interests include high-assurance security. She has a Ph.D. in Astronomy from Case Western Reserve University. She is a member of the IEEE, the ACM, and the Astronomical Society of the Pacific.



Timothy Levin is an Associate Research Professor of Computer Science at the Naval Postgraduate School in Monterey, California. His research interests include design, analysis, and verification of high-assurance security architectures and dynamic security policies. He has a B.S. in Computer Science from the University of California, Santa Cruz. He is a member of the IEEE and the ACM.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   8725 John J. Kingman Rd., Ste. 0944
   Ft. Belvoir, Virginia 22060-6218

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California 93943

3. Research Office
   Naval Postgraduate School
   Monterey, CA 93943

4. Jonathan Valamehr
   Department of Electrical and Computer Engineering
   University of California, Santa Barbara
   Santa Barbara, CA 93106

5. Timothy Sherwood
   Department of Computer Science
   University of California, Santa Barbara
   Santa Barbara, CA 93106

6. Ryan Kastner
   Department of Computer Science and Engineering
   University of California, San Diego
   La Jolla, CA 92093

7. David Marangoni Simonsen
   Department of Electrical Engineering
   Harvey Mudd College
   Claremont, CA 91711

8. Ted Huffmire
   Department of Computer Science
   Naval Postgraduate School
   Monterey, CA 93943

9. Cynthia Irvine
   Department of Computer Science
   Naval Postgraduate School
   Monterey, CA 93943

10. Timothy Levin
Department of Computer Science
Naval Postgraduate School
Monterey, CA 93943

11. Jeremy Epstein
National Science Foundation
4201 Wilson Blvd.
Arlington, VA 22230

12. Nina Amla
National Science Foundation
4201 Wilson Blvd.
Arlington, VA 22230

This page left intentionally blank