Hardware Trust Implications of 3-D Integration

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

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

Ryan Kastner
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92903
kastner@cs.ucsd.edu

Abstract

3-D circuit-level integration is a chip fabrication technique in which two or more dies are stacked and combined into a single circuit through the use of vertical electroconductive posts. Since the dies may be manufactured separately, 3-D circuit integration offers the option of enhancing a commodity processor with a variety of security functions. This paper examines the 3-D design approach and provides an analysis concluding that the commodity die system need not be independently trustworthy for the system of joined dies to provide certain trustworthy functions. In addition to describing the range of possible security enhancements (such as cryptographic services), we describe the ways in which multiple-die subsystems can depend on each other, and a set of processing abstractions and general design constraints with examples to address these dependencies.

1. INTRODUCTION

3-D integration is a promising technology for designing high-performance, low-power systems by stacking multiple integrated circuit dies and connecting them at the circuit level with conductive posts. While the history of 3-D technology can be traced back to 1978, it is not yet a mainstream technology, and there are not yet standard protocols and paradigms for composing circuits. Most current efforts are at the electro-mechanical level of getting 3-D to work efficiently and cost effectively.

We use the term *computation plane* to refer to a commodity processor die, and we use the term *control plane* to refer to an additional die, containing customized security functions, that is joined to the computation plane. Since the computation plane must be able to function correctly in the absence of the control plane, circuit-level primitives, described in Section 2.2, are used in conjunction with the posts for communication between the planes. We describe basic classes of 3-D "applications" that can be constructed. However, a first order consideration when building composite secure systems is whether or not the security properties of individual components are preserved under their composition. A classic example is running a "secure" application on an insecure operating system, in which the security properties of the application may be undermined by the OS.

Key aspects of secure composition are self-protection and dependency layering among components. Thus, trustworthy 3-D design requires making smart choices about the use of resources, I/O, and power in order to limit dependencies and ensure the protection of key components. *The question addressed by this paper is, "Can a 3-D control plane provide useful secure services when it is conjoined with an untrustworthy computation plane?"* Design-level investigation of this question yields a definite **yes**. This paper explores 3-D applications and their dependencies, and presents general solutions to resolve these dependencies such that secure 3-D applications and services can be presented from the control plane, taking into account the tradeoffs of each solution.

2. BACKGROUND

Developing customized trustworthy processors is costly to the consumer, largely because there is not a large base of customers over which to amortize the costs. The relatively small market also means that custom designed, trustworthy hardware components are likely to lag their commercial counterparts in performance and other expensive cutting-edge features. Thus, it is attractive for builders of trustworthy systems to use the latest, high-performance, low-cost commercial processors, except where they lack the desired security features that may be available in slower and more costly custom processors. In other words, because of the security shortcomings of commercial processors – e.g., most are manufactured in other countries – critical systems like the Space Shuttle are limited to using components that have lower performance than the gaming consoles used by consumers. We believe that 3-D integration offers the potential of shifting the economics of trustworthy system development, while providing greater design flexibility, to solve this conundrum.

2.1 3-D Basics

Circuit-level 3-D integration is a promising technology for manufacturing integrated circuits [4, 5]. Two or more chip dies are manufactured separately and then bonded together in a vertical stack. Posts, which can be created before or after bonding, provide connections between selected points in the two circuits and carry power and data between them at high bandwidth and ultra-low latency. "Going vertical" reduces the distance between two points of the combined circuit, allowing more transistors to be placed closer to each other. The reduced global interconnect length, and the option to parallelize communication through the use of parallel posts, provides the means to increase the performance and energy efficiency of the design. In addition, the manufacturing technology of each layer can be optimized for different requirements, e.g., in terms of feature size and verification techniques.

The control plane can be fabricated separately (e.g., in a trusted foundry) from the commodity computation plane, offering several advantages. A different lithography process can be used, e.g., 500nm for the control plane vs. 90nm for the commodity computation plane, or vice-versa, depending on performance and economic factors, e.g., using 500nm for a custom die is much cheaper than using the latest technology. The control plane can be subjected to more rigorous design and control practices, based on the customer's needs, than the commodity design of the computation plane. Subsequent security evaluation of the control plane can be independent of the security evaluation of the computation plane. Finally, implementing services in the control plane supports a "layering" design discipline as a way of assuring correct dependencies.

Separating policy enforcement mechanisms (i.e., the trusted computing base, or TCB) from the computation mechanisms (i.e., the target of enforcement) is a fundamental principle in the engineering of trustworthy systems but is a double-edged sword: while a compromise of the non-security system functionality is isolated from the security mechanisms, the adversary can target the security functionality more easily, which is less obscured than it would be entangled with the code and circuitry. Since our security design does not depend on its obscurity we find the *advantages* of separation to be more compelling.

We leverage these 3-D advantages to offer enhanced security in commodity hardware. The basic approach is to slightly modify the design of an existing integrated circuit so that it can accept connections from an optional layer called the *control plane*, without significantly increasing the complexity or cost of the commodity layer, called the *computation plane*. This provides the framework with which the functions, economics, and complexity of security features can be isolated from the underlying computing hardware, and can be managed as customer-selectable fabrication options. Similar to co-processors, the lineage (e.g., venue of manufacture) and developmental assurance of the control plane are also separated from the computation plane.

The interface between a processor and a traditional coprocessor is limited to the established I/O buses and ports, and the long distance severely limits bandwidth and latency. On the other hand, with 3-D integration, the control plane can observe and modify any element in the computation plane at its native granularity, effectively creating interfaces to the computation plane at the selected points in the circuitry, and the physical proximity can provide much higher bandwidth and throughput to the computation plane than can co-processors.

In summary, the key advantages of 3-D integration are (1) high bandwidth and low latency; (2) direct, granular access to chip features; and (3) controlled lineage (e.g., use of a trusted foundry). Additionally, we find the following general advantages: (4) the ability to change the economics of developing critical systems; (5) application-specific security enhancements to commodity hardware; (6) the ability to decouple security and non-security functionality; (7) the ability to create "interfaces" to the commodity processor at chosen locations; (8) the ability to combine independently optimized dies into a single stack; and (9) the ability to reduce delay by locating electrical functions on the control plane close to their counterparts on the computation plane. In addition, all hardware security approaches share the advantages that, when they are designed to do so, they have the ability to operate below the lowest level of the software stack in terms of privilege and dependency; and they can impose strong spatial separation on the software components.

However, 3-D integration is not without cost. The primary issues at this time are thermal and design expense. Pushing integrated circuits into three dimensions naturally adds a vector of heat, and various research efforts [36] are investigating how to deal with thermal concerns, including advanced external heat sinks and internal through-silicon vias (TSVs) dedicated to cooling. Also, the *design* of the commodity computation plane must be analyzed for the optimal placement of posts, and then it must be modified to accommodate the optional control plane.

Other potential problems with 3-D are that some yield loss can occur when joining dies together; although the increase in the cost of fabricating the modified computation plane is marginal. As with any new hardware solution, the designer also bears the brunt of describing how to bridge the semantic gap between low-level abstractions presented by hardware and the higher level semantics of how those abstractions are intended to be used in software.

In the following sections we describe circuit-level primitives

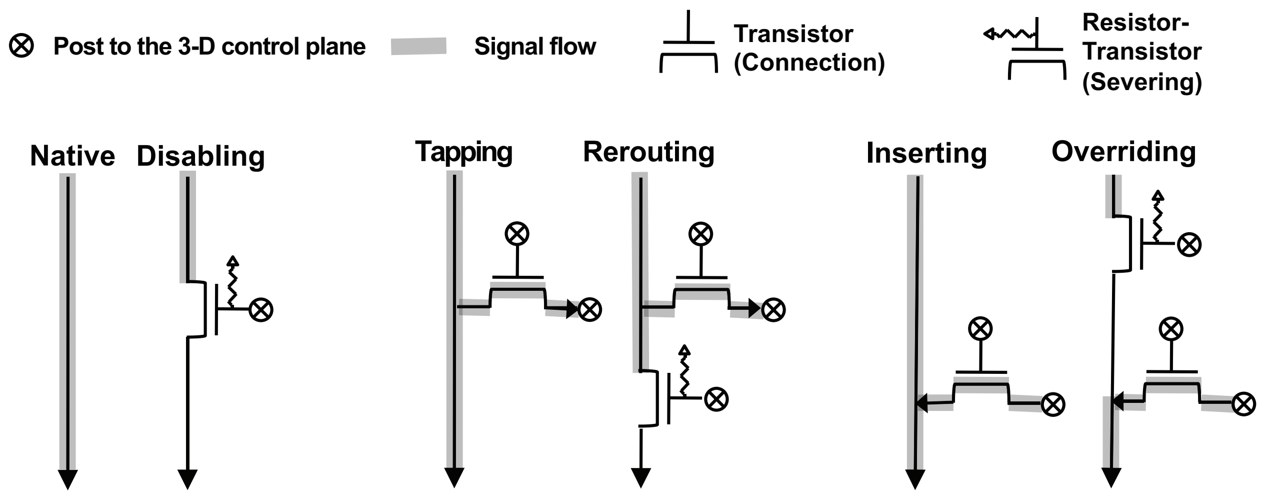

Figure 1: Circuit-level primitives for trustworthy 3-D design [42].

and application design abstractions for 3-D to address this gap and describe what attributes of security *can* be ensured, even if the security of the entire computation plane cannot be ensured.

2.2 Circuit Level Primitives

Five circuit-level primitives are defined for utilizing the conductive posts between the stacked dies [42]. These primitives form the lower-level building blocks of trustworthy 3-D design: disabling, tapping, rerouting, inserting, and overriding, as shown in Figure 1:

- The *disabling* circuit can stop a signal in the computation plane from flowing, based on the control plane's command, which is sent through a dedicated post.
- The *tapping* circuit copies a signal from the computation plane to the control plane. Two posts are needed: one to carry the signal to the control plane and another for the command to connect the signal.
- The *rerouting* circuit combines tapping and disabling so that the original signal only goes to the control plane. Three posts are needed: two for the tapping and one for the disabling.
- The *inserting* circuit carries a signal from the control plane to a circuit on the computation plane. Two posts are needed: one for the signal itself and another for the command to connect the signal.
- The *overriding* circuit combines inserting and disabling, first disabling the original signal in the computation plane and then introducing a new signal from the control plane. Three posts are needed: one for the disabling and two for the inserting.

3. THREAT MODEL

Our threat model includes unintentional flaws in the hardware design of the computation plane and malicious software in the computation plane. The threats of hardware malicious inclusions, physical probing of the control plane, and compromising RF emanations are beyond the scope of this research (see Future Work).

3.1 Hardware Threats

An unintentional flaw in the hardware design of the computation plane is an example of a hardware vulnerability. Our threat model excludes malicious modifications to the design, known as *hardware Trojans* or *Malicious Inclusions (MIs)* [41] which could nullify self-protection of the control plane. Research in this area is discussed under Future Work, but here we assume that the computation plane is not able to selectively reconfigure itself in order to change its behavior.

Our threat model does not include physical tampering or probing of the control plane: we are not proposing the use of 3-D integration to provide physical tamper resistance. We also do not include the threat of side channel attacks arising from compromising radiofrequency (RF) emanations or analysis [27]. While it is conceivable that the control plane could be used to insert RF and power signals, that is not part of our current research. Furthermore, we do not address second order threats such as MI scenarios in which the posts themselves have been bypassed by MIs. Methods to address this threat are described as part of inspection, in future work.

3.2 Software Threats

In the insecure environment of the computation plane, malicious software resulting from compromised source code or compilers can also be executing.

4. SECURITY MODEL

Our premise is that, in a hostile execution environment, secure applications can be constructed if the application can be protected from attack and the application does not depend on any components of lesser trustworthiness. Self-protection is a foundational concept in computer security: e.g., a security kernel would never call one of its applications. Applied to the 3-D context, self-protection requires that the designer not include interfaces that allow the computation plane to reach into the control plane. I.e., the designer must not place a post that allows the control plane to accept extraneous power (e.g., short-circuit or overvoltage), requests (e.g., overflowing buffers), or modifications.

4.1 Importance of Dependencies

It is well known that a component can be no more trusted than the components upon which it depends [28] since the dependency might be unfulfilled at any time. The design paradigm of dependency layering is used in complex systems to prevent looping and to ensure that the most trustworthy components are not undermined through dependence on less trustworthy components, which in the worst case, can reduce the trustworthiness of the entire system to that of the weakest element. Similarly, 3-D applications should not depend on the computation plane. In general, many control plane applications can be designed so as to not request service from, or wait on, the computation plane. Following this design constraint, and similar constraints for self protection, enables control applications to be secure in the presence of an insecure computation plane.

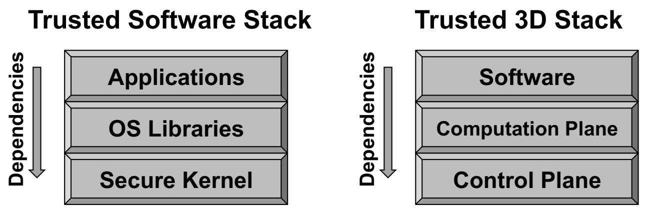

Figure 2: Layered Dependencies

This approach is also used to build secure software, where the security kernel is not allowed to depend on its applications (see Figure 2). Looking at the analogous control-plane/computation-plane layered system it says: what point would it be to build 3-D services if the security of any result is limited by the security of the insecure computation plane? If the computation plane is responsible for providing some service on which the control plane depends, then the control plane may be subverted whenever the computation plane fails to fulfill its responsibilities. Another form of this same question, which is the central theme of this paper, is: is it possible to build *any* highly secure ("trustworthy") service on the control plane when there is a commercial-grade OS on the computation plane?

4.2 Dependency Properties

Several possible forms of dependency exist between planes. Traditional software dependencies include service, call, protection and resource creation/provision. Traditional hardware dependencies include computation, storage, and synchronization. We consider all of these traditional dependencies in the context of 3-D integrated hardware. If the control plane is dependent on the computation plane for power, then untrustworthy functionality could disrupt the supply of power to the control plane, preventing it from operating.

If the thermal characteristics of the two planes are not designed well, the control plane may depend on the computation plane not to overheat the control plane. The thermal challenges of 3-D integrated circuits are well-established [36]. Suppose that a 3-D system is designed such that the device stays within acceptable thermal parameters, provided that the computation plane stays within a given activity profile. In this scenario, the computation plane could perform a large amount of computation, exceeding thermal parameters and causing the chip to fail or even to melt. Even if temperature sensors included in the packaging shut off the entire chip if it gets too hot, the availability dependency still exists even though the chip is prevented from melting. Hadzic described an attack on FPGAs in which a malicious design would short-circuit the device, resulting in its self-destruction [18].

4.2.1 Service Dependency

In a service dependency, the control plane depends on the computation plane to provide a service, such as communication, including off-chip I/O, and synchronization. In an *I/O dependency*, the control plane depends on the computation plane to make available or insert data at an expected time or in an expected order. The confidentiality of this I/O can be eroded because untrustworthy logic in the computation plane could observe the traffic as it passes through the computation plane in violation of the security policy. Dependencies in which the malefactor has access to control plane data can also erode the integrity of communications, processing, and data. Continuing with the off-chip I/O example, untrustworthy logic in the computation plane could modify the traffic as it passes through the computation plane.

Synchronization between the two planes is the coordination of the activities of the two circuits so that they operate together as a unified system. The cooperation of the two dies may be essential to prevent them from interfering with each other. Timing and synchronization dependencies present significant challenges for the 3-D system designer, especially when the two planes are manufactured, using disparate technology, resulting in different electrical properties. Even if the two dies are manufactured using the *same* technology, re-routing a bus in the computation plane to the control plane presents obvious timing issues, such as the small delay caused by introducing posts into the critical path. Clearly, synchronization of the activities of the two dies is critical.

4.2.2 Call Dependency

In a call dependency, the control plane depends on the computation plane to perform a function, such as a mathematical function. This type of dependency can be mitigated by implementing the function (e.g., an ALU) in the control plane [15] rather than in the computation plane, or by validating the correctness of the computation that was performed in the computation plane with a checker unit in the control plane [3]. A *computation dependency* is the same as a call dependency, except that it is a term better suited for the hardware context. A computation dependency can be addressed by implementing the computation hardware in the 3-D control plane. If the computation plane performs operations on data belonging to the control plane, the potential exists for violating the security (integrity) policy.

4.2.3 Resource Creation and Provision, Contention

In a resource creation and provision dependency, the computation plane has a responsibility to create memory objects and tasks and to provision resources to the control plane appropriately. A *contention dependency*, on the other hand, is when a component depends on another component to not use up all of the resources. A *storage dependency* is a type of resource creation and provision dependency that is specific to memory, both on-chip and off-chip. Storage dependencies can be mitigated by implementing the storage in the control plane [8, 16, 29, 35].

5. 3-D APPLICATION CLASSES

This section describes several classes of 3-D applications for the control plane having to do with observing, controlling, and servicing the processes executing on the computation plane. An *inspector* observes, tests, and reports on how the state of the processor and resource usage change over time, with high integrity. A *conflict resolution manager* observes the state of the processor and resource usage over time, and responds by altering certain of the subsequent related actions in order to avoid interference between actions or to prevent actions that would violate e.g. an access policy. A *secure alternate service, or SAS* provides a trustworthy enhancement or alternative alternate to the service provided in the computation plane. Earlier, we discussed six circuit primitives achievable with posts, with which the control plane communicates with the computation plane, enabling the various 3-D applications.

Native functions are normal features provided by the computation plane. The control plane can create applications for the measurement, enhancement, and control of (and alternative to) native functions. *Measurement* of native functions includes audit, performance profiling, and fault logging. *Enhancement* of native functions involves a reconfiguration or repackaging of a native function, including encryption of I/O or routing computation plane bus traffic through the control plane. An *alternative* to native functions (SAS) is an optional runtime choice (e.g., of the program or system) to use control plane features to communicate between computation cores or encrypt data, for example. Finally, *control* of native functions involves, for example, imposing a policy on the use of resources like devices (at the device or device address level) and memory (at different points in the memory hierarchy).

We describe three classes of 3-D applications based on their basic interactions with the computation plane: enhancer or provider of service, imposer of policy or other discipline, and observer of behavior. In addition, the 3-D application can, itself, request service from the computation plane, e.g., for access to resources, which is intrinsically an insecure operation, given an untrustworthy computation plane. In the following sections we describe examples from each class of 3-D application, and describe how each might be constructed to protect itself and to avoid dependencies on the computation plane.

5.1 Secure Alternate Service

An example of a secure alternate service is a cryptographic processor and secure storage implemented in the control plane, like a traditional coprocessor but with much higher bandwidth. Cryptographic keys and other valuable data can be stored and protected in the control plane. Other secure alternate services include computation logic (e.g, ALU, stack, etc.) and processor-core interconnect (e.g., bus and network-on-chip grids), which can be implemented in the control plane either as an addition or enhancement to existing mechanisms in the computation plane or as a duplicate but more highly assured service.

5.1.1 Examples

Examples of secure alternate service include cryptographic transform, key generation, memory cache, and secure general storage including keys. Placing a cryptographic coprocessor on the control plane is another example of a secure alternate service. The bandwidth between a traditional coprocessor connected at the board level to a CPU by a bus is much lower than the bandwidth possible between a crypto coprocessor implemented in the control plane and connected to the commodity computation plane by vertical posts [5, 4, 34].

Cryptographic keys can reside on the control plane rather than in main memory or disk. Keys can be hard-wired, reprogrammable, or based on physical unclonable functions (PUFs) [39]. In addition to the storage of keys, other valuable data can be stored in the control plane rather than the computation plane. This alternate storage service can restrict access to the resource by using encryption where a request for data is accompanied by a key, or the control plane can restrict access by *task ID* or *process ID*, if that information is available to the control plane (e.g., through a post).

In addition to security advantages, implementing functionality in the control plane as an alternative to implementing it in the computation plane also offers benefits for fault tolerance for critical systems. Failure of a computational unit in the computation plane, detected by static tests or runtime checks, can trigger various responses such as shut down and the activation of an identical backup computational unit in the control plane. In the latter case, Circuit-level primitives are used to disable the unit in the computation plane and force signals to take a detour to the unit in the control plane. Designs for 3-D memory [8], ALUs [15], and reconfigurable logic [37] have been devised.

5.1.2 Design Example: Processor Interconnect

Another example of a secure alternate service is the implementation of multicore interconnect (e.g., bus or network-on-chip) in the control plane rather than the computation plane. The security policy may require that two cores be isolated, which may impact the design of the interconnect. Locating an alternate interconnect (which can consume a large percentage of on-chip real estate [10]) in the control plane decouples the interconnect from the commodity design, allowing for the enforcement of an application-specific or dynamic isolation policy for a multi-core computation plane. For example, imposing a secure arbitration scheme on a bus, such as time division multiplexing or lattice scheduling [20]. For systems with many cores, network-on-chip (NoC) technology has been proposed [11]. Implementing the routers in the control plane allows for flexible, adaptable routing policies and has potential for reducing power usage and heat production on the computation plane. Kim et al. have proposed the use of 3-D integration for the implementation of NoC routers [25].

Design Example: For our design example, we pick the simplest possible design: two CPU cores in the computation plane and a bus in the control plane. Each core is connected to the bus by tapping circuits (used for signals going from the cores in the computation plane to the bus in the control plane), and inserting circuits (used for signals going in the other direction). We can design our bus to enforce whatever security policy we wish, e.g., only enabling the bus at certain times or in certain directions (see Figure 3).

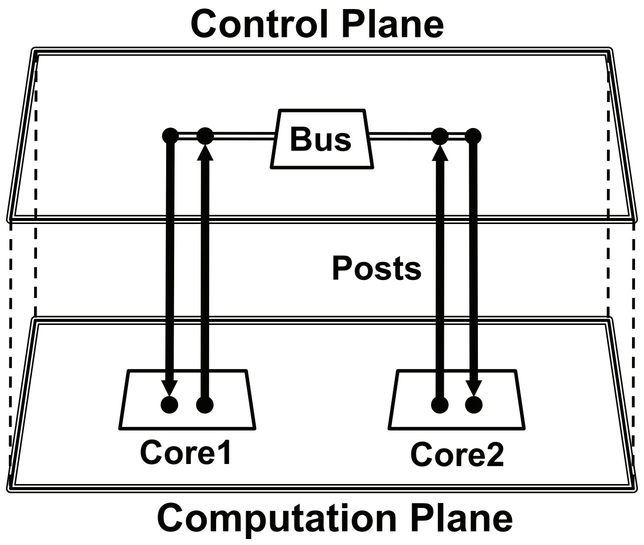

Figure 3: Design example.

Several types of dependency are possible. For a particular bus protocol, a synchronization dependency can occur if the control plane is stuck waiting for a response from the computation plane. This can be mitigated by selecting a bus protocol that gracefully handles the case when a core is abusing the protocol. If the control plane is dependent on the computation plane for power or I/O, a denial of service could result. We discuss mitigation of the I/O and power dependencies in Future Work, e.g., for designs in which denial of service would be critical.

5.2 Isolation and Protection

Another category of control plane application actively overrides the computation plane to enforce some security policy, such as for access control[1]. In one form, this type of mechanism can eliminate *points of interference*[2], such as the cache and branch prediction unit [1]. Disabling connections (e.g., "cutting" wires) to isolate or control the flow of information between cores in a multi-core computation plane (viz., per the reference monitor concept) also falls under this category.

[1]Isolation and access control are distinct concepts. If a system perfectly isolates partitions, access control shouldn't be necessary; however, in the real world, inter-partition flows are required, and access control can be used to check whether or not requested accesses are violating the partition policy.
[2]Having separated the system into partitions, if some interference occurs, it happens on some datum in a particular partition or in the neutral, system area, and that datum (a bit or byte) is the point of interference.

"Points of interference" in the computation plane include architectural elements such as the cache and branch prediction units which are hardware features that the OS must securely multiplex among processors from different protection domains. 3-D control functions that actively override the connections to these shared resources can eliminate their misuse, for example, if one process can observe the cache use of another process, a covert channel can be established between them. A function in the control plane can eliminate this covert channel by overriding the computation plane such that each process is restricted to its designated region of the cache. Side channels have become a vexing problem as new points of interference are realized, e.g., in the cache or branch prediction unit. To solve this requires identification of all of the cut points required to isolate a processor, severing the ones that are not critical and saving and restoring the rest.

Design Example: Returning to our earlier design example, in the context of actively overriding the computation plane to enforce a security policy, we modify our design example slightly: now there is a bus in the computation plane connecting the two cores, and we wish to force the two cores to use the bus in the control plane that enforces our security policy. Instead of using tapping and inserting circuits as before, we use rerouting and overriding circuits. In an alternative scenario, the control plane just has logic for disabling the bus in the computation plane at certain times, according to the policy. In this case, we simply use disabling circuits.

Overriding on-chip interconnect presents significant timing issues. For example, if a signal is expected in a specific time frame, to resolve this issue, synchronization logic could be added to the control and computation planes. However, A synchronization dependency can occur when the control plane depends on the computation plane for synchronization primitives or for promptness or regularity with respect to the bus protocol. If the computation plane is able to interfere with the proper operation of the synchronization primitives (e.g., by slowing down), this can affect the correct operation of the control plane. This can be mitigated by providing the control plane with synchronization primitives that are not dependent on the computation plane.

5.3 Passive Monitoring

Another broad category of control plane application is that which passively monitors the computation plane. These applications may record various properties and values from the computation plane, but do not alter any data or behavior. Examples include auditing and information flow tracking [40, 9], or computing and reporting a checksum on the architectural state of the computation plane to ensure a secure boot sequence, for example [17]. Passive monitoring can also be used to perform runtime checks on the computation plane for security and correctness, enhancing runtime self-tests of the processor and performance profiling.

5.3.1 Information Flow Tracking

A common form of information flow tracking associates a tag with each datum specifying its security attributes [9, 40], and tracking logic, e.g., with *shadow circuits*, is used to follow data through the system. 3-D offers the advantage of relocating the shadow logic to a separate plane rather than attempting to cram everything into a single plane. 3-D integration also offers security advantages since the tags must be immutable. Implementing the tracking logic in the control plane provides better protection of the tags as compared with implementing the shadow logic in the computation plane, although capability architectures have used hardware support to protect the tags by preventing their modification as a distinct data type. Additionally, the control plane can be enhanced to actively respond to certain flow behaviors, by, e.g., stopping the process or preventing access to related data: forms of isolation and protection.

5.3.2 Runtime Correctness Checks

A testing application residing in the control plane can test for design flaws, malicious inclusions, or manufacturing flaws in the computation plane, much like runtime self-tests that execute during idle time. For example, a functional unit such as a multiplier may have a design flaw that causes the result of a multiply to be intermittently incorrect, potentially disrupting asymmetric encryption [33]. Passive monitoring can be used to implement these runtime checks on the correctness of the computation plane. For example, the DIVA architecture uses a small hardware unit that checks the correctness of a much larger, more complex out-of-order processor. [3]. Other runtime checks include program analysis requiring full system data [34, 5, 4], which can benefit from the high bandwidth between the two planes provided by posts. The ability to monitor the processor's internal structure means that, potentially, the entire state space can be captured, and given values can be captured redundantly, e.g., at different points in a circuit, allowing for additional error checking. Passive monitoring could also be used to periodically compute a checksum on the architectural state of the computation plane to detect deviations from known good checksum values [17]. A crypto core in the control plane can compute cryptographic hash values to be used as validation checksums.

5.3.3 Runtime Security Auditing

In addition to runtime checks for correctness and performance optimization, control plane applications based on passive monitoring can also perform runtime security checks, e.g., monitoring every instruction or every memory access in order to detect and report on policy violations for audit. Since 3-D integration provides direct access to the internal state of the processor via the posts at very high bandwidth and low latency, the runtime checks can potentially be carried out more efficiently than using coprocessors [21], virtual machines [14], or binary instrumentation [38].

Design Example: Returning to our earlier design example, in the context of passively monitoring the computation plane, we modify our design example slightly: here we are just monitoring the traffic across the bus in the computation plane, e.g., for the purposes of auditing or performance profiling. In this scenario, tapping circuits are used.

6. RELATED WORK

This paper builds on the work of Valamehr et al., focusing specifically on the fundamental question of whether a 3-D control plane can provide useful secure services when joined with an untrustworthy computation plane, taking into account the requirements of self-protection and dependency layering. Valamehr et al. present the details of the circuit-level modifications that allow the two planes to function correctly when joined, while permitting the computation plane to work properly in the absence of the control plane [42].

While 3-D integration is an emerging technology, several successful designs have been built. Toshiba has mass-produced a Chip Scale Camera Module (CSCM), an image sensor for mobile handsets [44]. An implantable stacked retinal prosthesis chip is a medical application of 3-D integration [22]. Like the human retina, which has separate layers of cells each with its specific function, the stacked retinal chip contains a photodetector layer, a signal processing layer, and a stimulus current generator. High energy physicists have demanding requirements for the pixel sensors used in particle accelerators, for which 3-D integrated designs have been proposed [12]. Georgia Tech researchers have successfully fabricated a 3-D integrated microprocessor with stacked memory [5, 36, 31, 30]. In addition to memory, the implementation of network-on-chip (NoC) routers in a separate 3-D layer has been proposed [25]. Even 3-D Field Programmable Gate Arrays (FPGAs) have been proposed to reduce routing delay for reconfigurable hardware [37]. The work presented here differs from those previous efforts by focusing on security, presenting a taxonomy of 3-D applications and describing abstractions for the secure use of 3-D including active monitoring.

7. FUTURE WORK

In this section we describe preliminary ideas for protecting the control plane from malicious inclusions in the computation plane as well as for mitigating I/O and power dependencies.

7.1 Malicious Inclusions

In the face of compromised hardware, whether intentional or not, it cannot be ensured that a given post implements its intended semantics within a 3-D application, since the semantics of the computation plane are unclear. Detection and prevention of malicious inclusions is an unsolved research question. Detecting any, arbitrary digital modification made by a devious attacker requires a brute-force search, in general. This problem is well-known in software security and is related to undecidability, complexity, and the halting problem. An absence of malicious inclusions might be ensured through a thorough verification and validation (V&V) of the computation plane, although assurance is difficult to obtain through testing alone, since they can be triggered to occur after the V&V stage of development.

Another approach would be to detect *zero day* malicious inclusions through runtime monitoring in the field. This monitoring would seem to be more difficult were it to be implemented in the computation plane, where it would be subject to bypass and attack by the MIs themselves! However, if MI monitoring tools were to be located in the safe haven of the control plane, real-time monitoring for MIs might be effective, if and when we learn to recognize them. We plan to investigate recognition and monitoring techniques, e.g., using approaches discussed in Section 5.3.2, so that the control plane can protect itself from malicious inclusions in the computation plane.

7.2 Off-Chip I/O

The control plane needs to be able to communicate with the outside world to transmit and receive data. In addition, it is necessary to configure the control plane. Different control plane applications will have different off-chip I/O bandwidth and availability requirements. Some control plane applications will not require any I/O. An example of such an application is a static reference monitor or a mechanism that partitions the cache to prevent side channels [42]. While some 3-D applications just require configuration settings during the boot phase of the processor (e.g., loading an executable for a CPU), other 3-D applications require a great deal of off-chip I/O (e.g., real-time program profiling).

7.2.1 Dependency Paradigms

Several structural paradigms are available for implementing off-chip I/O in 3-D chips: (1) Independent I/O for each plane is ideal from a security standpoint but could be costly. (2) The computation plane depends on the control plane. If the control plane is fabricated using a different process than the computation plane (e.g., .5um vs. 45nm), I/O performance may suffer when the dies are joined. (3) The control plane depends on the computation plane. This is the least desirable choice. In addition, various technical solutions may be available in the future for providing independent I/O to the control plane.

7.2.2 Wireless

Wireless options for providing independent I/O to the control plane include capacitive/inductive coupling, short-range RF, short-range optical, and EEPROM. Capacitive and inductive couplings are used to test wafers wirelessly [24]. While inductive coupling permits communication over longer distance, this comes at the cost of area resources and power consumption. Capacitive coupling, on the other hand, is less costly but works over shorter distances. Capacitive coupling can be used to transfer digital data between two dies with high bandwidth and energy efficiency [7]. This scheme will also require modification to the motherboard.

Short-range RF can also be used to transmit information between a die and the motherboard. Sony has developed a short-range link based on millimeter-wave wireless technology capable of 11 gigabits per second at 56 GHz over 14 mm (50 mm with a secondary antenna) [6, 23], which is both compact and energy efficient. Wireless interconnect offers simpler packaging and greater reliability, and its short wavelength is amenable to silicon miniaturization. In addition to requiring modification to the motherboard, this scheme also requires power for transmitting data.

The control plane can house a silicon laser for transmitting data to a component on the motherboard through free space or a fiber optic link [26]. LEDs can also be used as a light source (emitter) [32, 43, 19] and even as a receiver (detector) [13] for bidirectional communication (although duplex communication requires both a transmitter and a receiver). It might not be necessary to encrypt the light beam if physical control over the system is maintained (i.e., if the threat model does not include physical probing). Implementing this scheme will require significant changes to the package and motherboard.

While not truly wireless, an EEPROM chip can be used to program the control plane. The chip is inserted into a socket located on the package of the 3-D chip, and wires connect the socket to the control plane. In the simplest scenario, the EEPROM contains instructions, which are loaded onto the control plane and executed by a CPU. In a more complicated scenario, a general-purpose, secure bootstrapping mechanism orchestrates initialization of EEPROM, CPU, and memory.

7.2.3 Wired

Wired options for providing independent I/O to the control plane include JTAG interface, serial cable, dedicated pins, TDMA over HyperTransport, and dedicated memory ranges. IEEE Standard 1149.1, Standard Test Access Port and Boundary-Scan Architecture, also known as Joint Test Action Group (JTAG), is used for testing circuits [2]. We assume that a similar physical interface can be provided to the control plane, e.g., for initialization. Alternatively, a serial cable can connect the chip to a port on the motherboard or directly to an output port of the machine (e.g., USB). Another option is to dedicate a set of pins to the control plane. If the pin field is saturated, any additional pins dedicated to one plane must be taken away from the other.

To avoid the material and engineering cost of adding extra pins or dedicating some pins for the control plane, one option is to share the pins between the two planes using a Time Division Multiple Access (TDMA) protocol. In this scenario, the computation plane has an I/O controller that uses all of the pins in the absence of the control plane. When the control plane is attached, however, an I/O controller in the control plane overrides the I/O controller in the computation plane. This I/O controller forces the pins to be shared between the two planes such that each plane uses the pins during its assigned time slice. This also potentially reduces modification to the motherboard, since the modified chip operates within the existing communication protocol (e.g., HyperTransport). Modification to other users of the pins would be required to demultiplex the signal.

A similar approach involves tapping the connection between the on-chip components and the pin array used for off-chip memory, and dedicating a range of memory for the exclusive use of the control plane using diversion and injection of the computation plane I/O signals. A hardware reference monitor in the control plane ensures that any attempt by the computation plane to access the range of memory dedicated to the control plane is denied.

7.3 Power

Similar to off-chip I/O, the control plane needs an independent source of power to avoid denial of service attacks from the computation plane. Just as the circuit-level primitives can disable, tap, re-route, insert, and override computation signals, these same primitives can also be used for power (e.g., to reroute power from the computation plane to the control plane). Near-field wireless power transmission technology can also be used to transmit power to the control plane. In addition, in the future, advanced packaging techniques may be able to provide independent power to the control plane.

8. CONCLUSION

3-D integration is a promising technology for the development of trustworthy systems. In this paper, we have described several classes of applications for the control plane. We have also described the ways in which the control plane depends on the computation plane, and we have described the conditions necessary for implementing 3-D applications that can behave securely despite the untrustworthy nature of the computation plane, or that can improve the security of the computation plane. 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.

The detection and protection from malicious inclusions on the computation plane is a complex problem that is not within the scope of this research. To be clear, 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. We describe potential future work regarding malicious inclusions and independent I/O. Additionally we anticipate that 3-D may prove to be a useful approach to provide fault-tolerant chips for critical systems.

9. ACKNOWLEDGEMENTS

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

10. REFERENCES

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

[2] I. S. Association. IEEE standard test access port and boundary-scan architecture-description. Technical Report IEEE Std 1149.1-2001, Institute of Electrical and Electronics Engineers, Piscataway, NJ, 2001.

[3] T. M. Austin. DIVA: A reliable substrate for deep submicron microarchitecture design. In *Proceedings of the 32nd international symposium on microarchitecture (MICRO-32)*, Haifa, Israel, November 1999.

[4] J. Baliga. Chips go vertical. *IEEE Spectrum*, 41(3), March 2004.

[5] B. Black, M. Annavaram, N. Brekelbaum, J. DeVale, L. Jiang, G. H. Loh, D. McCaule, P. Morrow, D. W. Nelson, D. Pantuso, P. Reed, J. Rupley, S. Shankar, J. Shen, and C. Webb. Die stacking (3D) microarchitecture. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Orlando, FL, December 2006.

[6] J. Boyd. Walkie-talkies for chips. *IEEE Spectrum*, April 2010.

[7] R. Cardu, M. Scandiuzzo, S. Cani, L. Perugini, E. Franchi, R. Canegallo, and R. Guerrieri. Chip-to-chip communication based on capacitive coupling. In *Proceedings of the IEEE International Conference on 3D System Integration*, San Francisco, CA, September 2009.

[8] X. Chen and W. R. Davis. Delay analysis and design exploration for 3D SRAM. In *Proceedings of the IEEE International Conference on 3D System Integration*, San Francisco, CA, September 2009.

[9] J. R. Crandall and F. T. Chong. Minos: Control data attack prevention orthogonal to memory model. In *Proceedings of the 37th annual IEEE/ACM International Symposium on Microarchitecture*, Portland, OR, December 2004.

[10] W. J. Dally. Computer architecture is all about interconnect (it is now and it will be more so in 2010). In *Eighth International Symposium on High-Performance Computer Architecture (HPCA) Panel: What Will Have the Greatest Impact in 2010: The Processor, the Memory, or the Interconnect?*, Boston, MA, February 2002.

[11] W. J. Dally and B. Towles. Route packets, not wires: on-chip interconnection networks. In *Proceedings of the 38th Design Automation Conference (DAC)*, Las Vegas, NV, June 2001.

[12] M. Demarteau, Y. Arai, H.-G. Moser, and V. Re. Developments of novel vertically integrated pixel sensors in the high energy physics community. In *IEEE International Conference on 3D System Integration*, San Francisco, CA, September 2009.

[13] P. Dietz, W. Yerazunis, and D. Leigh. Very low-cost sensing and communication using bidirectional LEDs. In *Proceedings of the International Conference on Ubiquitous Computing*, Seattle, WA, October 2003.

[14] G. W. Dunlap, S. T. King, S. Cinar, M. A. Basrai, and P. M. Chen. ReVirt: enabling intrusion analysis through virtual-machine logging and replay. In *Proceedings of the Symposium on Operating Systems Design and Implementation (OSDI)*, Boston, MA, December 2002.

[15] R. Egawa, J. Tada, H. Kobayashi, and G. Goto. Evaluation of fine grain 3-D integrated arithmetic units. In *Proceedings of the IEEE International Conference on 3D System Integration*, San Francisco, CA, September 2009.

[16] Y. Funaya, R. Egawa, H. Takizawa, and H. Kobayashi. 3D on-chip memory for the vector architecture. In *Proceedings of the IEEE International Conference on 3D System Integration*, San Francisco, CA, September 2009.

[17] B. Glas, A. Klimm, O. Sander, K. Muller-Glaser, and J. Becker. A system architecture for reconfigurable trusted platforms. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*, Munich, Germany, March 2008.

[18] I. Hadzic, S. Udani, and J. M. Smith. FPGA viruses. In *Proceedings of the Ninth International Workshop on Field-Programmable Logic and Applications (FPL)*, Glasgow, UK, August 1999.

[19] S. K. Hashemi, Z. Ghassemlooy, L. Chao, and D. Benhaddou. Orthogonal frequency division

multiplexing for indoor optical wireless communications using visible light LEDs. In *Proceedings of the Symposium on Communication Systems, Networks, and Digital Signal Processing (CSNDSP)*, Graz, Austria, July 2008.

[20] W.-M. Hu. Lattice scheduling and covert channels. In *Proceedings of the 1992 IEEE Symposium on Security and Privacy*, Oakland, CA, May 1992.

[21] N. L. P. Jr., T. Fraser, J. Molina, and W. A. Arbaugh. Copilot: A coprocessor-based kernel runtime integrity monitor. In *Proceedings of the USENIX Security Symposium*, San Diego, CA, August 2004.

[22] Y. Kaiho, Y. Ohara, H. Takeshita, K. Kiyoyama, K.-W. Lee, T. Tanaka, and M. Koyanagi. 3D integration technology for 3D stacked retinal chip. In *IEEE International Conference on 3D System Integration*, San Francisco, CA, September 2009.

[23] K. Kawasaki, Y. Akiyama, K. Komori, M. Uno, H. Takeuchi, T. Itagaki, Y. Hino, Y. Kawasaki, K. Ito, and A. Hajimiri. A millimeter-wave intra-connect solution. In *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, February 2010.

[24] G.-S. Kim, M. Takamiya, and T. Sakurai. Capacitive coupling interface with high sensitivity for wireless wafer testing. In *Proceedings of the IEEE International Conference on 3D System Integration*, San Francisco, CA, September 2009.

[25] J. Kim, C. Nicopoulos, D. Park, R. Das, Y. Xie, N. Vijaykrishnan, M. S. Yousif, and C. R. Das. A novel dimensionally-decomposed router for on-chip communication in 3D architectures. In *Proceedings of the 34th International Symposium on Computer Architecture*, San Diego, CA, June 2007.

[26] B. R. Koch, A. W. Fang, H.-H. Chang, H. Park, Y.-H. Kuo, R. Jones, O. Cohen, O. Raday, M. J. Paniccia, and J. E. Bowers. A 40 GHz mode locked silicon evanescent laser. In *Proceedings of the International Conference on Group IV Photonics*, Tokyo, Japan, September 2007.

[27] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Proceedings of the 19th Annual International Cryptology Conference (CRYPTO)*, Santa Barbara, CA, August 1999.

[28] T. E. Levin, C. E. Irvine, T. V. Benzel, G. Bhaskara, P. C. Clark, , and T. D. Nguyen. Design principles and guidelines for security. Technical Report NPS-CS-07-014, Naval Postgraduate School, Monterey, CA, November 2007.

[29] D. L. Lewis and H.-H. S. Lee. Architectural evaluation of 3D stacked RRAM caches. In *Proceedings of the IEEE International Conference on 3D System Integration*, San Francisco, CA, September 2009.

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

[31] G. H. Loh, Y. Xie, and B. Black. Processor design in 3D die-stacking technologies. *IEEE Micro*, 27(3), May-June 2007.

[32] J. Loughry and D. A. Umphress. Information leakage from optical emanations. *ACM Transactions on Information and System Security (TISSEC)*, 5(3), August 2002.

[33] J. Markoff. Adding math to list of security threats. *The New York Times*, November 2007.

[34] S. Mysore, B. Agrawal, N. Srivastava, S.-C. Lin, 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.

[35] E. C. Oh and P. D. Franzon. Technology impact analysis for 3D TCAM. In *Proceedings of the IEEE International Conference on 3D System Integration*, San Francisco, CA, September 2009.

[36] K. Puttaswamy and G. Loh. Thermal analysis of a 3D die-stacked high-performance microprocessor. In *Proceedings of the 16th ACM Great Lakes Symposium on VLSI (GLSVLSI'06)*, Philadelphia, PA, May 2006.

[37] S. A. Razavi, M. S. Zamani, and K. Bazargan. A tileable switch module architecture for homogeneous 3D FPGAs. In *Proceedings of the IEEE International Conference on 3D System Integration*, San Francisco, CA, September 2009.

[38] D. Song, D. Brumley, H. Yin, J. Caballero, I. Jager, M. G. Kang, Z. Liang, J. Newsome, P. Poosankam, and P. Saxena. BitBlaze: A new approach to computer security via binary analysis. In *Proceedings of the International Conference on Information Systems Security (ICISS)*, Hyderabad, India, December 2008.

[39] G. E. Suh and S. Devadas. Physical unclonable functions for device authentication and secret key generation. In *Proceedings of the 44th annual Design Automation Conference (DAC)*, San Diego, CA, June 2007.

[40] G. E. Suh, J. W. Lee, D. Zhang, and S. Devadas. Secure program execution via dynamic information flow tracking. In *Proceedings of the 11th international conference on Architectural support for programming languages and operating systems (ASPLOS)*, Boston, MA, October 2004.

[41] M. Tehranipoor and F. Koushanfar. A survey of hardware Trojan taxonomy and detection. *IEEE Design & Test of Computers*, 27(1), Jan./Feb. 2010.

[42] J. Valamehr, M. Tiwari, T. Sherwood, A. Arfaee, R. Kastner, T. Huffmire, C. Irvine, and T. Levin. Hardware assistance for trustworthy systems through 3-D integration. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, Austin, TX, December 2010.

[43] J. Vucic, C. Kottke, S. Nerreter, K. Habel, A. Buettner, K.-D. Langer, and J. W. Waleski. 230 MB/s via a wireless visible-light link based on OOK modulation of phosphorescent white LEDs. In *Proceedings of the Optical Fiber Communication Conference and Exposition (OFCNFOEC)*, San Diego, CA, March 2010.

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