2011

# 3D Execution Monitor (3D-EM): Using 3D Circuits to Detect Hardware Malicious Inclusions in General Purpose Processo

Bilzor, Michael

# 3D Execution Monitor (3D-EM): Using 3D Circuits to Detect Hardware Malicious Inclusions in General Purpose Processors

CDR Michael Bilzor
U.S. Naval Postgraduate School, Monterey, California, USA

**Abstract:** Hardware malicious inclusions (MIs), or "hardware trojans," are malicious artifacts planted in microprocessors. They present an increasing threat to computer systems due to vulnerabilities at several stages in the processor manufacturing and acquisition chain. Existing testing techniques, such as side-channel analysis and test-pattern generation, are limited in their ability to detect malicious inclusions. These hardware attacks can allow an adversary to gain total control over a system, and are therefore of particular concern to high-assurance customers like the U.S. Department of Defense.

In this paper, we describe how three-dimensional (3D) multi-layer processor fabrication techniques can be used to enhance the security of a target processor by providing secure off-chip services, monitoring the execution of the target processor's instruction set, and disabling potentially subverted control circuits in the target processor.

We propose a novel method by which some malicious inclusions, including those not detectable by existing means, may be detected and potentially mitigated in the lab and in fielded, real-time operation. Specifically, a target general-purpose processor, in one layer, is joined using 3D interconnects to a separate layer, which contains an Execution monitor for detecting deviations from the target processor's specified behavior. The Execution monitor layer is designed and fabricated separately from the target processor, using a trusted process, whereas the target processor may be fabricated by an untrusted source. For high-assurance applications, the monitor layer may be joined to the target layer, after each has been separately fabricated.

In the context of existing computer security theory, we discuss the limits of what an Execution monitor can do, and describe how one might be constructed for a processor. Specifically, we propose that the signals which carry out the target processor's instruction set actions may be described in a stateful representation, which serves as the input for a finite automata-based Execution monitor, whose acceptance predicate indicates when the target processor's behavior violates its specification. We postulate a connection between Execution monitor theory and the proposed 3D processor monitoring system, which can be used to detect a specific class of malicious inclusions.

Finally, we present the results of our first monitor experiment, in which we designed and tested (in simulation) a simple Execution monitor for a small open-source 32-bit processor design known as the ZPU. We analyzed the ZPU processor to determine which signals must be monitored, designed a system of monitor interconnects in the hardware description language (HDL) representation, developed a stateful representation of the microarchitectural behavior of the ZPU, and designed an Execution monitor for it. We demonstrated that the Execution monitor identifies correct operation of the original, unmodified ZPU, as it executed arbitrary code. Having introduced some minor deviations to the ZPU processor's microarchitectural design, we then showed in simulation that the Execution monitor correctly detected the deviations, in the same way that it might detect the presence of some malicious inclusions in a modern processor.

## 1. The Threat to Microprocessors

Today's Defense Department relies on advanced microprocessors for its high-assurance needs. Those applications include everything from advanced weaponry, fighter jets, ships, and tanks, to satellites and desktop computers for classified systems. Much attention and resources have been devoted to securing the software that runs these devices and the networks on which they communicate. However, two significant trends make it increasingly important that we also focus on securing the underlying hardware that runs these high-assurance devices. The first is the U.S.' greater reliance on processors produced overseas. The second is the increasing ease with which hardware may be maliciously modified and introduced into the supply chain.

Every year, more microprocessors destined for U.S. Department of Defense (DoD) systems are manufactured overseas, and fewer are made inside the U.S. As a result, there is a greater risk of processors being manufactured with malicious inclusions (MIs), which could compromise high-assurance systems. This concern was highlighted in a 2005 report by the Defense Science Board, which noted a continued exodus of high-technology fabrication facilities from the U.S. (Defense Science Board 2005). Since this report, "more U.S. companies have shifted production overseas, have sold or licensed high-end capabilities to foreign entities, or have exited the business." (McCormack 2008) One of the Defense Science Board report's key findings reads, "There is no longer a diverse base of U.S. integrated circuit fabricators capable of meeting trusted and classified chip needs." (Defense Science Board 2005)

Today, most semiconductor *design* still occurs in the U.S., but some design centers have recently developed in Taiwan and China (Yinung 2009). In addition, major U.S. corporations are moving more of their front-line fabrication operations overseas for economic reasons:

- "Press reports indicate that Intel received up to $1 billion in incentives from the Chinese government to build its new front-end fab in Dalian, which is scheduled to begin production in 2010." (Nystedt 2007)
- "Cisco Systems has pronounced that it is a 'Chinese company,' and that virtually all of its products are produced under contract in factories overseas." (McCormack 2008)
- "Raising even greater alarm in the defense electronics community was the announcement by IBM to transfer its 45-nanometer bulk process integrated circuit technology to Semiconductor Manufacturing International Corp., which is headquartered in Shanghai, China. There is a concern within the defense community that it is IBM's first step to becoming a 'fab-less' semiconductor company." (McCormack 2008)

Since modern processors are designed in software, the processor design plans become a potential target of attack. Malicious logic can also be inserted after a chip has been manufactured, such as with focused ion beam milling (Adee 2009).

Though reports of actual malicious inclusions are often classified or kept quiet for other reasons, some reports do surface, like this unverified account (Adee 2009):

> According to a U.S. defense contractor who spoke on condition of anonymity, a 'European chip maker' recently built into its microprocessors a "kill switch" that could be accessed remotely. French defense contractors have used the chips in military equipment, the contractor told IEEE Spectrum. If in the future the equipment fell into hostile hands, 'the French wanted a way to disable that circuit,' he said.

According to the New York Times, such a "kill switch" may have been used during the 2007 Israeli raid on a suspected Syrian nuclear facility under construction (Markoff 2009).

## 2. Characterizing Processor Malicious Inclusions

Several academic research efforts have demonstrated the insertion of MIs into general-purpose processor designs. In one example, King, et al., show how a very small change in the design of a processor facilitates "escalation-of-privilege" and "shadow mode" attacks, each of which can allow an adversary to gain arbitrary control over the targeted system (King 2009). In another example, Jin, et al., show how small, hard-to-detect MIs can allow an adversary to gain access to a secret encryption key (Jin 2009).

Researchers have created various taxonomies of MIs, based on their characteristics. One example comes from Tehranipoor and Koushanfar (Tehranipoor 2010), from which the following simplified diagram (Figure 1) is derived:

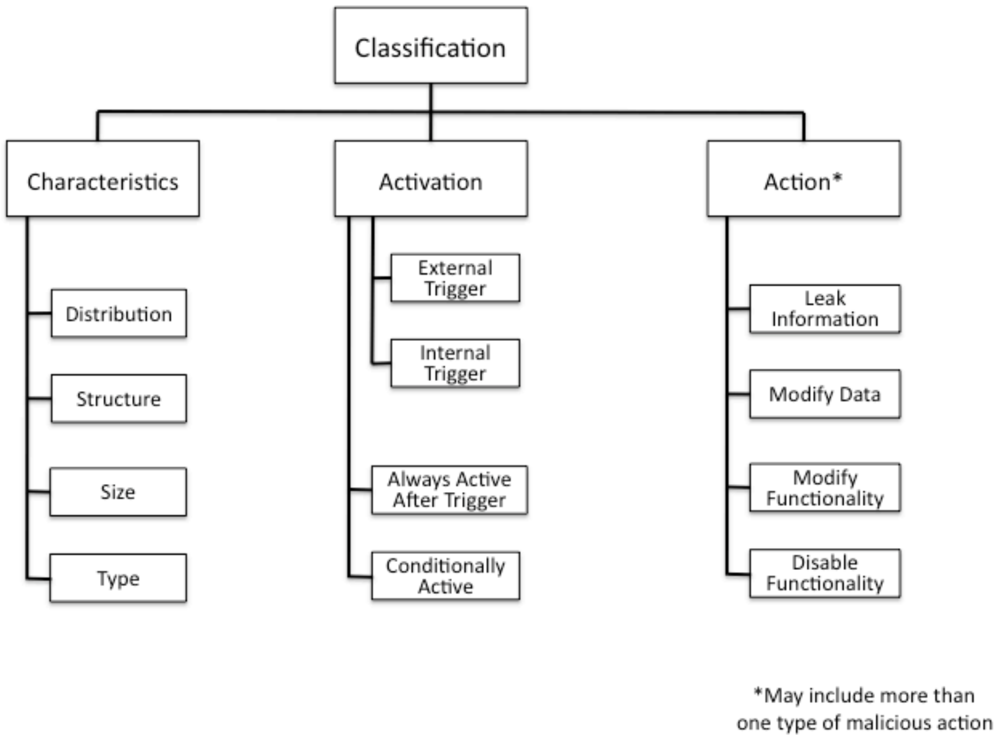

Figure 1: A taxonomy of Malicious Inclusions, modified slightly from (Tehranipoor 2010).

The components of a simple general-purpose processor are generally classifiable according to their function. For example, a circuit in a microprocessor may participate in control-flow execution (participate in fetch-decode-execute-retire), be part of a data path (like a bus), execute storage and retrieval (like a cache controller), assist with control, test and debug (as in a debug circuit), or perform arithmetic and logic computation (like an arithmetic-logic circuit, or ALU). This list may not be exhaustive, and some circuits' functions may overlap, but broadly speaking we can subdivide the component circuits in a processor using these classifications.

The main focus of our research is the detection of malicious inclusions which target the first category, control flow circuits. In considering processor malicious inclusions, it is worth noting that in some cases a *detection* strategy is warranted, and in others a *mitigation* strategy may be preferable. Table 1 lists each of the circuit functional types mentioned above, and pairs it with a potential 3D detection and/or mitigation strategy.

**Table 1:** Processor circuit type, with some associated MI mitigation and detection techniques.

| Circuit Type | Detection/Mitigation Technique |
|---|---|
| Control Flow | Control Flow Execution Monitor (subject of our experiments) |
| Chip Control, Test, and Debug | Keep-Alive Protections |
| Data Paths | Datapath Integrity Verification |
| Memory Storage and Retrieval | Load/Store Verification |
| Arithmetic and Logic Computation | Arithmetic/Logic Verification |

In Figure 2, we update the malicious inclusion taxonomy from Figure 1, and associate each MI action type with a matching detection or mitigation technique:

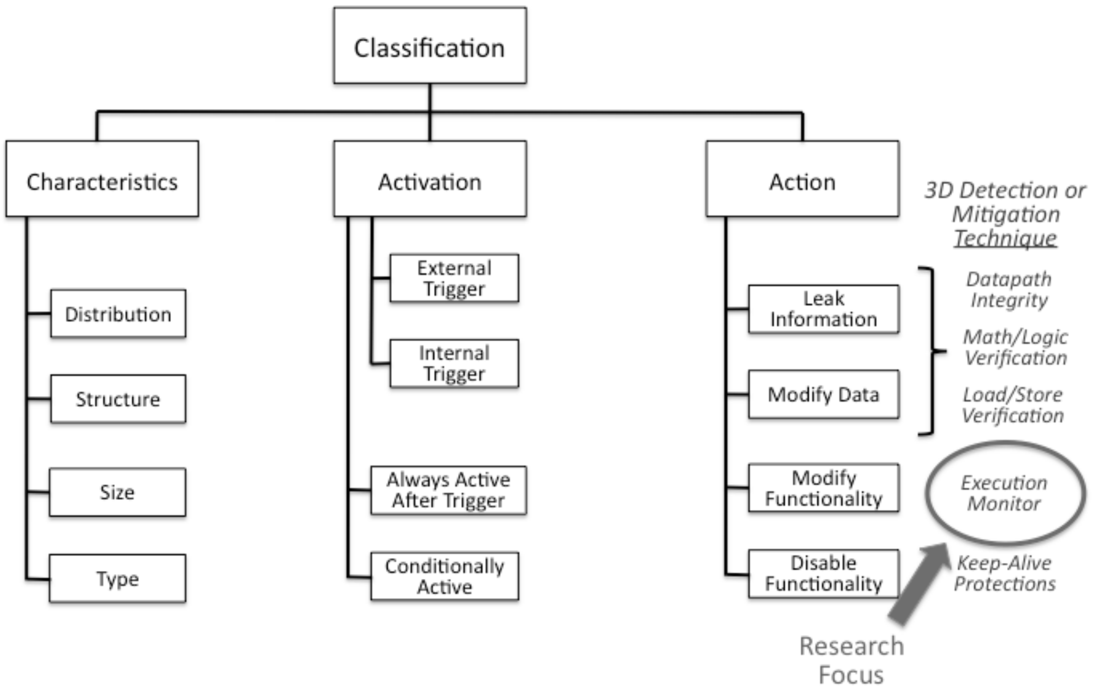

**Figure 2:** Malicious Inclusion taxonomy, with associated mitigation and detection methods.

In our current experiments, we intend to demonstrate an implementation of the execution monitor, which governs the operation of the instruction set of a general-purpose processor, and should detect MIs from the fourth action category, "Modify Functionality." MIs from this category might, for example, be designed to allow an adversary to leak secret information or to gain privileged access in system.

## 3. Limits of Existing Processor Tests

General-purpose processor designs go through verification testing before fabrication begins. Design-phase verification usually involves construction of a verification environment using tools like SystemVerilog and the Open Verification Methodology (OVM) (Iman 2008). There are several shortfalls with verifying processor designs, with respect to malicious inclusions:

- Not all processor designs, or portions of designs, undergo formal verification. Processor designs also may incorporate reused sub-components, as well as unverified open-source or third-party components.
- Processor design verification tends to ensure that the processor correctly executes its intended functions, but usually is not designed to verify the absence of additional, possibly malicious functionality, such as an MI.
- Processor design verification usually cannot be exhaustive, due to the exponential number of possible internal configurations of a processor. Modern functional verification often focuses on generating a sufficient number of random test cases to be reasonably confident of a design's correctness; as a result, rare-event malicious triggers may not be detected.

Once a processor has been fabricated, some sample dies may be examined, destructively or nondestructively, for the presence of MIs. Using destructive methods, a processor's top layers may be removed and its metal layers examined for anomalies, using specialized imagers. Since processors cannot be used operationally after destructive testing, it is limited to a small sample set, and not a complete solution.

Non-destructive processor tests include various power and timing "fingerprinting" techniques. Essentially, using sensitive measuring equipment, a tester can drive a processor's inputs with test patterns and measure current and timing delays at the outputs. The results from the device under test are statistically compared with the results from presumed-good, or "golden," sample processors. The principal limitations of nondestructive fingerprint-based testing include: (Agrawal 2007, Jin 2008, Jin 2009, Rad 2008)

- Such tests rely on the existence of a presumed-good "golden" sample. Therefore, if the subversion occurred in the design phase, and hence was cast into all the fabricated processors, the subversion will not be detected through these comparisons.
- Very small MIs, involving fewer than around .1% of the transistors on a die, are generally not detectable using these techniques, and it is not very difficult for an attacker to design a subversion which remains below this threshold.

## 4. 3D Fabrication and Potential Security Applications

Because feature sizes are shrinking very near to their theoretical limits, processor manufacturers are constrained in improving performance through the use of traditional methods on a single-layer design. As a result, manufacturers and designers have been rapidly advancing the technologies needed to make "3D" processors. In a 3D processor design, two or more silicon layers are joined together face to face or face to back, using a variety of interconnection methods. As a result, off-chip resources, like extra cache memory or another processor, which might normally be elsewhere on the printed circuit board, are physically much closer to the primary processing layer, resulting in shorter communication delays, and hence better performance (Mysore 2006).

Though the development of 3D interconnect technology has been driven by performance, several security-relevant applications have also been suggested (Valamehr 2010):

- 3D security services, such as those that might be found in a security coprocessor, could be made available to the primary processor layer.
- A 3D layer acting as a "control plane" could monitor and restrict the behavior of a target processor in the "computation plane." For example, the control plane processor could facilitate the segregation of multi-level data by partitioning the cache lines inside the target.

Another potential security-relevant application of 3D is the Execution monitor, or 3D-EM. With a 3D-EM, key control signals of the target processor, or computation plane, are monitored, through 3D interconnects, by another processor in the control plane. The EM's sole purpose is to monitor the execution of the target processor, and identify when the sequences of observed signal values deviate from those sequences allowed by the target processor's design.

Design and construction of a 3D-EM alongside a target processor could occur as follows:

- The target processor's architectural design is developed and translated into hardware design language (HDL).
- From the design documents and HDL specification, the processor's design undergoes normal functional verification (e.g., formal methods, OVM, simulation, FPGA test), to determine:
  - Correctness of the expected functionality (as normal).
  - Absence of any malicious additional functionality (additional steps for MI detection).
- Once the target's HDL design is finalized, the target's execution control signals (those which must be monitored) are identified. An HDL version of the monitor is constructed. One of our research goals is to develop a "recipe" for these two steps.
- During floorplanning (including power, area, and heat optimizations) of the target, the appropriate 3D monitoring interconnects are physically laid out, from the target layer to the monitor layer.

- The target's final floorplanned design is transferred to a set of fabrication masks and sent to the foundry for production. The target processors may be fabricated at either a trusted or an untrusted foundry.
  - Target processors which are *not* destined for high-assurance applications are finished and assembled onto printed circuit boards.
  - Target processors which *are* destined for monitored, high-assurance applications are shipped for further assembly.
  - The monitors are fabricated at a trusted facility.
  - The target processors and monitors are then joined, assembled onto printed circuit boards, and tested again.

Adding the extra steps to co-design a monitor will slow the overall development process; one goal of our research is to find ways to automate or semi-automate the monitor co-design portion.

The target processors could still be produced in large volume for non-high-assurance customers, where monitoring is not required, in order to keep their unit cost down. Only the high-assurance customers need to go through the extra steps of designing, fabricating, and joining the monitor layer.

The monitor layer might be placed above or below the target layer. One possible arrangement is shown in Figure 3:

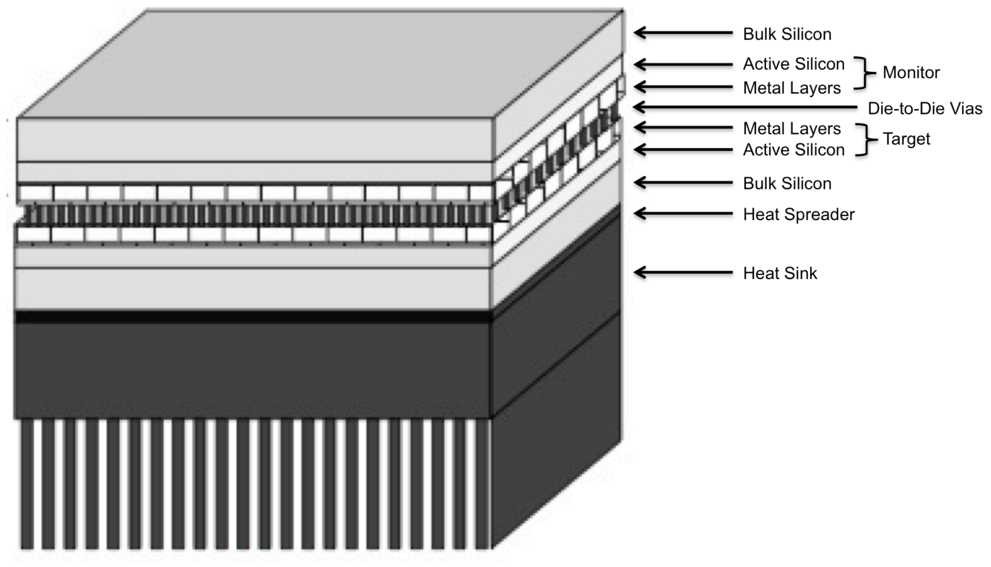

**Figure 3:** A possible 3D arrangement of the monitor and target layers, adapted from (Puttaswamy 2006).

## 5. Execution Monitor Theory

Several of the important characteristics of an EM were described by Schneider (Schneider 2000). A brief summary of some of the conclusions is listed below (see source for formal definitions of *safety property* and *security automata*).

- The target's execution is characterized by (finite or infinite) sequences, where $\Psi$ denotes a universe of all possible sequences, and a target *S* defines a subset $\Sigma_S$ of $\Psi$ corresponding to the executions of *S*. The sequences may be comprised of atomic actions, events, or system states, for example.
- A security policy is specified by giving a predicate on sets of executions. A target *S* satisfies security policy *P* if an only if $P(\Sigma_S)$ equals true.

- If the set of executions for a security policy $P$ is not a *safety property*, then an enforcement mechanism from an EM does not exist for $P$.
- EM-enforceable security policies are composable: when multiple EMs are used in tandem, the policy enforced by the aggregate is the conjunction of the policies enforced by each in isolation.
- A *security automata* can serve as the basis for an enforcement mechanism in EM.

Consider a set of signals $A$ which are dependent on the value of an instruction opcode in a processor. We assume that, within the set $A$, all the signals change values synchronously, as they would in a common clock domain. The possible values of a single member $a \in A$ may be described by a set of finite, discrete values $V$ (e.g., logic low, logic high, high impedance, etc.). These physical values are represented discretely in an HDL description, as well. For example, a VHDL "standard logic" signal is nine-valued: $V$ = {U, X, 0, 1, Z, W, L, H, -}. If set $A$ contains $n$ signals, we can denote them $a_1$, $a_2$, ... $a_n$. For a target processor $S$, containing the signals of $A$ (and others), the state of $A$ at time $t$ may be denoted $A_t$, and the *execution trace* of the signals in $A$ of processor $S$ may be described as an ordered set of states $\Sigma_S = \{A_0, A_1, \ldots\}$. Here, $\Psi$ represents the universe of all possible execution traces.

We hypothesize that, in terms of instruction set execution:

- The signals comprising $A$ may be systematically identified,
- The permitted and prohibited sequences of signal states, defining $P(\Sigma_S)$ = True and $P(\Sigma_S)$ = False, may be inferred from the processor's specification and HDL definition, and
- A 3D-EM developed using our construction meets the criteria of a *security automata*, enforcing a *safety property*.

One goal of our research is to demonstrate that a 3D processor Execution monitor can be developed which satisfies the conditions of (Schneider 2000) and is able to detect a certain class of MI - specifically, an MI which causes the processor's instruction-control signals, comprising the microarchitectural state of the machine, to deviate from their allowable control flow.

## 6. Experimental Evaluation

The ZPU is a simple general-purpose, open-source processor, whose VHDL design we obtained from OpenCores.org (OpenCores 2010). The ZPU uses 32-bit operands and a subset of the MIPS instruction set. It has a stack-based architecture, without an accumulator, and no internal processor registers. It is an unpipelined, single-core design, supporting interrupts, but with no privilege rings or other complex features. It is intended primarily for system-on-chip implementations in FPGAs.

The top level design of the ZPU (Figure 4) contains a processor core, a timer, a CPU-to-memory I/O unit, and a DRAM (memory) unit:

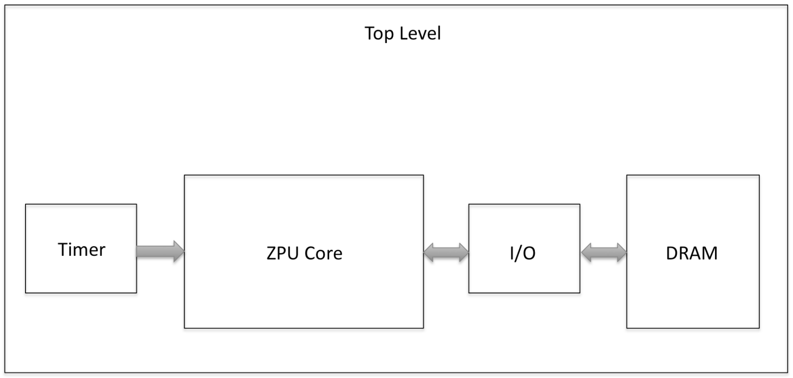

**Figure 4:** Processor and system configuration without Execution monitor.

We created and added a monitor entity for the processor core. The units communicated as below, in Figure 5:

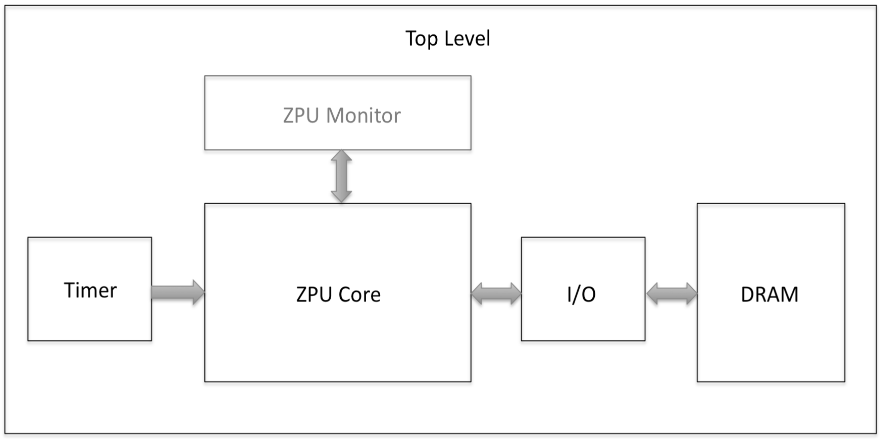

**Figure 5:** Processor and system configuration with Execution monitor added.

From the VHDL design of the ZPU core, we manually identified the control-type signals, i.e., the signals directly carrying out the instruction-set execution. Some examples of these include *memory_read_enable* and *memory_write_enable*, an *interrupt* signal, an *operand_immediate* signal, etc. The ZPU VHDL design explicitly characterizes the internal state of the processor with named states, from which we constructed a full finite state machine (of control signal states) and identified all the legal state-to-state transitions. Some of the ZPU's internal states and are shown in Figure 6.

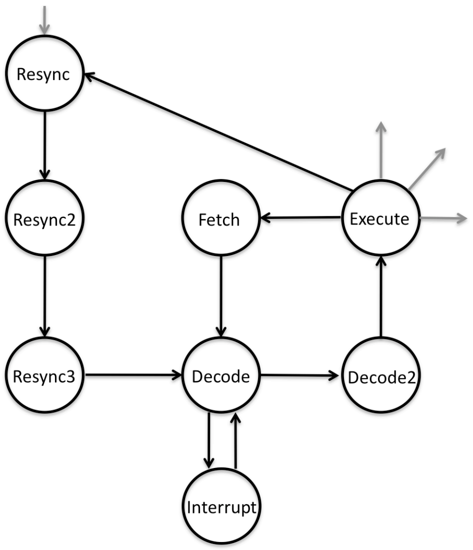

**Figure 6:** Some of the ZPU processor internal control states.

The ZPU monitor accesses the identified control signals through VHDL "ports". In a physical 3D design, these signals would transit from the target layer to the monitor layer by through-silicon vias (TSVs) or some other 3D joining method. This mapping might occur at the 3D floorplanning stage, before the netlist files have been synthesized into mask database files for each layer. Since this ZPU design was run in simulation but not physically synthesized, the physical 3D translation is notional. However, the circuit delay (one full clock cycle) for interlayer signal transmission and the number of 3D posts - approximately 50, in this case - are reasonable, given the current state of 3D interconnect design (Mysore 2006).

The monitoring logic actually makes two checks. The first check consults a lookup table that contains the state transition logic. For example, if the monitor detects that the ZPU went from state A to state B, and that the signal set was S at the completion of the clock cycle when it was in state A, the monitor looks to see if a matching legal transition exists in the table. The construction of the table is such that each transition must be unique; the processor can't choose nondeterministically among several available choices. If the monitor detects that no legal transition from state A with signal set S to state B existed, then it sets the output "predicate" to false to flag a violation.

The second check verifies that any changes to the signal set S, in state A, to the new signal set S', in state B, were legal, according to the transition table. Using the transition that was selected in the previous step, the monitor evaluates each signal in S' to see if it violated any of the post-conditions of the transition. If not, it again signals the appropriate predicate to false.

The monitor was evaluated using Mentor Graphics' Model Sim tool. In the first test, the unmodified ZPU processor executes code with the monitor observing. The ZPU software program used for these particular tests included a broad mix of all of the ZPU instruction set opcodes. In the first test, the execution of the unmodified ZPU did not cause the monitor to flag any transitions or signal modifications as illegal. Next, we made small modifications to the ZPU core, then recompiled the design and ran the simulation again.

Some of the small deviations we introduced in the ZPU processor design included:

- When visiting the internal "No-op" state, the ZPU increments a counter which ticks up to 5 "No-op" instructions, then on the next one sets the "inInterrupt" signal to 1, causing a violation to be observed by the monitor.
- In another modification, the ZPU tries to go straight from the internal "No-op" state to the "Resync" state (which is not allowed by the design specification), and again a violation is observed by the monitor.

The HDL code for these example deviations is below:

```
when State_Nop =>
        begin_inst <= '1';          -- normal instructions here
        idim_flag <= '0';
        pc <= pc + 1;
                                    -- deviations added below, to test the monitor
                HACKCOUNTER := HACKCOUNTER + 1;
                if (HACKCOUNTER > 5) then
                        -- state <= State_Resync;
                        -- inInterrupt <= '1';
                end if;
```

These test modifications of the ZPU were basic anomalies, not designed to accomplish any malicious action. More complex design modifications are necessary to implement truly malicious behavior, like leaking secret data.

## 7. Results

In Figure 7 below, the ZPU core is unmodified, and the processor only detects valid transitions. The two signals at the bottom, named "valid_transition" and "valid_changes", correspond to the two checks made above (failure of the first check automatically leads to failure of the second check). The signal "match_index_signal" is the entry number, from the transition table, of the valid transition observed; when none is found, it shows 0, and it also initializes to 0 at startup.

In Figure 8, the first anomaly has been introduced in the ZPU core. The vertical cursor indicates where the anomaly was detected, and the monitor's "valid" signals were both low. Similarly, in Figure 9, the second anomaly was detected.

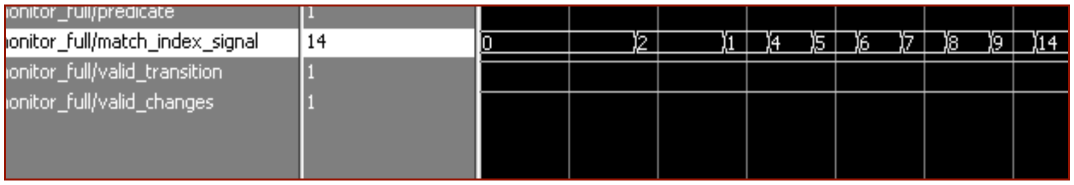
onitor_full/predicate
onitor_full/match_index_signal
onitor_full/valid_transition
onitor_full/valid_changes
1
14
1
1
0
2
1
4
5
6
7
8
9
14

**Figure 7:** The processor executed normally, and no anomalies were detected.

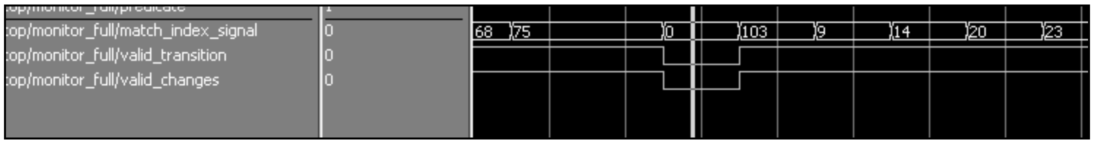
op/monitor_full/predicate
op/monitor_full/match_index_signal
op/monitor_full/valid_transition
op/monitor_full/valid_changes
1
0
0
0
68
75
0
103
9
14
20
23

**Figure 8:** The first processor anomaly was active, and was detected by the monitor.

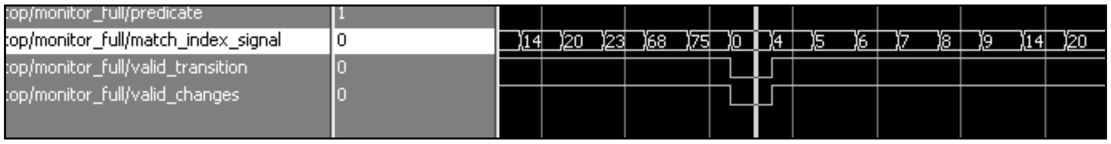
op/monitor_full/predicate
op/monitor_full/match_index_signal
op/monitor_full/valid_transition
op/monitor_full/valid_changes
1
0
0
0
14
20
23
68
75
0
4
5
6
7
8
9
14
20

**Figure 9:** The second processor anomaly was active, and was detected by the monitor.

The monitor's transition table had 112 records in it, to cover the 112 allowable transitions among the 23 unique internal processor states. These are reasonably small numbers to implement in a monitor, but we are also interested in the growth of the size of the monitor, as the target processor becomes more complex.

Recall from Section 5 that a standard circuit's voltage, as described in VHDL, can represent one of 9 discrete values. For *n* circuits, then, we would expect $9^n$ possible signal permutations - an impractically large number, if the state machine must have $9^n$ states, one for each permutation. We will explore in future research whether the actual number of required signal permutations, and hence monitor states, is typically much smaller, as was the case in this example.

We synthesized the design, using a Virtex-5 FPGA target, in two different configurations - the processor architecture alone, and the processor architecture with the monitor. In both cases, the maximum design speed was 228Mhz, indicating that adding the monitor did not impose a speed performance limit on the processor.

## 8. Conclusions

The following are some of the limitations of this research:

- The techniques illustrated are focused on only one of the categories of malicious inclusion from the taxonomy described earlier; detection and mitigation techniques should be developed for the other types as well, and this is an open research area.
- The Execution monitor's performance must not limit the performance of the target processor which it monitors. For example, the maximum clock speed of the EM should be at least as fast as the maximum intended clock speed of the target processor. The power, area, and heat requirements of the monitor should not exceed the practical limits of the overall 3D design. Also, the clock-cycle latency between MI activation and detection should be small enough to permit effective correction. We plan to evaluate 3D-EM designs further, using these performance measures, in the future.

From our preliminary work on the ZPU 3D-EM design, we reached the following conclusions:

- Designing and simulating the operation of a basic 3D monitor for a simple processor design is feasible. However, the physical design space for 3D monitors needs further exploration, and monitors for more complex processors should be developed.
- As expected, simple deviations from the processor's specified instruction-control behavior can be detected at runtime.
- The 3D Execution monitor is the first hardware-based approach with the potential for identifying processor MIs both during testing and during real-time, fielded operation - an important advantage over testbench methods, since delayed triggers may cause an MI to be inactive during predeployment testing.

## 9. Future Work

For this demonstration, we selected the control signals and developed the stateful representation manually. In future experiments, we hope to work on methods whereby the microarchitectural control signals can be automatically identified, and the monitor constructed automatically or semi-automatically (or identify any reasons why the process cannot be automated). We would like to design a monitor for a register-based processor with one or more data buses, in order to compare it with monitoring a stack-based processor like the ZPU. We would also like to design processor anomalies which accomplish some more meaningful subversions. Finally, we wish to test whether the monitor can detect unknown MIs, designed by third parties unfamiliar with the monitor construction.

It would be useful to scale up the 3D Execution monitor experiments to more complex processor designs, with modern features like pipelined and speculative execution, multithreading, vector operations, virtualization support, and multi-core.

## 10. Acknowledgements

This research was funded in part by National Science Foundation Grant CNS-0910734.

## 11. References

Adee, S., (2008) "The Hunt for the Kill Switch", [online] *IEEE Spectrum*, May 2008, http://spectrum.ieee.org/semiconductors/design/the-hunt-for-the-kill-switch

Agrawal, D., Baktir, S., Karakoyunlu, D., Rohatgi, P., and Sunar, B. (2007) "Trojan Detection Using IC Fingerprinting", *2007 IEEE Symposium on Security and Privacy.*

Defense Science Board (2005). *Report of the 2005 Defense Science Board Task Force on High Performance Microchip Supply*, Office of the Undersecretary of Defense for Acquisition, Technology, and Logistics.

Iman, S. (2008) *Step-by-Step Functional Verification with SystemVerilog and OVM*, Hansen Brown Publishing, San Francisco.

Jin, Y. and Makris, Y. (2008) "Hardware Trojan Detection Using Path Delay Fingerprint", *Proceedings of the 2008 IEEE International Workshop on Hardware-Oriented Security and Trust*.

Jin, Y., Kupp, N., and Makris, Y. (2009) "Experiences in Hardware Trojan Design and Implementation", *Proceedings of the IEEE International Workshop on Hardware-Oriented Security and Trust*.

King, S., Tucek, J., Cozzie, A., Grier, C. Jiang, W., and Zhou, Y. (2009) "Designing and Implementing Malicious Hardware", *Proceedings of the IEEE International Workshop on Hardware Oriented Security and Trust.*

Markoff, J. (2009) "Old Trick Threatens Newest Weapons", [online], New York Times, 27 October. http://www.nytimes.com/2009/10/27/science/27trojan.html?_r=2.

McCormack, Richard (2008) "DoD Broadens 'Trusted' Foundry Program to Include Microelectronics Supply Chain", *Manufacturing & Technology News*, Thursday, 28 February.

Mysore, S., Agrawal, B., Srivastava, N., Lin, S., Banerjee, K., and Sherwood, T. (2006) "Introspective 3D Chips", 2006 *International Conference on Architectural Support for Programming Languages and Operating Systems.*

Nystedt, D. (2007) "Intel Got its New China Fab for a Bargain, Analyst Says", [online] CIO.com, http://www.cio.com/article/101450/Intel_Got_Its_New_China_Fab_for_a_Bargain_Analyst_Says

OpenCores.org (2010), [online] http://opencores.org.

Pellerin, D., and Taylor, D. (1997) *VHDL Made Easy*, Prentice Hall, Upper Saddle River, NJ.

Puttaswany, K., and Loh, G., (2006) "Implementing Register Files for High-Performance Microprocessors in a Die-Stacked (3D) Technology", *Proceedings of the 2006 Emerging VLSI Technologies and Architectures*, Vol. 00, March.

Rad, R., Plusquellic, J., and Tehranipoor, M. (2008) "Sensitivity Analysis to Hardware Trojans Using Power Supply Transient Signals", *2008 IEEE International Workshop on Hardware Oriented Security and Trust.*

Schneider, F. (2000) "Enforceable Security Policies", *ACM Transactions on Information and System Security*, Vol. 3, No. 1, February, pp 30-50.

Tehranipoor, M. and Koushanfar, F. (2010) "A Survey of Hardware Trojan Taxonomy and Detection", *IEEE Design and Test of Computers*, vol. 27, issue 1, January/February, pp10-24.

Valamehr, J., Tiwari, M., Sherwood, T., Kastner, R., Huffmire, T., Irvine, C., and Levin, T., (2010) *Hardware Assistance for Trustworthy Systems through 3-D Integration*, Proceedings of the 2010 Annual Computer Security Applications Conference (ACSAC), Austin, TX, December.

Yinung, F. (2009) "Challenges to Foreign Investment in High-Tech Semiconductor Production in China", *United States International Trade Commission, Journal of International Commerce and Economics*, May.