Catching the Flu: Emerging Threats from a Third Party Power Management Unit

ABSTRACT
Power management units (PMU) have come into the spotlight with energy efficiency becoming a first order constraint in modern MPSoC designs. To cater to the exponential rise in power events due to the high device density, and to meet the demands of tight power and energy budgets, PMUs are evolving to more complex and intelligent designs. In an era defined by energy efficient computing, a malicious circuit embedded in a third party PMU can adversely affect the operation of the entire MPSoC.

This work presents and evaluates two covert security vulnerabilities, P-VIRUS and DROWSY, for the MPSoC systems, designed using a malicious third party PMU. Further, we propose a non-invasive IP risk assessment module to monitor the trustworthiness of the deployed PMU. Our techniques help flag malicious activities in the PMU with less than 1% area and power overheads.

1. INTRODUCTION
The phenomenal growth and integration of transistor devices has reshaped our notion of reliable and trustworthy hardware. Several emerging trends in hardware integration and user computing practices are exposing new challenges for secure hardware design. First, high performance Multiprocessor-System-on-Chips (MPSoCs) are emerging as an alternative to many general purpose and embedded processors [23]. Second, these MPSoCs promote unprecedented integration of Third Party Intellectual Property (3PIP) components, for reducing cost and design complexity. Third, MPSoC integrators are reluctant to re-verify the procured 3PIP, due to sky-rocketing verification costs and aggressive time-to-market schedules [3]. Fourth, the 3PIP vendors are averse to expose their internal design RTL to the MPSoC integrator to protect their competitive advantages, which creates a stiff barrier to many existing hardware trust assurance techniques relying on gate-level introspection [20,22]. In this ecosystem, an MPSoC integrator cannot ensure full trustworthiness of every 3PIP.

In this context, security assurance of the 3PIP power management unit (PMU) has profound implications for future MPSoC designs. To cater to the growing power and energy budgets, PMUs are evolving to more complex and intelligent designs. In an era defined by energy efficient computing, a malicious circuit embedded in a third party PMU can adversely affect the operation of the entire MPSoC.

This work presents and evaluates two covert security vulnerabilities, P-VIRUS and DROWSY, for the MPSoC systems, designed using a malicious third party PMU. Further, we propose a non-invasive IP risk assessment module to monitor the trustworthiness of the deployed PMU. Our techniques help flag malicious activities in the PMU with less than 1% area and power overheads.

1. INTRODUCTION
The phenomenal growth and integration of transistor devices has reshaped our notion of reliable and trustworthy hardware. Several emerging trends in hardware integration and user computing practices are exposing new challenges for secure hardware design. First, high performance Multiprocessor-System-on-Chips (MPSoCs) are emerging as an alternative to many general purpose and embedded processors [23]. Second, these MPSoCs promote unprecedented integration of Third Party Intellectual Property (3PIP) components, for reducing cost and design complexity. Third, MPSoC integrators are reluctant to re-verify the procured 3PIP, due to sky-rocketing verification costs and aggressive time-to-market schedules [3]. Fourth, the 3PIP vendors are averse to expose their internal design RTL to the MPSoC integrator to protect their competitive advantages, which creates a stiff barrier to many existing hardware trust assurance techniques relying on gate-level introspection [20,22]. In this ecosystem, an MPSoC integrator cannot ensure full trustworthiness of every 3PIP.

In this context, security assurance of the 3PIP power management unit (PMU) has profound implications for future MPSoC designs. To cater to the growing power and energy budgets, PMUs are evolving to more complex and intelligent designs. In an era defined by energy efficient computing, a malicious circuit embedded in a third party PMU can adversely affect the operation of the entire MPSoC.

This work presents and evaluates two covert security vulnerabilities, P-VIRUS and DROWSY, for the MPSoC systems, designed using a malicious third party PMU. Further, we propose a non-invasive IP risk assessment module to monitor the trustworthiness of the deployed PMU. Our techniques help flag malicious activities in the PMU with less than 1% area and power overheads.
3.1 Life Cycle of a 3PIP M-PMU Trojan

We outline the sequence of phases to realize a generic hardware trojan in the 3PIP PMU.

- **Trojan Insertion**: A few key engineers on the PMU 3PIP team can insert the malicious circuit during the design or test phase [22]. We have seen many cases of disgruntled employees involved in unprofessional practices motivated by vengeance or personal gain. The Volkswagen scandal exposed the presence of corporate sabotage to ward off competition [2]. Recently, a key player in the processor sector was accused of using its position to ward off competition [5]. In such scenarios, an unsuspecting IP vendor will end up supplying a trojan embedded IP to its clients.

- **Trojan Activation**: The attacker can employ a variety of subtle activation mechanisms that can either be triggered by the internal or the external environment [20]. Another class of trojan triggers involves the coalition of software-hardware, where an application running on the hardware can send a sequence of cryptic messages/requests to activate a dormant trojan in the 3PIP PMU [11].

- **Trojan Operation**: Once activated, the trojan can inflict a plethora of attacks on the on-chip components, adversely affecting the overall system behavior. In this work, we demonstrate two specific attacks that can cause a substantial degradation in energy efficiency and performance.

3.2 M-PMU Activation

The activation phase is particularly important to a trojan design. Careful choice of a rare event trigger can conceal the trojan during the pre-deployment testing of the chip. We envisage a software-hardware coalition based sequential trigger that takes advantage of the voltage change requests made by the on-chip components. Analogue to the idea presented by Waksman and Sethumadhavan (sequence cheat code) in their work [21], we use a sequence of power event requests such as specific voltage identification (VID) patterns to trigger the underlying trojan. An unsuspecting software can be designed, to send a particular sequence of voltage change requests in the form of VID signals to the PMU. The embedded trojan in the 3PIP PMU monitors the staggered incoming requests for the complex sequence of power events. In modern processors, VID signals are typically 6 to 8 bits and a sequence of these requests can potentially have an enormous state space resulting in astronomical test times. The complexity of trojan activation during testing due to the logic depth is investigated in Section 6.1.

3.3 P-VIRUS

Our first attack model, P-VIRUS, targets the energy efficiency and performance of the MPSoC. The embedded trojan in the PMU, inconspicuously manipulates the supply voltage request made by a processor for a set frequency, leading to improper voltage-frequency (VF) assignments.

3.3.1 P-VIRUS - Attack Environment

Figure 1a, illustrates the environment of a P-VIRUS attack. The processor requests the PMU for a specific voltage level based on the decision made by a dynamic voltage frequency scaling (DVFS) control algorithm. For example, the Intel processors use a Serial Voltage IDentification (SVID) interface to communicate the VID requests to the PMU, which in turn decodes the VID and supplies the corresponding voltage to the processor through the power rails. Specifically, in Intel’s i7-4650 processor line, the 8-bit VID signal (00h-FFh) can request for a voltage ranging between 0 to 3.04V, though the specified processor operating voltage range is between 1.64V to 1.85V [10].

<table>
<thead>
<tr>
<th>Req. $V_{CC}$</th>
<th>VID$_{TF}$</th>
<th>VID$_{FEV}$</th>
<th>$V_{CCP}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.65V</td>
<td>$V_0$</td>
<td>$7C$ - $AC$</td>
<td>1.73V - 2.02V</td>
</tr>
<tr>
<td>1.8V</td>
<td>$V_1$</td>
<td>$8B$ - $BB$</td>
<td>1.88V - 2.02V</td>
</tr>
</tbody>
</table>

Table 1: VID manipulation example. Req. $V_{CC}$ is the voltage requested by the core by sending VID$_{TF}$ (Hex). The LFSR generates random values between 001 to 111 and manipulates the VID$_{TF}$ to a value in the range of VID$_{FEV}$ (Hex). $V_{CC}$ is the range of infected voltage supplied to the core.

3.3.2 P-VIRUS - Attack Variants

We envisage three variants in a P-VIRUS attack, based on the manipulation of the voltage request and the resulting behavior. The three variants are discussed next.

- **FRACTURE ($V_{supply} < V_{request}$)**: The voltage supplied by the PMU is less than the requested voltage. The processor circuit cannot meet the timing constraints for the particular frequency due to the higher delay caused by the lower supply voltage, leading to functional errors.

- **FEVER ($V_{supply} > V_{request}$)**: The supply voltage is higher than the requested voltage. The higher supply voltage will result in a loss of energy efficiency, and cause a rise in the on-chip temperature, similar to the onset of a fever. Such a subtle manipulation of the supply voltage can also keep the attack stealthy. To mitigate the elevated temperature, the system may lower the operating frequency, thereby degrading the processor performance.

- **STROKE ($V_{supply} >> V_{request}$)**: The supplied voltage is substantially higher than the requested voltage. The excess heat dissipation due to a sudden increase in the temperature can lead to a catastrophic failure and chip burnout, mimicking a stroke attack.

To demonstrate that even a modest voltage manipulation can significantly affect the system behavior, in this paper we model the FEVER variant of P-VIRUS.
In our second attack, DROWSY, the M-PMU tampers with the sleep and wake up requests of the on-chip components. The attack mimics the effects of drowsiness, affecting the availability of on-chip resources.

3.4 DROWSY

In our second attack, DROWSY, the M-PMU tampers with the sleep and wake up requests of the on-chip components. The attack mimics the effects of drowsiness, affecting the availability of on-chip resources.

3.4.1 DROWSY - Attack Environment

Figure 2a illustrates the operation of the DROWSY attack. The VID decoder block decipher the incoming VID signal and the trojan manipulates the deciphered VID signal. To keep the attack stealthy, the degree of attack is constrained by altering only 3 bits (4, 5 & 6) of the VID signal in a random fashion using a linear feedback shift register (LFSR). The infected VID (VID_{F,EV}) request is forwarded to the control circuit. The control block manages requests from various on-chip components and instructs the regulator to scale the voltage to the required level. The regulator block outputs the scaled voltage to the power rail corresponding to the block that requested the voltage change. The P-VIRUS infected supply voltage value (V_{CC}) corresponding to the VID_{F,EV} is constrained within the Max. Core V_{CC}\textsuperscript{1}. The limitations and randomness in the degree of voltage manipulation obscures the attack, while ensuring that the manipulation is significant enough to cause a considerable degradation in system behavior.

Table 1 gives an example of the VID manipulation for an incoming request of 1.65V. A 3-bit LFSR generates a random number between 001 to 111, to inflict an inconsistent attack and circumvent the generation of identifiable patterns in system behavior. On adding it to VID_{F,EV}, we get a range of possible VID_{F,EV} (Column 3). Note that the V_{CC} is limited to a maximum of 2.02V (Max. Core V_{CC}) though the VID_{F,EV} corresponds to a voltage range of 1.75V to 2.21V.

3.4.3 P-VIRUS - Implementation

Figure 1b shows the detailed implementation of the P-VIRUS attack. The VID decoder block decipher the incoming VID signal and the trojan manipulates the deciphered VID signal. To keep the attack stealthy, the degree of attack is constrained by altering only 3 bits (4, 5 & 6) of the VID signal in a random fashion using a linear feedback shift register (LFSR). The infected VID (VID_{F,EV}) request is forwarded to the control circuit. The control block manages requests from various on-chip components and instructs the regulator to scale the voltage to the required level. The regulator block outputs the scaled voltage to the power rail corresponding to the block that requested the voltage change. The P-VIRUS infected supply voltage value (V_{CC}) corresponding to the VID_{F,EV} is constrained within the Max. Core V_{CC}\textsuperscript{1}. The limitations and randomness in the degree of voltage manipulation obscures the attack, while ensuring that the manipulation is significant enough to cause a considerable degradation in system behavior.

Table 1 gives an example of the VID manipulation for an incoming request of 1.65V. A 3-bit LFSR generates a random number between 001 to 111, to inflict an inconsistent attack and circumvent the generation of identifiable patterns in system behavior. On adding it to VID_{F,EV}, we get a range of possible VID_{F,EV} (Column 3). Note that the V_{CC} is limited to a maximum of 2.02V (Max. Core V_{CC}) though the VID_{F,EV} corresponds to a voltage range of 1.75V to 2.21V.

3.4 DROWSY

In our second attack, DROWSY, the M-PMU tampers with the sleep and wake up requests of the on-chip components. The attack mimics the effects of drowsiness, affecting the availability of on-chip resources.

3.4.1 DROWSY - Attack Environment

Figure 2a illustrates the operation of the DROWSY attack. When the on-chip resources are idle, they are put to sleep to save power. The control circuit manages the requests coming from the PMIP or the OS and puts the corresponding block to the sleep state. When the OS requests for the resource to wake-up and resume operation, the PMU delays the response to this request, adversely affecting the application performance. During an epoch, one or many blocks may send the wake-up request to the control circuit in the PMU. A block select module randomly chooses a subset of blocks to inflict a sluggish response. The response to these blocks are delayed by a random, but bounded time, using the delay block fed by the LFSR. The randomness in attack, the constraint in delay threshold, and the incoherent focus on a subset of blocks, obscure the DROWSY attack.

3.5 Evaluation Methodology

We evaluate the effects of the trojan using Sniper6.0 [6]. We model the Intel i7-4650U processor (Table 2) and inject the trojan in it. To model the effect of P-VIRUS, we tamper with the operating voltages of processor performance-states (p-states), while running the Splash2 benchmarks. A widely-used on-demand DVFS algorithm is used for p-state assignment [16]. A peak thermal design power (TDP) is set. The frequency is throttled down when the dynamic power (thermal load) rises beyond the set TDP, to shun assignments that can lead to a chip burn-out or system failure.

To model the effect of DROWSY, we repurpose the cost associated with sleep periods and add an extra latency whenever a block resumes from a sleep state.
Performance degrades due to the thermal throttling to ensure chip safety.

Figure 3: Effect of the P-VIRUS trojan on the peak power, energy and performance of applications. Peak power increases as a result of increased operating voltage. Energy increases due to the increased power and degradation in application performance. Performance degrades due to the thermal throttling to ensure chip safety.

3.6 Potency and Design Footprint

We evaluate the potential of the threat in terms of increase in energy consumption, peak power and degradation in performance (Section 3.6.1). We also present the design footprint to realize these trojans in the PMU (Section 3.6.2).

3.6.1 Potency

**P-VIRUS**: Figure 3a shows the comparison of peak power between the P-VIRUS free and P-VIRUS infected executions. On an average, the peak power increases by 19% with ocean.cont incurring the highest increase in peak (23%). The rise in peak power is limited by the thermal design power (TDP), beyond which the operating frequency of the processor is throttled down to prevent a thermal failure.

Figure 3b shows the impact of P-VIRUS on the energy consumed. On an average, the energy consumed increases by 18% with ftb incurring the highest increase in energy consumption (26%). We observe an anomaly for the ocean.cont benchmark, which incurs a meager 5% increase in energy. Figure 3c, reveals that ocean.cont does not incur any performance degradation. For this benchmark, the processor is able to operate at a higher voltage without breaching the set TDP. The loss in energy efficiency is purely due to the increased power. For the other benchmarks, Figure 3c shows the degradation in processor performance due to thermal throttling (up to 26%). Overall, the trojan degrades the system behavior in all the three measured metrics.

**DROWSY**: Figure 4 shows the performance degradation due to DROWSY. The sluggish responses to wake-up requests increase the number of idle cycles, resulting in a maximum performance degradation of 59% (average across benchmarks = 34%). The blocks in sleep or those under transition cannot be accessed, thereby preventing any functional errors due to DROWSY. The energy overhead is negligible, as the blocks consume minimal power during sleep.

3.6.2 Design Footprint

To evaluate the footprint of P-VIRUS and DROWSY, we augment the PMU RTL of the OpenSPARC T2 processor [14]. We implement the logic discussed in Sections 3.3 and 3.4, and synthesize with the TSMC 45nm library using the Synopsys Design Compiler. P-VIRUS incurs area and power overheads of 1.39% and 1.06%, respectively. DROWSY incurs an overhead of 1.84% in area and 1.92% in power.

4. M-PMU DETECTION

In this section, we explore a novel Intellectual Property Risk Assessment Module (IPRAM), designed by the MP-SoC integrator and placed at the interface of critical 3PIP blocks. The IPRAM assumes no support from the 3PIP PMU vendor and effectively detects malicious activities in PMUs, across various vendors. The IPRAM consists of two low complexity blocks: (a) P-VIRUS Monitor (Section 4.2) and (b) Wakeup Alarm (Section 4.3).

4.1 IPRAM OVERVIEW

Figure 5a shows the block diagram of the proposed IPRAM. The outgoing power management requests from the 3PIP blocks are in parallel sent to the IPRAM and the corresponding response from the PMU is forwarded to it too. Since the IPRAM is placed in parallel to the communication between a 3PIP core and the PMU, it does not add to the latency of the power management state changes. The novel blocks in the IPRAM match the requests to responses and identify the presence of a malicious activity in the PMU’s response.

4.2 P-VIRUS Monitor

P-VIRUS Monitor (PM) is based on the insight that the supply voltage ($V_{cc}$) of a digital circuit profoundly influences its delay. By characterizing the delay of a known circuit, we can estimate the imposed supply voltage on the block.

Figure 5a presents the interface of PM, modeled as a delay estimation block (DEB), fed by the same supply line as that of the 3PIP block. Figure 5b shows a detailed operation of the DEB. The control unit power gates the DEB when not in use, to curtail the power consumption overhead. A series of cascaded delay buffers (DB) in the form of a tunable replica circuit are used to estimate the parasitic delay of the circuit [7]. The cascaded buffers are sampled at equal epochs to capture the state transition at different stages and thereby
estimate the delay for the imposed V_{cc}. The estimated delay is correlated to the values stored in the look up table to obtain theVID corresponding to the delay. The look up table is filled, based on the post-silicon time characterization. The identified VID from the look up table is then matched to the VID request in the comparator unit, by ignoring the 2 least significant bits to grant a margin for error. If the values do not match, the DEB flags an anomalous behavior.

4.3 Wake Up Alarm

Wake Up Alarm (WA) is a finite state machine (FSM) to observe the arrival of requests and responses of the sleep state transitions. Delayed and unprompted state changes triggered by the malicious PMU can be detected.

WA is modeled as a response audit block (RAB) in Figure 5a, where the wake up request sent to the PMU is also sent in parallel to the RAB. Figure 5c shows a detailed diagram of the RAB. MPSoC blocks usually have multiple sleep states, varying in state transition times and power saving capabilities. The control circuit in the RAB feeds a multiplexer to select the appropriate delay associated with the sleep-to-wake transition. This delay is imposed on the request and forwarded to the capture circuit, that consists of a FSM with 3 states, default (D), wait (W) and anomaly detected (T). The FSM is in the D state until it detects a signal from the 3PIP. The state transitions are discussed below.

- \( D \rightarrow W \): On receiving a wake-up request from the 3PIP, the state changes from D to W.
- \( W \rightarrow D \): If the PMU response is observed within the vendor specified response time, it triggers a transition from W to D. The 3PIP resumes its operation.
- \( W \rightarrow T \): If the PMU response is delayed, the delay block triggers a transition from W to T, flagging the anomalous behavior—a delayed wake up attack.
- \( T \rightarrow D \): Once the attack is flagged, the signals and monitor blocks are reset, and the state transitions to the default (D) state to wait for the next request.
- \( D \rightarrow T \): An unsolicited shutdown/wake-up signal from the PMU, is captured and flagged—an abrupt attack.

5. METHODOLOGY

In this section, we discuss the methodology used to evaluate the efficacy and overhead of the IPRAM.

PM: To evaluate the efficacy of PM, we model the cascaded buffers present in the DEB, in HSPICE and obtain the delay characteristics for different V_{cc} values. To account for process variation, we include a Monte Carlo analysis by creating a Gaussian distribution of three parameters: threshold voltage, effective channel length and transistor width [17].

6. EXPERIMENTAL RESULTS

In this section, we evaluate the complexity of trojan activation during the pre-deployment test (Section 6.1). We then present the results for PM and WA based on two metrics: efficacy (Section 6.2) and design overhead (Section 6.3).

6.1 Limitations of Post Silicon Testing

The necessity for the IPRAM arises from the limitations of post-silicon testing, as demonstrated in Table 3. Our evaluation reveals the magnitude of complexity involved in guaranteed trojan activation during testing [8]. A modern PMU can be tested by applying test patterns (VID), and observing the resulting voltage levels. However, unlike a logic circuit, one must wait for voltage stabilization before running subsequent tests. Typical voltage level stabilization times lie between 0.1 to a few milliseconds [1]. With these considerations, we present the data for an 8-bit VID and sequence lengths (SL) of 8, 16 and 64 signals. For the SL of 8, we see a meager area (0.38%) and power (0.17%) overhead, while the state space exploration time is enormous. The data shows the extremely low probability of activating the M-PMU during the post-silicon tests, thereby emphasizing the need for the proposed security assurance techniques.

6.2 Solution Efficacy

PM: Figure 6 illustrates two representative cases of PM. In Figure 6a, we see the worst case scenario for detection, where the difference between V_{CCmin} (1.73V) and V_{CCmax} (1.65V) is a mere 0.08V. This is the lowest increment in voltage that the FEVER can impose, based on the trojan design (Section 3.3). The overlap in the distribution implies that these delay values can occur both in the presence and absence of a P-VIRUS attack leading to an inaccuracy in trojan detection (false detection). Figure 6c represents a typical case, where P-VIRUS increases the voltage by 0.1V. For attacks with a voltage increment greater than 0.1V, the distribution

<table>
<thead>
<tr>
<th>SL</th>
<th>Area</th>
<th>Power</th>
<th>TP</th>
<th>Time (years)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0.38%</td>
<td>0.17%</td>
<td>5.42 \times 10^{-4}</td>
<td>2.92 \times 10^{-8}</td>
</tr>
<tr>
<td>16</td>
<td>0.57%</td>
<td>0.30%</td>
<td>2.84 \times 10^{-3}</td>
<td>5.40 \times 10^{-7}</td>
</tr>
<tr>
<td>64</td>
<td>3.40%</td>
<td>2.10%</td>
<td>7.46 \times 10^{-3}</td>
<td>2.13 \times 10^{-7}</td>
</tr>
</tbody>
</table>

Table 3: Area and power overhead, and state space exploration time for the proposed trigger. TP represents trigger probability.

WA: Since WA identifies DROWSY via a FSM, based on the incoming signals, we evaluate WA by implementing a Verilog RTL model. We use Xilinx ISE to perform exhaustive tests and observe the behavior for different scenarios.

We implement the blocks of IPRAM in Verilog RTL and synthesize the RTL with the TSMC 45nm library using Synopsys Design Compiler to find the design overheads.
respectively. The power overheads are negligible, as block s
behavior, we propose a low complexity IPRAM. Our mod-
ification Overhead

PM and WA incurs area overheads of 0.97% and 0.14%, respectively. The power overheads are negligible, as blocks in the IPRAM are power gated when not in use.

7. CONCLUSION

This work outlines a novel security threat stemming from a malicious third party PMU that can adversely impact the MPSoC’s system behavior. To detect anomalous M-PMU behavior, we propose a low complexity IPRAM. Our mod-
ules incur marginal overheads and have a FDR of 5%, while detecting anomalous behaviors inflicted by the M-PMU.

8. REFERENCES