Latency Reduction in a System with IOPMP

Paul Shan-Chyun Ku and Warren Pen-Chih Chen

Andes Technology Corporation

Abstract

When people talk about platform security, memory isolation is considered a key fundamental. In a RISC-V-based platform, there are a couple of mechanisms inside a hart to perform physical memory isolation, such as PMP, ePMP, sPMP, etc. They are used to control the access from the CPU itself. Other than CPU, I/O agents' accesses are controlled by IOPMP. IOPMP is a checker with a set of ordered rules. Checking an access can be time-consuming because the check may not finish in one cycle, and sometimes more than one access is needed for one check. This creates a problem for latency-sensitive systems. This document will introduce two features to mitigate this problem, which were discussed in the RISC-V IOPMP Task Group. We will first introduce the latency reduction in respect of the downstream devices of an IOPMP, and then we will discuss the cooperation between IOPMP and its upstream prefetcher.

Introduction

When people talk about platform security, memory isolation is considered a key fundamental. In a RISC-V-based platform, there are a couple of mechanisms inside a hart to perform physical memory isolation, such as PMP, ePMP, sPMP, etc. They are used to control the access from the CPU itself. Other than CPU, I/O agents' accesses are controlled by IOPMP. IOPMP is a checker with a set of ordered rules. Checking an access can be time-consuming because the check may not finish in one cycle, and sometimes more than one access is needed for one check. It creates a problem in latency-sensitive systems. This document will introduce two features to mitigate this problem, which were discussed in the RISC-V IOPMP Task Group. We will first introduce the latency reduction in respect of the downstream devices of an IOPMP, and then we will discuss the cooperation between IOPMP and its upstream prefetcher.

An IOPMP is located in the bus fabric and checks every access going through it. Thus, an IOPMP has at least one input port and at least one output port. An IOPMP receives accesses from an upstream device through an input port, checks these accesses, and let them go to its downstream device from an output port. An IOPMP could be a standalone component in the bus fabric and integrated into other components, such as the bus bridge or the controller of the DDR SDR, Double Data Rate Synchronized Dynamic RAM.

To apply IOPMP, an access should carry a source ID, SID for short, as the identification of security. SID can be shared between multiple I/O agents as long as they have exactly the same permission on all downstream devices. When an access reaches IOPMP, its SID will be used to look up a set of memory domains. Every memory domain is consisted of a set of IOPMP rules. A memory domain can belong to multiple SIDs, but IOPMP rules can belong to at most one memory domain. The concept of a memory domain is a set of regions with their permissions for a specific function, such as a Network Interface Controller, NIC. One can group all its buffers and MMIO registers into a memory domain, and switch the NIC operator from one SID to another SID efficiently. Besides, sharing a set of memory regions between multiple SIDs is another use case.

In IOPMP's architecture spec, it doesn't define the upper bound of the number of I/O agents so it can support thousands of rules resulting that each check may take multiple cycles. Besides, multiple accesses may come simultaneously, so the time spent on IOPMP may be extended. Thus, the first latency reduction in this document tries to mitigate this type of latency.

Besides, an access initiator may utilize the prefetch mechanism to guess the memory address about to access. The prefetcher may read a memory by a guessed address. However, the guessed address could unintentionally fall into an illegal region and then trigger the process to handle a violation. It typically invokes the security software that should examine if it is a prefetch violation. Such a false alarm significantly degrades the performance, so our second subject is to smooth the cooperation between IOPMP and its upstream prefetcher.

Speculation to Downstream

As mentioned previously, an access may have to take a while to check its legality in IOPMP, which could cause a long latency and/or degrade overall performance. Access violation happens rarely; however, the check latency is always there. For a read access, if we can speculatively read back the data in advance and not return the real data on violation, we can reduce the amotized latency. If all

downstream devices are idempotent, IOPMP could read the data back without leaving any side effects before the check completes. If the access is found legal later, the data can be returned to its initiator. Otherwise, the data will be discarded and kick off the normal violation process. The violation is rare in a typical system, so such a speculation can overlap the time spent on checking access and reading data. That is, we could hide a large moment latency caused by IOPMP's checking.

Write latency could sometimes impact performance. DDR would need to switch the bank and turn on the row before accessing data. If IOPMP could issue the write command to the downstream DDR controller before the access check completes, the controller could prepare the bank and the row in advance. Once the access is identified as illegal, we can either issue an abort to DDR for the following actions or send the data with all byte lanes of zero.

IOPMP spec would include speculation related controls mentioned above, which are implementation-dependent. These controls are only used to hint the IOPMP that it can enable speculation. IOPMP can ignore them partially or totally.

Prefetch Access

Prefetch is a widely used technic to reduce read latency by guessing the next or next few addresses and reading them back in advance. However, a guessed address could reach an illegal region, which is caught by IOPMP. Such an illegal access is considered a false alarm because it is the prefetcher's speculation causing the violation, not an actual security attack. Asking the security software to examine such a false alarm will increase unnecessary burdens. Thus, IOPMP can adopt a different response to a violation caused by a prefetch access, rather than the usual response to a normal access violation. IOPMP could respond to the prefetcher, which issues the violation, with a programmable bus error. Then, the prefetcher will notice its mistake and stop the following speculation, at least in this speculation stream. All these happen without software's intervention, and IOPMP works with prefetchers more smoothly.

Discussion

When we mentioned the speculative accesses to downstream, we presume there should be switches to enable this function. The switch could be IOPMP-wide which means all transactions going through the IOPMP can speculative or none can be speculative. Other possibilities could be per memory domain, per entry, per SID, by QoS, and so on. The design of per memory domain or per entry may not be practical because the time to retrieve the memory domain index or entry index is supposed to be in a very late stage of checking. Per SID or by QoS seems more feasible. Anyways, whenever an IOPMP implementation allows some speculative transactions and some waiting for check completion, the ordering rule should also be considered on the bus implementing the thread identifier (on OCP), the transaction ID (on AXI), or something similar.

In the configuration registers for normal violations, we have a record for a normal violation. However, we don't plan to have a record of the prefetch violation since we don't think it is a typical case to retrieve information about a prefetch violation. If needed, it is considered a part of a bus tracer or debugger.