iPIC: A Hardware Mechanism for Faster Interrupt Handling on Embedded Virtualizations

Tomoaki Ukezono
School of Information Science
Japan Advanced Institute of Science and Technology (JAIST)
1-1, Asahidai, Nomi, Ishikawa, Japan
Email: t-ukezo@jaist.ac.jp

***Abstract*—Our final goal is to implement entirety of the embedded virtualizations as hardware. The hardware VMM (called H-VMM in this paper) can eliminate software overheads of virtualizations. Therefore, the H-VMM can exploit virtualizations to embedded systems which are nervous about real-time execution such as automotive systems, control of airplanes and medical devices.This paper especially focuses on overheads which are caused by interrupt handling, and propose novel hardware mechanism to eliminate the overheads.**

In our first step, we are focusing on I/O handling by virtualizations. Figure1 shows system model of the VMM controls and proposed method. The figure shows three types of implementation for I/O handling. The Virtualized I/O (left side of the figure) is typically used for general-purpose (desktop or laptop computer) system and can handle I/O devices which are shared between VMs. This implementation is the most flexible. However, it has unrealistic overheads for embedded system due to the device emulation. The Direct I/O (center of the figure) is widely used for embedded system or HPC system. Instead of device emulation, VMs must control individual I/O devices by physical driver directly. The True Direct I/O (right side of the figure) is our proposed architecture. In the True Direct I/O, two dedicated hardware with action comparable to VMM processing by the Direct I/O are implemented. Therefore, there is no interference by software whatsoever when VMs control I/O devices. The I/O handling by VMMs consists of two types of processing. One is I/O port handling. I/O port handling is required when the I/O port map of new platform is different from old platform. Our previous work[1] showed I/O performance improvement by 10% to 14% by implementing only address translation of I/O port map. The other is PIC Control. VMMs must know which interrupt from devices is supported by which VM and configure PIC (Programmable Interrupt Controller) appropriately. Naturally, the PIC control has software overhead. However, our previous work has no mention of the PIC control. In this paper, we focus on the PIC control and propose the intelligent PIC (named iPIC) which can act as a substitute for PIC control by VMMs.

Common PICs can configure only mask of interrupt. Therefore, VMMs must check current running VM and call appropriate interrupt handler on each interrupt, since multiple VMs are running on the VMM. Figure2 shows our proposed iPIC hardware. The iPIC conform hardware composition to OpenRISC 1000 architecture[2], since implantation of the

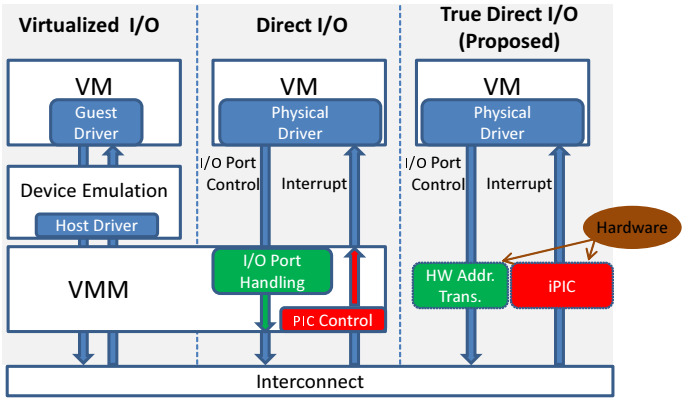

Fig. 1. The True Direct I/O Architecture.

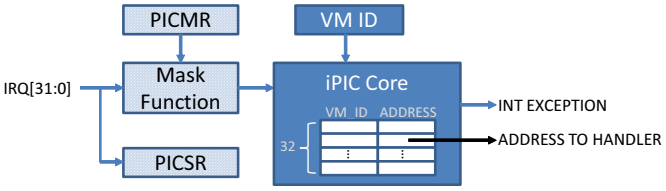

Fig. 2. Mechanism of iPIC.

iPIC depends strongly on individual micro-architecture. The PICMR (PIC Mask Register), Mask Function and PICSR (PIC Status Register) are already defined by OpenRISC 1000. The VM ID and iPIC Core are additional functions by H-VMM. VM ID represents identifier of current running VM. The iPIC Core has an address table. The table is indexed by IRQ (Interrupt ReQuest) number. When IRQ is received from Mask Function, iPIC Core compare VM ID register and VM ID field of the address table. If the VM ID filed is matched, corresponding ADDRESS field is loaded and set to the program counter. At the same time, signal of external interrupt (INT EXCEPTION) is supplied to integer unit. Otherwise, the iPIC does not assert INT EXCEPTION. In other words, the iPIC behave like additional Mask Function for virtualizations. By introducing this simple hardware, overheads on each interrupt can be eliminated.

REFERENCES

[1] T. Ukezono and K. Araki, "Performance Evaluation for Hardware Translation of I/O Address Map in Embedded Virtualization", Proc. of IPSJ Embedded System Symposium 2013, pp.131-139, 2013. (Japanese)

[2] "OpenRISC Architecture Manual", OPEN-ORES.ORG, in http://www.da.isy.liu.se/courses/tsea44/OpenRISC/openrisc arch3.pdf, 2003.