SgInt: Safeguarding Interrupts for Hardware-Based I/O Virtualization for Mixed-Criticality Embedded Real-Time Systems Using Non Transparent Bridges

Daniel Münch[1(✉)], Michael Paulitsch[1], Oliver Hanka[1], and Andreas Herkersdorf[2]

[1] Airbus Group Innovation, Munich, Germany
{Daniel.Muench,Michael.Paulitsch,Oliver.Hanka}@airbus.com
[2] Institute for Integrated Systems, TU München, Munich, Germany
herkersdorf@tum.de

Abstract. Safety critical systems and in particular higher functional integrated systems like mixed-criticality systems in avionics require a safeguarding that functionalities cannot interfere with each other. A notably underestimated issue are I/O devices and their (message-signaled) interrupts. Message-signaled interrupts are the omnipresent type of interrupts in modern serial high-speed I/O subsystems. These interrupts can be considered as small DMA write packets. If there is no safeguarding for interrupts, an I/O device associated with a distinct functionality can trigger any interrupt or manipulate any control register like triggering reset of all processing cores to provoke a complete system failure. This is a particular issue for available embedded processor architectures, since they do not provide adequate means for interrupt separation like an IOMMU with a granularity sufficient for interrupts.

This paper presents the *SgInt concept* to enable the safeguarding of interrupts for hardware-based I/O virtualization for safety-critical and mixed-criticality embedded real-time systems using non-transparent bridges in single (multi-core) processor systems and multi (multi-core) processor systems. The advantage of this SgInt concept is that it is an general and reusable interrupt separation solution which is scalable from a single (multi-core) processor to a multi (multi-core) processor system and builds on available COTS chip solutions. It allows to upgrade spatial separation for interrupts to available processors having no means for interrupt separation. A practical evaluation shows that the SgInt concept provides the required spatial separation and even slightly outperforms state-of-the-art doorbell interrupt handling in transfer time and transfer rate (by about 0.04 %).

1 Introduction

Driven by the demand for more and more functionality, there is a trend in avionics similar to other field of electronics to a higher functional integration. To save space, weight and power, functionalities are integrated onto one computing

L.M. Pinho et al. (Eds): ARCS 2015, LNCS 9017, pp. 15–27, 2015.
DOI: 10.1007/978-3-319-16086-3_2

platform. This trend is pushed further by integrating functionalities of different criticality levels onto the same platform to so called mixed-criticality systems.

Functionalities of different criticality levels on one shared (multi-core) platform require that these functionalities cannot interfere with each other or with the entire system. To manage this interference issue, temporal separation and spatial separation are essential to grant a safe and secure system operation. The Input/Output (I/O) subsystem is a central part, because almost every function needs I/O for its operation. Since I/O is an often underestimated problem, this paper focuses on I/O. Temporal separation means having separation in the time domain. For example, it is guaranteed that an I/O device has a granted transfer rate or maximum transfer time [1]. Spatial separation means having separation in the address space domain. For example, it is assured that an I/O device only writes into a distinct address range or memory area belonging to a distinct functionality or application [2]. A particularly underestimated issue in I/O handling are (message-signaled) interrupts. Message-signaled interrupts are the ubiquitous type of interrupts in modern memory-mapped I/O subsystems and can be considered as small Direct Memory Access (DMA) write packets (e.g. with only 4 Byte payload). If there is no spatial separation for interrupts, an erroneous I/O device can trigger any interrupt of the system-on-chip of the processor or manipulate any memory-mapped control register like triggering reset of all processing cores. Such a situation could lead to a complete system failure [2] [3]. Therefore, it is common in today's avionics and similar highly safety-critical systems to effectively turn off all interrupts and handle I/O via polling. This is a very resource-consuming and ineffective, but a safe approach to solve the problem. Further constraints are the use of Commercial Of–The–Shelf (COTS) components, low complexity, determinism and predictability (cf. Section 3).

The challenge is that available embedded processor architectures do not offer spatial separation means for interrupts like an Input/Output Memory Management Unit (IOMMU) with sufficiently fine granularity (cf. Section 3 and [2]). Server or high-end workstation processor architectures providing such means (cf. Section 2 and [4] [5]) are not usable for embedded real-time systems because of size, weight, power, cooling, harsh environmental conditions, certification considerations, etc. Further constraints are the use of Commercial Off–The–Shelf (COTS) components. This is essential to keep costs low for products with low piece numbers / volume like aircraft. A fully customized design of a processor chip or system-on-chip is economically infeasible. For these reasons, this paper does not discuss the design of interrupt controllers or IOMMUs. Instead, it focuses on an approach to extend available embedded COTS processors or system-on-chip by additional means to provide spatial separation for interrupts with the least possible impact on performance.

The contribution of the Safeguarding Interrupts (SgInt) concept of this paper is an efficient, high-performance and safe interrupt handling approach for highly safety-critical systems. It enables spatial separation at interrupt level in systems that does not have already built-in means. This concept is a reusable and general solution, which is scalable from a single (multi-core) processor to a multi

(multi-core) processor system and builds on available COTS chip solutions. The SgInt concept uses a source / origin ID check in the Non-Transparent Bridge (NTB) with an exclusive address range within the NTB aperture for interrupts of one distinct I/O device in combination with a dedicated alias page in the processor only containing the interrupt triggering register as mapping target. Furthermore, the paper contributes a implementation and an application of the SgInt concept in context of hardware-based I/O virtualization (cf. Section 2). The result of the presented practical evaluation is that the performance in terms of transfer time and transfer rate of the SgInt concept is by about 0.04% better than state-of-the-art doorbell interrupt handling.

To our best knowledge, we are the first to discuss an interrupt separation solution for single (multi-core) processor systems and multi (multi-core) processor systems in mixed-criticality embedded real-time systems that do not provide adequate means for interrupt separation.

2 Related Work

The application context of this paper is hardware-based I/O virtualization (cf. [1,2,6]). This is the hardware-managed sharing of I/O in virtualized embedded systems. Virualized embedded systems are systems where multiple virtual machines or application partitions are running on a shared computing platform managed by virtual machine manager or hypervisor. The key point is that the sharing or virtualization management is offloaded to hardware. This hardware management provides a Physical Function (PF) (management interface) and several Virtual Functions (VFs) interfaces (application interfaces) [7]. A memory-mapped I/O like PCI Express (PCIe) serves as basic I/O technology. This allows to map the PF to a control partition or hypervisor. The VFs are mapped to the corresponding application partitions. Already available means for memory management and mapping like Memory Management Unit (MMU) and IOMMU ensure the spatial separation between the application partitions and I/O interfaces.

Non-transparent bridging in context of PCIe is the non-transparent connection of two dedicated tree-like (single-root) PCIe hierarchies or address spaces together to enable multiple processors to communicate and exchange data [8]. A (single-root) PCIe hierarchy or address space is a tree-like topology with maximally one Central Processing Unit (CPU), master or root. Therefore, a communication between two root or CPUs is originally not possible. To solve this issue, an NTB connects two PCIe hierarchies by presenting itself as an end-point to both PCIe hierarchies. An NTB is constructed by two end-points back to back with an address translation functionality. Each side of an NTB opens an address window (aperture) from one PCIe single root hierarchy to the other PCIe single root hierarchy. The behavior of an NTB is considered as non-transparent, since the NTB and its address translation feature has to be setup before it allows to exchange data. It is not checked if a device or function is allowed to transfer data to a distinct destination. Interrupts are transferred over an NTB by the so-called doorbell mechanism. This mechanism consumes the interrupt on the first side of the NTB and newly generates the interrupt on the second side and transmits it to the processing unit. It is not checked if a device or function is allowed to trigger an interrupt. The current concept uses NTB technology in a different way than formerly intended to enable multi-processor communication. It extends NTBs to enable spatial separation for interrupts of shared PCIe devices in a single (multi-core) processor or multi (multi-core) processor system.

[9] uses PCIe interconnect, NTB and Intel VT-d to share a PCIe Single Root I/O Virtualization (SR-IOV) network card among multiple Intel Xeon hosts in the IT-server domain. It is suggested to use a dedicated address window in the NTB to transfer interrupts from one NTB side to the other instead of using the doorbell mechanism to improve performance. The interrupt remapping feature of Intel VT-d – the Intel implementation of an IOMMU – is able to check if a device or function is allowed to trigger an interrupt [4] [10]. AMD provides a similar technology as part of AMD-Vi or AMD IOMMU [5] [11] [12]. In contrast to this, the current paper uses PCIe interconnect, NTB technology without an IOMMU – like Intel VT-d – to share a PCIe SR-IOV or PCIe multifunction device while still providing spatial separation for data transactions and interrupts in a mixed-criticality real-time embedded system. The current concept presents a more general interrupt separation solution, which does not rely on special interrupt separating features of Intel VT-d or AMD IOMMU.

[6] uses NTB technology to emulate an external IOMMU to provide spatial separation for data transactions of I/O devices like the separation feature of an IOMMU for a single (multi-core) computing host lacking an IOMMU. It is enforced that transactions (for example a DMA write) initiated by I/O device(s) flow over the NTBs. The control engine in the NTB checks the target address and source / origin ID (e.g. PCIe ID) of these transactions. A rule set in the control engine (e.g. white list) decides whether to block the transaction or pass the transaction and translate the target address to the defined target address in the (bus) address space on the other side of the NTB. [13] extends this idea to provide spatial separation for sharing I/O devices among multi (multi-core) processor systems which usually do not have means for separation like an IOMMU. The current paper extends this approach to increase the separation granularity further to provide spatial separation also for interrupts of I/O devices in a single (multi-core) processor system as well as a multi (multi-core) processor system, whose processors lack means to separate interrupts. In addition to the origin / source ID check in the NTB, the SgInt concept uses an exclusive address range (page) within the NTB aperture for the interrupts of each I/O device. Mapping target for this interrupt page is a dedicated page (alias page) in the processor that only contains the interrupt triggering register.

3 SgInt (Safeguarding Interrupts)

A fundamental assumption is a static system configuration proving low complexity. This is prioritized over dynamic flexibility to obtain a predictable and deterministic system behavior. Determinism and predictability is an essential prerequisite to moderate the effort for the required assurance or certification process of a safety-oriented and security-oriented development project like in avionics [14]. Another assumption is the use of COTS components. This is essential to keep costs low for products with low piece numbers / volume and long life cycles like aircraft.

The SgInt concept enables the safeguarding of interrupts for hardware-based I/O virtualization for mixed-criticality embedded real-time systems using non-transparent bridges in single (multi-core) processor systems as well as in multi (multi-core) processor systems.

The already described separation mechanism (cf. [6] and [13]) using NTBs with additional checking of the target address and source / origin ID can also be extended to safeguard interrupts (cf. Figure 1). Message-signaled interrupts are the omnipresent type of interrupts in modern serial high-speed memory-mapped I/O standards, since dedicated interrupt wires are no longer available. Message-signaled interrupts can be considered as small DMA write transactions (e.g. 4 Byte). The SgInt concept uses an exclusive entry in the rule set in the NTB per I/O device (or PCIe function or application interface) for its associated interrupts (cf. Figure 1). An entry represents an address window or memory page of a typical size of 4kB. The mapping target of this entry or page is a memory-mapped page containing the interrupt trigger register of the interrupt controller. The interrupt trigger register converts the message-signaled interrupt to an actual interrupt. The access to this NTB entry is controlled by the control engine in the NTB performing the origin/source ID check (cf. Figure 1). This means that only the message-signaled interrupt sent by a distinct I/O device (or PCIe function or application interface) can pass this special interrupt window over the NTB. However, the protection granularity at page level is still not sufficient for a safe and secure handling of interrupts. The mapping target of this interrupt entry or page is a page containing this interrupt trigger register and a variety of additional control registers. Since a message-signaled interrupt is a DMA write packet, it is able to manipulate any memory-mapped control register within the target page. For example, an interrupt can trigger interrupts associated with other devices or other system-on-chip interrupts or processor interrupts by targeting another interrupt trigger register (cf. Figure 1). In addition, an interrupt can manipulate any memory-mapped control register of the target page like triggering the reset of all processing cores (cf. Figure 1). This could lead to a complete system failure. To prevent this, the granularity or precision of the origin/source ID check needs to be increased. A possibility is to isolate the interrupt trigger register within a page. This means, a page only contains this single interrupt trigger register or an alias register to this interrupt trigger register. An I/O device (or PCIe function or application interface) that is allowed to access this page can only change this register and nothing else since the page does not contain more control registers. Such a page is called alias page or page with an alias to the interrupt trigger register (cf. Figure 1).

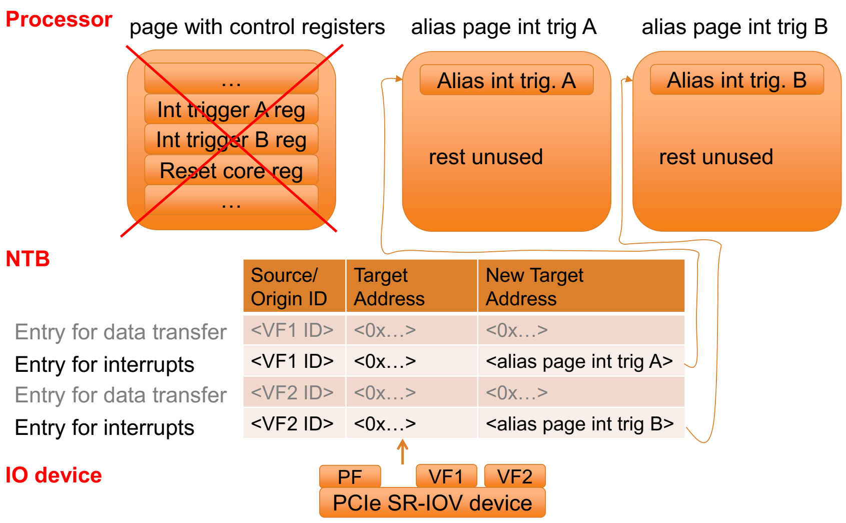

Fig. 1. SgInt (Safeguarding Interrupts): Origin/source ID check in combination with alias pages

To demonstrate an application, we have implemented the SgInt concept in the context of sharing a DMA-capable multi-function PCIe I/O card in a mixed-criticality embedded processing platform. Figure 2 depicts the implemented system setup. A Xilinx VC709 FPGA evaluation board is used as PCIe I/O card. A PLX 8749 chip serves as PCIe switch containing the two non-transparent bridges. The two system hosts are built up by two Freescale QorIQ P4080 Development Systems (P4080DS). The P4080 platform is a PowerPC-based embedded multi-core processing platform and a reference model of the Freescale QorIQ series. Freescale's Software Development Kit (SDK) Version 1.2 is used as software foundation. The avionics industry considers the PowerPC architecture-based P4080 platform as a platform candidate for embedded avionics systems [1,2,6,14,15].

For simplicity reasons, the demonstration system considers only two multi-core processors and one DMA-capable and bus-mastering capable PCIe card with two physical PCIe functions. Physical function (PF) 0 is used as management interface and application interface 1 and PF 1 servers as application interface 2. However, the SgInt concept is scalable from one application interface per processing host to multiple application interfaces per processing host with one NTB with multiple windows or multiple NTBs. An additional reason for using only two physical functions is that the SR-IOV capability of the Xilinx VC709 FPGA evaluation board is not compatible to the P4080DS. The Xilinx SR-IOV IP-core requires the optional PCIe Alternative Routing-ID Interpretation (ARI) extension to address VFs. The P4080DS does not support PCIe ARI [1]. Xilinx has

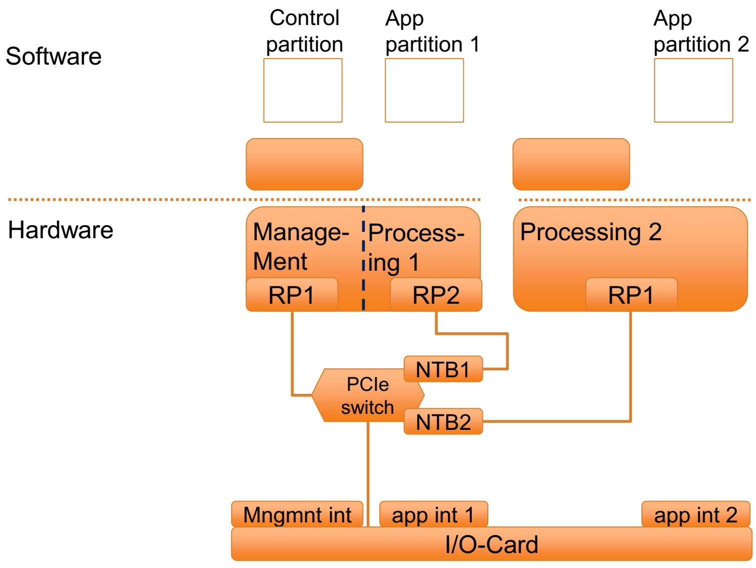

Fig. 2. Implementation of the Concept

confirmed this and we are in dialog with Xilinx to eliminate this limitation in the succeeding generation of Xilinx FPGAs.

The demonstration system encompasses two multi-core processors. If desired, the management part can be outsourced to a third management processor. The left multi-core processor runs the management section and one application section. One core and one dedicated (bus) address space or PCIe hierarchy or root port (RP) takes over the tasks of the management section. A second core and a second dedicated address space or PCIe hierarchy or root port runs one application section. This part of the demonstration system is representative to apply the concept in a single (multi-core) processing system. To be able to evaluate the concept also in multi (multi-core) processor systems, the additional second multi-processor takes over the task of another application section. This management control partition sets up the system, controls the main address space and controls the NTBs and the management interface of the I/O card. Each of the dedicated address spaces of a application section is connected to the main address spaces by an NTB. Application partition 1 running on the first multi-core processor is directly mapped to application interface 1 of the I/O card whereas application partition 2 running on the second multi-core processor is mapped to application interface 2 of the I/O card. The IOMMU of the P4080 platform has no means to safeguard interrupts of multiple PCIe devices or PCIe devices with multiple functions [2] [16]. Therefore, the spatial separation of interrupts of the two application interfaces are performed by the SgInt concept.

4 Evaluation

4.1 Evaluation Setup

The evaluation of the enforcement of the source / origin ID check for interrupts is analyzed with the following procedure:
The control partition sets up the NTB and the PCIe advanced error reporting (AER) registers. A DMA write transaction followed by a synchronization interrupt is triggered. The interrupt contains an allowed origin / source ID and target address, which complies to the rule set. Application partition 1 waits for the receiving of the interrupt while a time out timer is started. In this case, the receiving of the interrupt is expected and no time out should occur. The AER registers report no error. As a next step, another DMA write transaction with a synchronization interrupt is triggered. Here, the interrupt contains a target address associated to a disallowed origin / source ID. Application partition 1 waits for the receiving of the interrupt while a time out timer is started. The receiving of the interrupt is expected but does not occur and the time out occurs. The AER registers report the header and the first 32 data bits of the blocked packet.

The evaluation of the performance overhead (transfer time, transfer rate) of the SgInt concept is investigated with the following procedure:
The control partition configures the NTB and the I/O card. It is defined by the management interface that application interface 1 is assigned 50% of the available transfer rate and application interface 2 is assigned 50% of the available transfer rate. DMA read and write transactions hit the two application partitions. The transfer time and transfer rate of transactions are measured including the low-level software overhead and synchronization interrupts. The DMA transactions are composed of a number of 128 Byte-sized packets sent back to back. The number of packets is increased from 1 to 255. For each packet count, the measurements are run 100 times. The described measurement procedure is executed twice. One time it is conducted using the presented SgInt concept with interrupt separation. The other time it is performed using the state-of-the-art doorbell interrupt mechanism without separation (cf. Section 2 and [8]). Then both results are compared.

4.2 Evaluation Results

The evaluation result of the enforcement of the source / origin ID check for interrupts is given by the following output:

```
Test case 1: ID ok -> pass
//setup NTB
  (application interface 1 (source ID=0C00)
   is allowed to trigger sync interrupt
   (address 0xE070A140, data 0x13)
//trigger DMA write with end interrupt
//printout of application partition 1
```

```
  <no error>
//printout of PCIe Advanced Error Reporting (AER) registers
  PLX_AER_HEADER0 +0x3EFD0: 0x00000000
  PLX_AER_HEADER1 +0x3EFD4: 0x00000000
  PLX_AER_HEADER2 +0x3EFD8: 0x00000000
  PLX_AER_HEADER3 +0x3EFDC: 0x00000000

Test case 2: ID violation -> block
//setup NTB
  (source ID=0C03 is allowed
   to trigger sync interrupt
   (address 0xE070A140, data 0x13);
   application interface 1 (source ID=0C00)
   is NOT allowed to trigger the interrupt)
//trigger DMA write with end interrupt
//printout of application partition 1
  // indicating time out occurred
  ntbxpcieappdrv wait for interrupt:
  Operation not permitted
//printout of PCIe Advanced Error Reporting (AER) registers
  PLX_AER_HEADER0 +0x3EFD0: 0x40000001
  PLX_AER_HEADER1 +0x3EFD4: 0x0C00000F //ID=0C00
  PLX_AER_HEADER2 +0x3EFD8: 0xE070A140
  PLX_AER_HEADER3 +0x3EFDC: 0x13000000
```

Figure 3 shows the relative difference of the transfer time between the SgInt concept and no interrupt separation, whereas Figure 4 depicts the relative difference of the transfer rate between the SgInt concept and no interrupt separation. For the transfer time, the values of the SgInt concept are about 0.04% (for writes) to 0.08% (for reads) lower than the values of no interrupt separation. In case of transfer rate transactions, the data of the SgInt concept are about 0.04% (for writes) to (0.09%) for reads higher than the data of no interrupt separation.

5 Discussion and Impact

In test case 1 of the interrupt source / origin ID check, the interrupt is allowed to pass and to trigger the interrupt in the processing system. In test case 2, the origin ID of the actual sent interrupt does not comply to the origin ID of the corresponding target address in the rule set in the NTB. Therefore, the interrupt is blocked. Concluding, the origin ID check of the SgInt concept shows that the spacial separation in dependency of the origin ID can be enforced for interrupts.

The transfer time figure (cf. Figure 3 and Section 4.2) shows that the SgInt concept with separation has a 0.04% better transfer time than the state-of-the-art NTB configuration without interrupt separation. The reason for this can be explained by the nature of the state-of-the-art doorbell interrupt mechanism [8]. This mechanism consumes the interrupt on the first side of the NTB and generates a new interrupt on the second side and transmits it to the processing

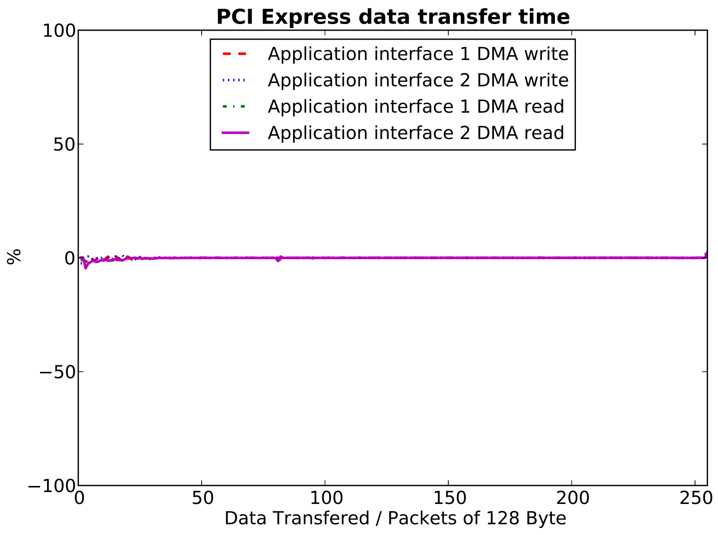

Fig. 3. Relative difference between the transfer time results using the SgInt concept and no interrupt separation

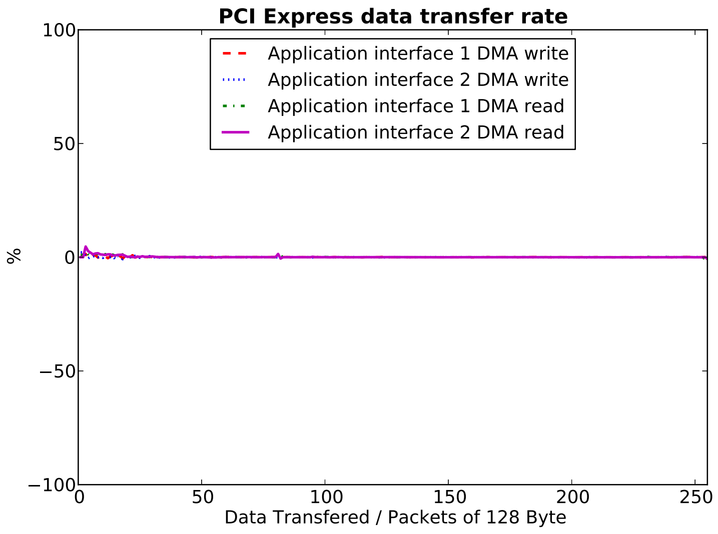

Fig. 4. Relative difference between the transfer rate results using the SgInt concept and no interrupt separation

unit. In contrast to this, the presented SgInt concept handles the interrupt like any other data packet passing the NTB. After the source / origin ID check decided to pass the packet through the NTB window, the packet is forwarded and its target address is translated. This forwarding process inclusive source / origin ID check is marginally more efficient than the traditional way of consuming and recreating without source / origin ID check of the interrupt. The figure for the transfer rate (cf. Figure 4) confirms the statements of the transfer time.

The the demonstration system considers two multi-core processors sharing one I/O card with two application interfaces. However, the scalability of the SgInt concept ranges from one application interface per processing host to multiple application interfaces per processing host with one NTB with multiple windows or multiple NTBs.

A really relevant item is that the SgInt concept can provide spatial separation for interrupts in systems, which do not have an IOMMU or have an IOMMU that is not able to safeguard interrupts. In contrast to Intel server systems using Intel VT-d [4], embedded real-time systems do not have means to protect interrupts. For these systems, the presented SgInt concept is a real benefit. The SgInt concept requires that the processor platform provides special alias pages encapsulating an interrupt trigger register (cf. Figure 1 and Section 3). The most of Freescale's PowerPC-based processors (e.g. the Freescale QorIQ families) provide three to four of such special alias pages for interrupt trigger registers. This allows to provide spatial separation for three to four application interfaces using safeguarded interrupts per processor. The SgInt concept fulfills the required separation and offers a growth of 300-400% in protected interrupts. Since the avionics industry currently has most certification-related experience for critical avionic components for the PowerPC architecture, future developments based on this architecture are focused. The aircraft certification authorities EASA and FAA lately recommended to restrict the usage of multi-core processors for safety-critical systems due to safety concerns to dual-core processor systems at the moment [17]. For future (multi-)processor systems making use of (multiple) dual-core processors (like Freescale's P5020), the presented SgInt concept is practically applicable and has still spare resources for extensions.

6 Summary and Conclusion

The presented SgInt concept enables the safeguarding of interrupts in single (multi-core) processor systems and multiple (multi-core) processor systems.

The SgInt concept uses an exclusive page within the NTB aperture for interrupts of one distinct application interface and a dedicated page in the processor only containing the interrupt triggering register as mapping target in addition to the source/ origin ID check in the NTB. The evaluation results of the SgInt concept reveals that SgInt concept slightly outperforms state-of-the-art doorbell interrupt handling in transfer time and transfer rate (by about 0.04%). The SgInt concept can provide spatial separation for interrupts in systems, which do not have an IOMMU or have an IOMMU that is not able to safeguard interrupts. This is especially important for safety-critical embedded real-time systems since these systems usually do not have means to protect interrupts. This SgInt concept is not limited to safeguard device interrupts. It can also be applied to provide spatial separation for inter-processor communication interrupts.

While this paper focuses on avionics, the results are applicable to adjacent markets which have similar stringent security and safety requirements such as automotive, railway and industrial control.

Acknowledgments. This work was supported by the projects ARAMiS and SIBASE funded by the German Federal Ministry of Education and Research (BMBF) under the funding ID 01IS11035R and 01IS13020B. We also thank Xilinx and PLX Technologies for their support. Special thanks also to our colleagues from Airbus Defence and Space, especially Peter Ganal.

References

1. Muench, D., Paulitsch, M., Herkersdorf, A.: Temporal separation for hardware-based I/O virtualization for mixed-criticality embedded real-time systems using PCIe SR-IOV. In: International Conference on Architecture of Computing Systems (ARCS) (2014)
2. Muench, D., Isfort, O., Mueller, K., Paulitsch, M., Herkersdorf, A.: Hardware-based I/O virtualization for mixed criticality real-time systems using PCIe SR-IOV. In: International Conference on Embedded Software and Systems (ICESS) (2013)
3. Pek, G., Lanzi, A., Srivastava, A., Balzarotti, D., Francillon, A., Neumann, C.: On the feasibility of software attacks on commodity virtual machine monitors via direct device assignment. In: ACM Symposium on Information, Computer and Communications Security (ASIA CCS) (2014)
4. Intel: Intel Virtualization Technology for Directed I/O (VT-d spec) (2011)
5. AMD: AMD I/O Virtualization Technology (IOMMU) Specification Rev2.0 (2011)
6. Muench, D.: IOMPU: Spatial Separation for Hardware-Based I/O Virtualization for Mixed-Criticality Embedded Real-Time Systems Using Non Transparent Bridges (TR-TX4-399). Technical report, Airbus Group (2014)
7. PCI-SIG: Single Root I/O Virtualization and Sharing Specification 1.1 (2010)
8. Regula, J.: Using Non-transparent Bridging in PCI Express Systems. Technical report, PLX (2004)
9. Tu, C.C., Lee, C.T., Chiueh, T.C.: Secure I/O device sharing among virtual machines on multiple hosts. In: International Symposium on Computer Architecture (ISCA) (2013)
10. Nguyen, T.L., Carbonari, S.R.: Message Signaled Interrupt Redirection Table (2004)
11. Hummel, M.D., Strongin, G.S., Alsup, M., Haertel, M., Lueck, A.W.: Address Translation for Input/Output (I/O) Devices and Interrupt Remapping for I/O devices in an I/O Memory Management Unit (IOMMU) (2006)
12. Serebrin, B.C., Wiederhirn, J.F., Cooper, E.M., Hummel, M.D.: Guest Interrupt Manager that Records Interrupts for Guests and Delivers Interrupts to Executing Guests (2009)

13. Muench, D., Paulitsch, M., Hanka, O., Herkersdorf, A.: MPIOV: scaling hardware-based I/O virtualization for mixed-criticality embedded real-time systems using non transparent bridges to (multi-core) multi-processor systems. In: Conference on Design, Automation and Test in Europe (DATE) (2015)
14. Muench, D., Paulitsch, M., Honold, M., Schlecker, W., Herkersdorf, A.: Iterative FPGA implementation easing safety certification for mixed-criticality embedded real-time systems. In: Euromicro Conference on Digital System Design (DSD) (2014)
15. Jean, X., Gatti, M., Berthon, G., Fumey, M.: MULCORS - Use of Multicore Processors in airborne systems. Technical report, EASA (2012)
16. Freescale: P4080 QorIQ Integrated Multicore Communication Processor Family Reference Manual (2011)
17. FAA: Position Paper Certification Authorities Software Team (CAST) CAST-32 Multi-core Processors (2014)

http://www.springer.com/978-3-319-16085-6

Architecture of Computing Systems – ARCS 2015
28th International Conference, Porto, Portugal, March 24-27, 2015, Proceedings
Pinho, L.M.; Karl, W.; Cohen, A.; Brinkschulte, U. (Eds.)
2015, XVIII, 249 p. 19 illus., Softcover
ISBN: 978-3-319-16085-6