Redundant I/O interconnect

U. Helmich
M. Becht
M. J. Becht
J. R. Easton
R. K. Errickson
T. Gehrmann
S. G. Glassen
S. R. Greenspan
F. Koeble
H. Lehmann
C. Mayer
J. S. Nikfarjam
F. A. Schumacher
W. Storz

The outstanding reliability, availability, and serviceability (RAS) characteristics of IBM mainframe computers are among the features that gained the IBM eServer™ family its reputation as a leading platform for business-critical applications. The aim now is to further improve IBM System z9® RAS by introducing redundant I/O interconnect (RII) as a building block of enhanced book availability and recovery scenarios. RII provides a means of maintaining I/O connectivity during planned or unplanned outages in a way that is transparent to the operating system and customer applications. The mechanism that meets this requirement is the provision of an alternate path to the I/O cage, which provides high-bandwidth I/O slots to enable a higher number of I/O ports per card. This paper discusses the I/O subsystem hardware and firmware aspects of RII.

Introduction

One of the many improvements of the IBM System z9* compared with its predecessors is the ability to perform a broader variety of service actions without causing any disruption to a customer's business. Another paper published in this issue focuses on a category of service actions that allow the customer to add or replace processor books to the processor cage of a system—the enhanced book availability (EBA) features that reduce planned outages for book hardware upgrades and maintenance [1]. This approach first requires a solution to the problem presented by a limitation that originates from the classical organization of the self-timed interface (STI) network [2]. This is a proprietary point-to-point network that was first introduced with the third generation of IBM S/390* CMOS servers (G3). It fans out the I/O bandwidth from the memory bus adapters (MBAs) through one or more multiplexer switch chip levels to the channel cards, located in the I/O frame. The MBA card is plugged to the processor book and provides the connection to the system bus. The channels provide connectivity to I/O devices outside the server, such as redundant arrays of independent disks and other types of storage systems, network adapters, and even other IBM zSeries* servers (coupling). On systems before the z9*, the topology of the STI network was organized as a special case of a general network, a collection of I/O trees in which the port of an MBA was at the root of each tree, and the channels could be seen as leaves. While this is an appropriate and robust approach to gain the required I/O fan-out, it has the limitation that only a single path exists from an MBA to a channel. Removing an MBA card or a processor book (to which the MBA cards are plugged) is disruptive to all channels in the tree. The redundant I/O interconnect (RII) addresses this limitation by adding a connection between every two I/O trees at the first multiplexer switch chip (TNT) level.

Hardware and software engineers have different concepts of redundancy. While redundancy is avoided as much as possible in code development, it is a key aspect of high-availability hardware design. This is especially true for zSeries systems, where availability is of paramount importance. Most critical components, such as the distributed converter assembly (DCA) and cage controller, are present in an $N+1$ configuration [3, 4]. In the memory subsystem, redundancy exists on several levels. Each memory chip provides spare memory lines. Entire spare memory cards are plugged in and used to compensate if an entire card should fail. In the I/O subsystem, the redundancy concept is extended to a guideline that advises the customer to provide multiple logical paths to a device wherever possible. Different logical paths should make use of different channel and switch hardware. Because this approach cannot always be taken, or it cannot be taken for all I/O devices, the redundancy provided by hardware was taken one step further with the introduction of RII.

0018-8646/07/$5.00 © 2007 IBM

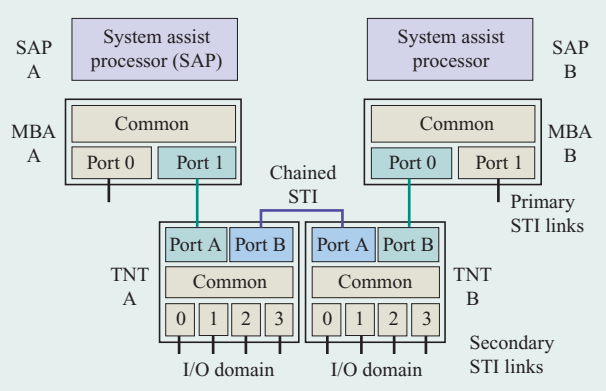

Figure 1

STI interconnections between MBA and TNT chips.

Prior systems were already able to bypass failing units in the I/O network by exploiting redundant I/O connections that were designed for some other purpose. The first hardware implementation of a redundant interconnection to a channel-attached device was called *STI chaining*. It was introduced with the common I/O platform (CIOP) cards on the IBM S/390 G5/G6 server. STI chaining provided a way to attach multiple CIOP bridge cards to a single MBA port. This was desirable because the bandwidth of each port of an MBA chip was much larger than the bandwidth of a single bridge card. Instead of attaching open-ended chains of bridge cards to an MBA, the endpoints of every two chains were connected to form a closed chain of CIOP cards between two MBA ports, thus forming a redundant connection to each card.

On IBM eServer* z900 and z990, a different approach was taken. The fan-out of MBA bandwidth to the I/O network was improved by cascading STI switch chips into multiple levels of multiplexers (increasing the depth of the I/O tree). With the increased density of packaging units, more than one CIOP channel could be placed on a single field-replaceable unit (FRU). The chaining cable between channels on the same board was replaced by on-board wiring. This new CIOP card design improved the reliability of the chaining link, but the redundant connection provided by the older cable approach had to be sacrificed.

On System z9, the idea of an RII emerged again. The RII feature focuses on providing the redundant path that is required by EBA. A new link was added between different I/O trees at the highest possible point in the I/O network—the first-level multiplexer. This connection can be used to disable or remove a processor book, an MBA, or an MBA–TNT (Triton-T) cable without impact on I/O connectivity. The feature is also exploited by recovery in the event that one of the above components fails and cannot be recovered. This paper explains some of the hardware and firmware aspects of this major building block that had to be invented to support CBA. We further discuss how recovery scenarios make use of RII.

Overview

The System z9 channel subsystem (CSS) presents two aspects. One is the I/O subsystem [5], also known as the physical CSS, which comprises the I/O-related hardware and its associated firmware support and control structures. The second aspect is the multiple-logical-channel subsystem (MCSS) [6], which provides for the logical replication of facilities within the CSS. At the MCSS level, RII is completely transparent, but at the physical CSS view it is not.

The I/O subsystem is further broken down into the I/O subsystem hardware and the I/O subsystem Licensed Internal Code (LIC), usually referred to as *system firmware*. The firmware maintains data structures and configuration information that reflect the state of the hardware. It performs all hardware accesses and controls state transitions.

More complete information on I/O chips and the STI network can be found in [2, 7, 8]. Changes to the hardware that were required in order to make RII possible are described in subsequent sections of this paper.

Exploiters of the RII feature are the new enhanced book availability (EBA) scenarios as well as recovery. On System z9 it is possible to add a new processor book or repair a defective book without stopping traffic to affected I/O units. The RII is required to make this possible. If an MBA port is seen as the root of an I/O tree, the redundant connection is added as a second root to each tree. This can be seen in **Figure 1,** where all I/O units attached to a first-level TNT multiplexer chip (an I/O domain) are accessed via two different paths. The default path to an I/O domain connects MBA and TNT ports (light blue). This is the only path to an I/O domain on systems prior to z9. On z9 the darker blue port was added to the TNT chip. This port allows a connection between TNT chips at the same level in two different I/O trees, the chained STI link. With the help of this chained link, an alternate path to an I/O domain can be established. For example, the loss of the default path from MBA A to TNT A can be compensated by routing traffic to the I/O domain attached to TNT B via the link between MBA B and TNT B followed by the chained link. To make the best use of this feature, the only requirement is that the two MBAs should be plugged to different processor books.

When a new book is concurrently added to a system by an EBA concurrent book add (CBA) service action, the I/O subsystem becomes unbalanced. In many cases the

new book also brings new MBAs into the system. For example, if a single-book configuration is upgraded to a two-book configuration, at first all old I/O domain pairs are attached to the old book and all new I/O domain pairs are connected to the new book. This situation creates two problems that can be solved by rebalancing the I/O, a second step of the CBA. The first problem is that the I/O is not evenly distributed between the books, which might create an I/O performance penalty. The second problem would be a later repair action to a book. Because the I/O is not plugged across different books, it would be necessary to stop all I/O attached to the book that is replaced. The RII feature makes it possible to run a domain pair via a single MBA connection. During such a transition phase, the other MBA connection to the pair can be plugged to a new position. When all I/O domain pairs are evenly distributed across books, the traffic can be switched back to the default STI connections. I/O operations are not interrupted during the rebalancing.

A second EBA scenario that exploits RII is concurrent book repair (CBR). A processor book that contains defective components but is still able to run I/O can be replaced without stopping customer operations (hot book replace). The CBR service action must evacuate all resources from the book before it can be safely removed. I/O can be taken from the book without interrupting any operations provided that I/O domain pairs are plugged across different books. In this case, the RII feature provides connectivity to those domains that have a default path to the book that is being replaced.

In addition to the need to support CBA and CBR, the RII feature can be exploited by I/O subsystem recovery. RII was not designed with a focus on recovery, but in many cases recovery can prevent the loss of an I/O domain by switching to the alternate path. A failure of the MBA chip or the connection to the first-level TNT can be recovered by redirecting traffic to the alternate path.

STI switch chip

The TNT application-specific integrated circuit (ASIC) chip is the next generation of the STI switch chip used on previous zSeries servers. This chip was based on the z990 STI switch chip, with several enhancements, improvements, and new functions. The TNT STI switch chip is used to increase the number of STI links in z9–109 systems and to connect I/O devices through the enhanced STI (eSTI) link to the MBA chip in the central electronic complex (CEC) book.

The following are the new features for TNT:

- A second 2.7-GB/s eSTI link was added to provide an alternate system path for concurrent node servicing.
- All multispeed STI (mSTI) links now support 2 GB/s in addition to the previously supported speeds of 333 MB/s, 500 MB/s, and 1 GB/s.
- A 1-KB buffer was added for each sink port to improve performance. This change minimizes the effect of a slow sink port throttling north port traffic.
- First-error-capture registers were added to help isolate errors by capturing the first error that occurs and no other errors until the register has been reset.
- The sense control packet storage was increased.
- The command reject storage for stray packets was increased.

There are three source links on the TNT chip: two eSTI ports and a single mSTI port. The eSTI provides connections to the 2.7-GB/s links. The mSTI provides connections to a 333-MB/s, 500-MB/s, 1-GB/s, or 2-GB/s link. In any given TNT application, either the mSTI or the eSTIs are used as the source port or ports.

Use of the mSTI source port is restricted to the connection between a first-level and second-level TNT; an alternate path is not supported in this configuration. Use of the eSTI ports is reserved for the first-level TNT system connection. In this application both of the eSTI ports are typically used, one as a primary path and the other as a secondary or backup path to the system. The primary path is always connected to an MBA-GX+, and the secondary path always to another TNT. In a cascading configuration in which a TNT is connected to another TNT, the first-level sink mSTI port is connected to a second-level TNT source mSTI port. This allows for higher fan-out of the STI links. A maximum depth of two TNT chips is supported.

Under normal conditions the primary link is timed, physically operational, and serves as the active path through which system traffic flows. Regarding the secondary path, there is a minimum requirement for the link to be maintained in a timed and operational state in order to provide early identification of failures and to schedule preventive maintenance. When an event occurs that affects the usability of the active path, an action is taken to swap active paths. This action is a code-initiated recovery swap that is described below.

TNT is a 9.3-mm ASIC chip implemented with IBM CMOS Cu-11 130-nm lithography technology. The chip uses seven levels of metal and has supply voltages of 1.35 V for the core, 1.94 V for the mSTI I/O circuits, and 3.3 V for the phase-locked loop (PLL) circuits. The chip resides on a 32-mm ceramic ball grid array package.

There are multiple operating frequencies in TNT, since the STI ports can be run at different speeds. The host logic runs at a frequency of 189 MHz, the eSTI interface runs at 1.136 GHz, and the mSTI interface can run at

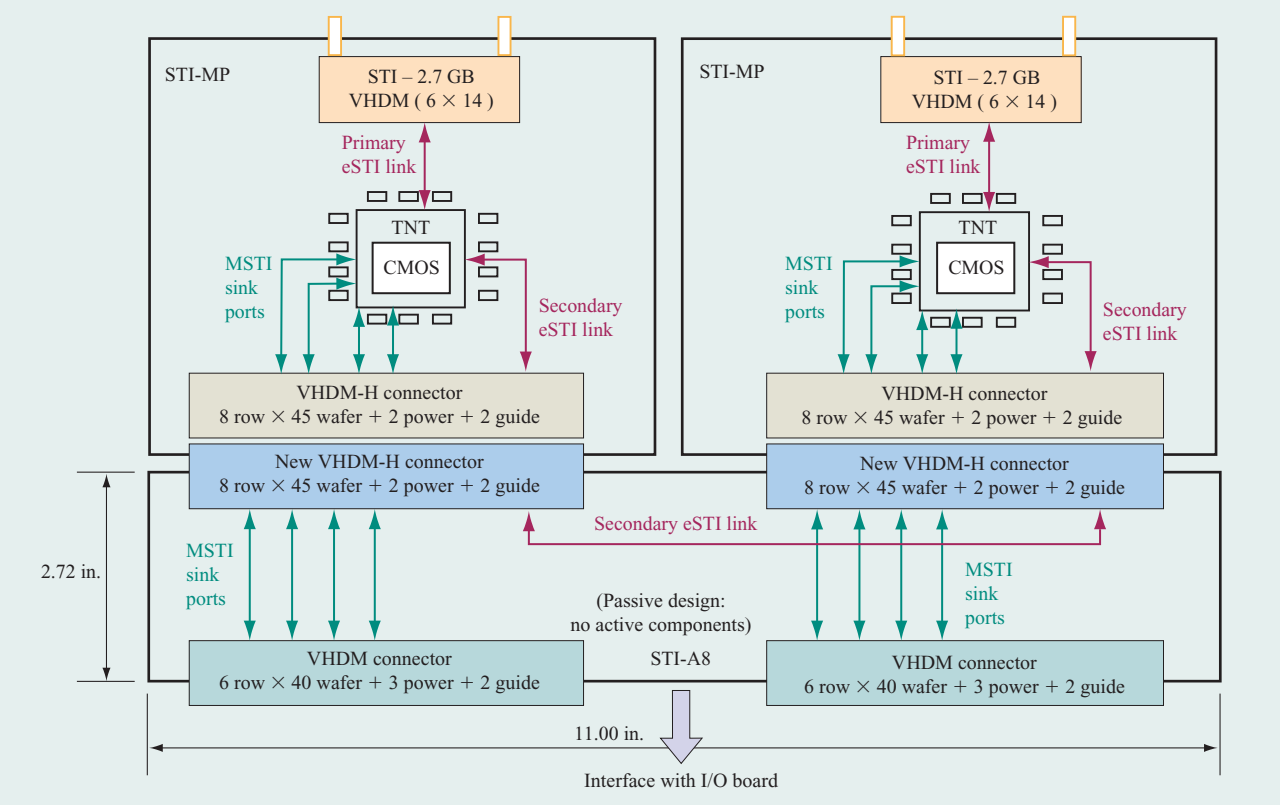

Figure 2

Mother-and-daughters card package.

1 GHz, 500 MHz, 250 MHz, or 166 MHz, depending on the configuration.

Measured power consumption in a configuration in which all four south ports are running at 1 GHz is approximately 11 watts. This configuration generates the greatest power consumption.

Performance measurements were made on the TNT chip by running the mSTI links at 500 MHz. Measurements were taken while running with 64-kilobyte × 6-byte I/O starts, with a 50/50 ratio of reads to writes. Results showed that the eSTI link ran at 100% of target, 2.7 GB/s, and the mSTI link ran at 103% of target, 1,029 MB/s in duplex mode (read and write). At the time the measurements were made, there was no available adapter that would take advantage of the 2-GB/s interface, so performance numbers are not available for the high-speed link.

Card packaging

The hardware that enables the RII consists of the TNT module, STI-MP (STI-multiport), STI-A8, and STI-A4 cards. The concept of adding an additional eSTI link to provide an alternate path to another I/O domain presented several challenges. There had to be a means to connect two TNT modules through an eSTI interface. The previous-generation STI switch card was a half-high full-length card with a very high-density metric (VHDM**) connector for an eSTI cable on one end and on the other, a VHDM connector for four mSTI ports that plugged directly into the I/O cage. The new STI card had to support an eSTI speed of 2.7 GB/s (an increase over the previous speed of 2 GB/s), and the maximum mSTI speed had to double to 2 GB/s. There were limitations on the extent to which the card could be redesigned. The mSTI connector on the card could not change because it had to remain compatible with the I/O cage, which was not going to be redesigned.

Each STI-MP card had to be an FRU to be able to support repair actions without affecting the other domain. A proposal to provide the alternate eSTI connection on the cable side of the card was rejected because of space limitations for an additional or larger connector on the STI-MP card. Because the I/O cage was not to be redesigned, providing the alternate eSTI connection on the I/O board was not an option.

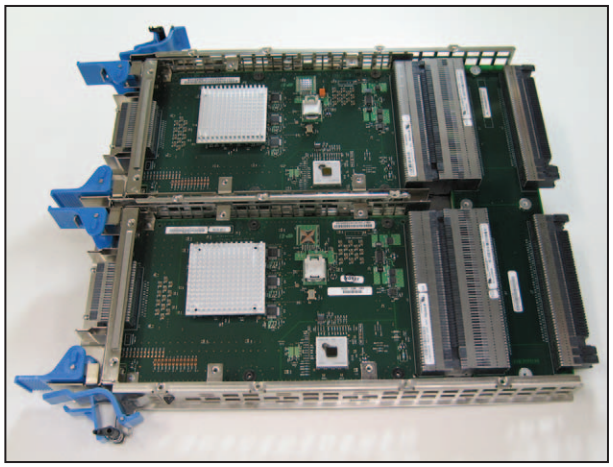

Figure 3

STI-MP/STI-A8 package: Two STI-MP cards (left, top and bottom) are plugged into one STI-A8 card (right).

The final card packaging solution was a mother-and-daughter concept consisting of two cards: the STI-MP card (daughter) and either the STI-A8 or STI-A4 card (mother). Two STI-MP cards plug into one mother card. **Figure 2** shows a logical representation of the card package.

In this approach, the alternate or secondary eSTI link is routed through a VHDM connector from one STI-MP to the mother card and to the other STI-MP card. The mSTI ports are wired straight through the mother card to the I/O board connector. Both types of mother cards (STI-A8 and STI-A4) are passive with no active components, which eliminates the potential for field failures on the card. The STI-MP card is approximately four inches shorter than the previous STI-M card to make room for the mother cards. With the addition of the second eSTI interface and another PLL on top of the existing components of the former STI-M card, care had to be taken in the STI-MP card design. With the additional eSTI interface, two more signal planes were added. Also, the VHDM connector to the mother card had to grow in order to carry this interface along with the extra ground planes for signal integrity. As the speed of mSTI doubled, wiring rules were more strict to preserve signal integrity. The STI-MP/STI-A8 package is shown in **Figure 3.**

In the I/O cage, seven I/O domains are available (I/O-A through I/O-G in **Figure 4**), each with connectivity to an STI-MP card, as shown in the figure. Each STI-MP is connected to four I/O card slots in each domain. The STI-MP/STI-A8 card package resides in slots 5, 14, and 23 in Figure 4. In slot 28, the STI-MP/STI-A4 package is used. The STI-MP on the bottom of I/O slot 28 does not have I/O connectivity because of the lack of space created by the presence of the DCAs in the back of the cage. Because

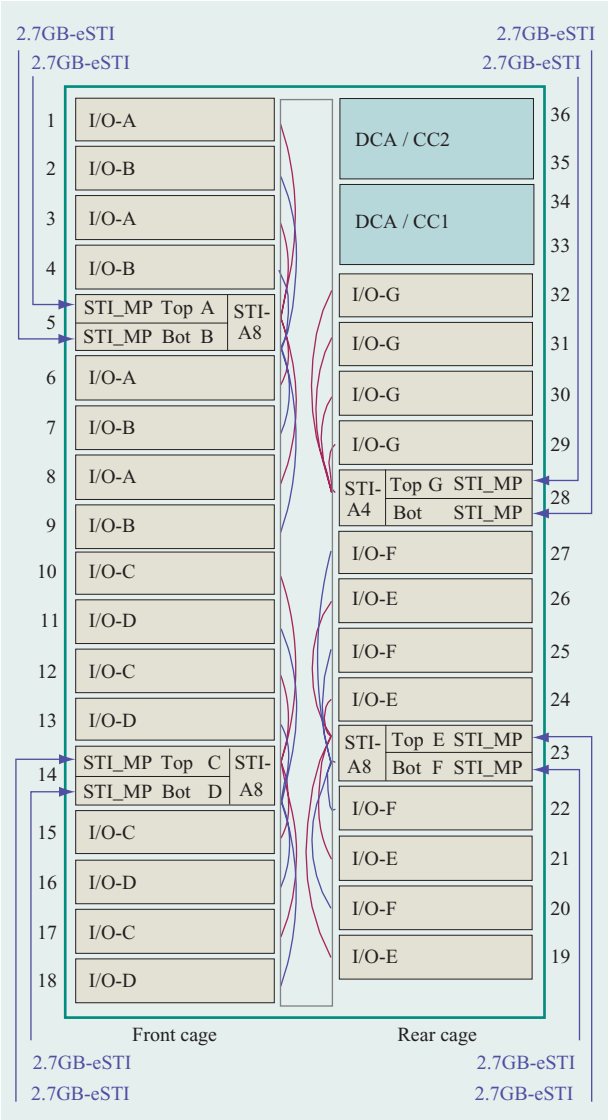

Figure 4

I/O cage (top view). (DCA/CC: distributed converter assembly/cage controller.)

there is no mSTI connector for the bottom of this slot, a separate mother card had to be designed to fit in the slot. The STI-A4 card fits only in slot 28 and provides a secondary eSTI connection to the STI-MP cards, and mSTI links to only the top STI-MP card in that slot. The bottom STI-MP in this slot provides only a secondary eSTI path for the top STI-MP card for the RII function and does not have direct I/O connectivity.

Firmware implementation

As a result of the organization of the I/O subsystem into separated I/O domains, information about traffic and

interrupts is grouped with an I/O-domain granularity. Such data sets are kept in several places in the hardware and firmware, and are indexed with the hardware address (HWA) of the domain. The new chaining link introduced with RII breaks this paradigm of isolation among different I/O domains. Traffic and interrupts associated with a channel in one domain can flow via two different MBAs, depending on the RII path selection. These different paths are associated with different HWAs. This creates a requirement that the firmware know the active path to a device because state information for the same channel is kept in different locations, depending on the path taken. For this reason, it is necessary to allow a path swap only under firmware control in order to give the code a chance to move information from the old data set to the new one. Also, the TNT chip can be operated in a mode in which a failing eSTI connection automatically causes a failover to the chained link. This mode is not used because firmware is not prepared to handle traffic on the new path without proper preparation. During the time window that exists before firmware is informed of the automatic switch, the automatic switch would cause the risk of a configuration mismatch between path information kept by firmware and the actual path selection in the hardware.

The downside of this approach is that a switch under code control takes longer than it would if the automatic hardware failover could be used. Further, all processors handling state information must be synchronized on a safe task boundary before the configuration update can be performed. In order to make path switches during service scenarios absolutely transparent to the operating system and the channels, the swap task must be designed for speed.

The firmware module that controls an RII path switch, also known as a *TNT alternate path swap task*, always executes a certain sequence of steps to drain all traffic in the affected I/O domain. Here we give a simplified summary of the steps that are required for a transparent support element (SE) controlled swap:

1. Quiesce interrupt processing: Disable interrupt handling for inbound traffic.
2. STI restriction: Disable outbound traffic, i.e., place STI access restriction on links.
3. Hardware swap: Disable inbound direct memory access (DMA) traffic and perform hardware path switch. DMA is stopped only during a very brief period while arbitration is disabled in the TNT chip.
4. Update or modify the I/O configuration data: All firmware data structures associated with channels in the affected domain have to be updated with the new HWA information.
5. Remove STI restriction: Enable outbound traffic by granting access to the new path to all processor units (PUs).
6. Re-enable interrupt processing: Inbound traffic that was delayed during the swap now causes interrupts on the new path. The interrupt handling on this path is enabled.

The following sections highlight only a few of the key aspects that had to be improved by firmware. A recovery-initiated swap task will follow the same design point but may deviate from the described sequence in some details that are beyond the scope of this paper.

Interrupt handling

One of the goals that was set for the RII design was to perform the TNT swap function without affecting the I/O traffic on any port not directly involved in the swap. Another goal was to avoid stopping any threads midstream, although this proved to be not entirely possible.

A typical I/O task consists of front-end processing (where the request is passed to the channel) and back-end processing (where the channel signals completion and sends the associated status back to the channel subsystem). The objective is to reach a point at which the back-end processing can be completed and new work can be prevented from starting. Instructions, on the other hand, continue to be executed unless they attempt to access hardware resources that are affected by the swap.

Without a quiesce of the interrupt processing, a change to the I/O path configuration during a running I/O operation could lead to dropping traffic, causing an unacceptable loss of application transparency. To avoid this scenario, it is important to update the I/O path configuration on each affected PU on an I/O task boundary. The PUs requiring updates to their I/O path configuration can be subdivided into two groups: system assist processors (SAPs), whose functions include I/O handling and managing system resources, and customer CPUs. Each PU category has different facilities available and requires a different strategy to reach an I/O task boundary. Upon a path configuration change request, a SAP self-issues a CSS event and can perform the update operation from the CSS interrupt handler (CSSIH). This is not available to CPUs, but similarly a CPU is issued a signal work (SIGW) interrupt and can safely handle the update in the SIGW interrupt handler. Since an SIGW interrupt is characteristically issued only to SAPs, a CPU that is issued an SIGW interrupt can be assured of a pending path configuration update.

To securely transfer I/O traffic from a source path to a destination path, we defined a two-stage operation requiring special synchronization among all affected PUs.

In the first stage (quiesce), traffic is disabled on the source path for each affected PU and is brought back up on the destination path only during the second stage (re-enable). Because re-enable brings traffic up on an empty path, only the quiesce requires an initiative on an I/O task boundary. Traffic cannot be safely moved until every PU signals it that it has successfully completed the quiesce. Should any PU in the quiesce process fail or time out, we defined a third operation, cancel, which attempts to back out of the procedure by re-enabling traffic on the source path.

STI access restriction

The STI restriction capability (STI port fencing) was first introduced on the z900 to restrict and release hardware accesses to a primary or secondary STI port for all PUs except for the PU that restricted the STI port. It was used primarily during a recovery action (STI link retiming). For z9–109, this STI restriction capability was extended to prevent hardware access during a controlled or uncontrolled TNT alternate path swap action. The design enables us to restrict or release both a primary STI link and a secondary STI link. To restrict and release the chained STI link between two STI-MP cards, all four secondary STIs of the affected TNT have to be restricted and released.

The semantics of an STI restriction is as follows: If an STI port has been restricted by a certain firmware function, all instructions that detect this restriction during a hardware access must wait and retranslate the associated target hardware address until the swap is complete and the restriction is removed.

Hardware swap

After the system is prepared for the actual swap, e.g., the interrupt processing is quiesced and the STI is restricted (meaning that the PU responsible for the swap is the only one granted access to the affected STI network), the hardware swap is performed. Because of time constraints, no differentiation between a controlled and a recovery swap is done there. The duration of the entire swap depends partially on the traffic in the PU–memory–MBA network. The routing and the access grant of the commands to the TNT can vary, and to avoid timeouts of the attached I/O, the path length is kept as short as possible. This results in a short swap sequence that can handle both types of the swap without distinguishing between them. In addition to the hardware accesses that are required to perform the swap, a few more actions must be taken. The TNT error reporting must be adapted, DMA traffic that has been initiated by the I/O and has already started must be observed, some configuration setup in the MBA and TNT chips must be moved, and finally the TNT is triggered to perform the hardware swap.

As shown in Figure 1, the general hardware swap sequence executed by firmware is the following (The TNT A primary eSTI is the active path, and this one gets switched to the chained link. All commands to TNT A are routed via MBA B and TNT B and no longer via MBA A because this path might already be down.):

1. The arbitration in TNT A is stopped. Request packets are no longer routed.
2. The error reporting of the TNT A logic up to MBA A is disabled. No errors are cleared. Only the reporting is disabled.
3. The TNT A secondary eSTI is set up as an eSTI slave. The original master setup for the default case would prevent a possible retiming of the chained link because no PU has access other than over the chained link.
4. Any other TNT setup dedicated to the active eSTI link, such as logical receive buffer timeout enablement, is moved from the primary to the secondary eSTI.
5. The I/O-specific setup in MBA A is copied to MBA B.
6. DMA responses already originated by the I/O before the swap was triggered are given a small time period to finish.
7. TNT A hardware is triggered to do the internal hardware swap and afterward to re-enable arbitration.
8. The error reporting of TNT A logic in the secondary eSTI is enabled. The reporting is now routed up to MBA B.

Coupling consideration

Coupling channel paths place unique requirements upon the RII function. First of all, the integrated cluster bus (ICB) links cannot implement the function at all. Unlike other channel paths that are connected through the bridge (TNT) logic, the ICB channel path logic resides in the MBA hub chip STI port logic. This is true both for the ICB-4 channel type, which directly uses the STI port on the hub chip, and for the ICB-3 channel type, which uses the bridge chip to create two channels from a single STI port on the hub.

Intersystem connect (ISC) channels also present unique challenges. Unlike the channels that are used for I/O, ISC channels use interrupt and error vector bits in the MBA port to carry additional information on top of the initiative. In effect, this distributes the channel function between the MBA hub chip and the channel cards that are attached via the bridge chips. In contrast to I/O

channels, where over-indication in the vector results in false initiative that can be safely processed, over-indication for ISC channels can result in data corruption. Loss of the information causes message timeouts. This means that the redundant I/O interconnect function must move information from the primary hub port to the alternate hub port when the alternate path is activated.

Because of these restrictive requirements, the traffic from the channels through the bridge to the hub is blocked before the switch occurs. This ensures that no updates are made to the vectors during the swap process. While accesses to the hub are blocked, the information is copied from the vectors in the first hub to the vectors in the alternate hub. After this is completed, the traffic from the channels to the hub is enabled once more. This copies all of the information from the old hub to the new hub without loss and without introducing extra information.

A switch to the alternate path as a result of a recovery action for the hub is also complicated by the presence of ISC and ICB channels. Because ICB channels cannot utilize the redundant I/O interconnection, a recovery action that instigates a switch to the alternate path causes all ICB channels on the affected hub to be stopped. ISC channels fare somewhat better, since they can perform the switch, but because they have functional information at the primary hub port, they cannot be switched transparently. The recovery action at the primary hub makes any information there inaccessible, so to preserve data integrity, all ISC channels on that hub must undergo a reset recovery action which forces channel-control check status to be posted for any active operation on the channel paths. This status is generally recoverable by the operating system software, so it does not affect customer operation, and it is certainly an improvement over the stop recovery that would be required without the redundant path.

Recovery considerations

Error recovery uses the data captured by hardware error detection to isolate the error source and to determine the effect of the error on operation. A comprehensive description of the RAS strategy for IBM S/390 systems can be found in [9]. Depending on the severity and extent of errors, the adequate recovery action is executed to remove the source of error. If an error is detected in the data flow of the I/O network (MBA, STI multiplexer), most of the time a simple bad status is returned to the originator of the operation, with no requirement for a special firmware-initiated hardware recovery action. For some more-severe errors, especially within control logic of the I/O network, a firmware-controlled error recovery is necessary to reset and initialize the hardware area in error or to retime and initialize the STI link. During this direct intervention into hardware, the functional use of this I/O path is blocked on both sides of the defective subunit for inbound and outbound operations. Hardware and firmware provide methods to block the functional work on the I/O path. Only error-recovery functions have exclusive access rights to resolve error situations within affected I/O subunits. Newly started I/O operations during this recovery task are handled like a temporary busy condition of the I/O path and are started again repeatedly until the error situation is removed and the I/O path is again ready. If an ongoing I/O operation is affected by the error, the affected function or superordinate hardware units at the respective endpoints of this path retry the operation or continue at a known synchronization point. When error recovery is not successful or the same failure reoccurs with high frequency, the failed unit is fenced from the configuration. On previous zSeries system generations, an entire I/O domain related to the fenced unit was stopped.

On System z9, error recovery now exploits the RII feature. All channel-attached devices are switched to the alternate I/O path and are not stopped when the unit in error cannot be reset. This can be done for all MBA errors and errors in TNT north ports. This error domain is the MBA common logic and the building blocks shown in light blue in Figure 1.

To guarantee a small switch-delay time, the chained STI link is initialized during system initialization and is set into standby mode. Idle patterns, which are continuously sent across the link by the hardware itself, are sufficient to constantly examine the operability of the link. In this state, the chained STI link is physically operational, and it is also fully supported by error recovery if errors are detected by hardware.

Application interface

On the SE, the hardware object model (HOM) [10] is the object-oriented representation of each sensed physical part in the System z9–109. The HOM supports the self-configuring and self-managing of hardware components and their status representation and management. The specification of the object hierarchy is rules-based, including attributes and interconnections to other system parts. The class instantiations representing the system components and abstractions consist of packaging units, functional units, and logical units. A packaging unit object is created for every sensed hardware FRU, i.e., the MBA fan-out, STI-MP, STI-A8, and STI-A4 cards. The functional unit objects reflect functional entities, which are active elements residing on packaging units. An example of a functional unit is the TNT module that resides on the STI-MP card. In addition to the clock states and function states, e.g., running or isolated (fenced), a functional unit maintains connections to other functional units. Application

interfaces and services have been established to provide access to the HOM for application programs such as repair and verify (R&V) and manufacturing engineering service (MES) hot plug.

The STI path between an MBA port and the TNT eSTI port is established via an STI cable. During the HOM self-configuration, i.e., during system power-on, these connections are sensed by the flexible support processor (FSP) [11]. All functional units in the I/O cages maintain an STI connection object. The STI connection holds two single STI connection objects, one for the default and one for the alternate path. The STI connection object has the information indicating which of the two is the active one. The two single STI connections for each functional unit are calculated on the basis of the sensed topology. The chained link is given by the STI-MP card plug positions. Depending on the results of the STI sensing, the default hardware address and the alternate hardware address are calculated and set in the corresponding STI connection object.

A controlled swap to the alternate path can be triggered by the SE during CBR, MBA fan-out card rebalance, MBA repair, STI-cable repair, or at the STI status and control panel. A recovery swap can be initiated by the CEC firmware in the event of an error in an MBA or STI path from an MBA port to a TNT eSTI port. If the cause for the recovery swap has been removed, that is, the failed unit has been replaced, it is of course desirable to swap back to the default configuration. The difference between the two scenarios is that in the controlled swap case, the SE requests the CEC to perform a swap on a specific link by means of a service word **(Figure 5)** and receives a service word indicating whether the swap was successful.

Because recovery swaps can occur asynchronously, the HOM STI manager splits off a separate thread that waits for such events. If the service word arrives, it will be parsed and checked if there are any differences from the information currently reflected in HOM. If so, the affected objects will be updated.

To perform controlled swaps and additional necessary tasks (for instance during a repair action), the HOM STI manager provides the following set of application interfaces:

- *Activate alternate path:* Activates the alternate path to a unit (and to all units within the same STI domain). The target is the unit ID.
- *Restore default path:* Activates the default path to a unit (and to all units within the same STI domain). The target is the unit ID.
- *Quiesce and disable link:* Stops all I/O traffic to the specified link (drops the link). The target is the unit ID along with an indication for default or alternate path.
- *Retime and enable link:* Re-enables the I/O traffic to the specified link (retimes the link). The target is the unit ID and an indication for default or alternate path.

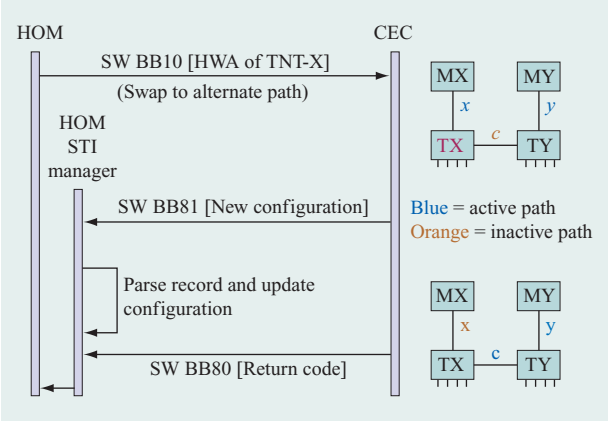

Figure 5

Service word communication during swap sequence.

To ensure that the SE and the CEC have the same view of the recent swap configuration in the event of a recovery swap that occurs while the SE is not available, the SE reboots and then explicitly requests the complete swap-related configuration information from the CEC firmware and updates the HOM configuration accordingly.

As stated earlier, the swap to the alternate path can occur under control or as a result of a recovery action. Therefore, a method of viewing the current state of the STI links is needed. This is the purpose of the STI status and control screen on the SE. The screen is designed to represent the logical layout of the STI links. Descriptions of the most important display items are shown in **Table 1**.

The STI status and control screen is used to view the current state of the STI links. Debug tools are used to manually switch between the default and alternate paths in a bring-up or test environment as an aid in hardware debugging. In a customer environment, repair and verify procedures are used to restore the default path.

Results

The RII feature was designed primarily to support path swap actions that are controlled by the SE. This enabled the implementation of advanced service features such as EBA. RII was also used to improve the automatic error recovery functionality that is implemented in CEC firmware. Today, after several months have passed since

Table 1 STI status and control screen display items.

Display item	*Description*
Select check box	This is how an STI row is selected.
Attention	This is a field to indicate that something is abnormal on this STI row.
STI link status	This shows the status of the STI link. Possible values are • Operational: Link is active and being used. • Standby: Link is operational but not being used. • Fenced: Link is not operational. • Stopped: Link is not operational, stopped by hardware.
Chain link status	This shows the status of the link between the two TNTs. Possible values are • Standby: Link is operational but not being used. • Operational: Link is active and being used.
Display PCHID/CSS.CHPID[1] button	This button displays the PCHIDs (and the CSS.CHPIDs that map to them) controlled by the selected STI. The display shows the PCHIDs/CSS.CHPIDs for both STI links.
Search PCHID/CSS.CHPID button	This button allows the user to enter a CSS.CHPIDs or PCHID value and search for the STI that controls them. The result is a panel that displays which row the controlling STI is on in the main display. This function can be used when the user is having problems with one of the PCHIDs/CSS.CHPIDs and wants to see the status of the STI that controls that channel.

[1] PCHID: physical channel ID; CSS.CHPID: channel subsystem set.channel path ID [6].

System z9 became generally available to the market, we can say that RII proved to work perfectly transparently to customer applications during controlled swap tasks. By *perfectly transparently*, we mean that the customer did not see any messages on the operator console or in any operating system running on the system. The benefit to error recovery is well above our expectations, but leaves some room for improvement on later systems. To cover the very few failure situations where a recovery swap is not possible will require a different approach to the RII concept. The RII feature cannot be used if a TNT chip is lost, because this chip provides the hardware for the RII feature.

The new feature does not create any performance impact while all I/O domains are running on their default paths. The logic that provides the alternate path is not "used" while traffic is flowing via the default path. The on-chip logic is parallel to the logic providing the default. No part of it can add path length to the default path. The chained connection is monitored by logic added to the TNT module that does not interfere with traffic on the other port. In a swapped configuration, peak performance is theoretically degraded because two I/O domains share hardware resources that are otherwise used by only one I/O domain. Test setups with fully populated I/O domains and heavy I/O traffic did not show any significant performance degradation when running in swapped mode. This is due to the bandwidth layout of the primary STI links, which include sufficient performance reserves to carry the additional traffic. If a given customer setup causes concern regarding the I/O bandwidth, it is possible to schedule the service slot to a time at which traffic is expected to be low.

Conclusion

By breaking with the former concept of full isolation between I/O domains, the RII feature provides a new method to increase the availability and serviceability of the System z9 server. While interacting in a complex and time-critical manner with multiple processors and I/O resources, a simple and easy-to-use primitive is provided to any high-level application exploiting this feature. The primitive is designed to be fast and transparent to I/O operations in progress. Exhaustive testing was done to ensure that there are no side effects, and these tests showed that there are none.

While IBM plans that future systems will keep I/O domains isolated from one another to make failure domains manageable, the RII feature has proven to be a valuable exception to a general rule. The plan is for the next system to change the handling of hardware interrupt locations and addressing tables in a way that removes much of the complexity that the current firmware implementations must handle. A key problem that had to be solved by RII is the synchronization among several processors that access an I/O path. The design for the next server generation will remove the requirement for such an interaction between different processors. This reduced complexity will further help integrate the swap task with all error-recovery scenarios in which it

theoretically can prevent the loss of resources. While RII supports our strategy to achieve and maintain RAS on zSeries servers as the best in the market, it has the potential to evolve further and provide an important building block for a variety of future service and recovery applications.

Acknowledgments

Design, implementation, and test of the RII feature required the effort of many workers in the Poughkeepsie, Endicott, and Boeblingen laboratories. We thank the many individuals (too numerous to list here) for their many contributions over the years.

*Trademark, service mark, or registered trademark of International Business Machines Corporation in the United States, other countries, or both.

**Trademark, service mark, or registered trademark of Teradyne Inc. in the United States, other countries, or both.

References

1. C. R. Conklin, C. J. Hollenback, C. Mayer, and A. Winter, "Reducing Planned Outages for Book Hardware Maintenance with Concurrent Book Replacement," *IBM J. Res. & Dev.* **51**, No. 1/2, 157–171 (2007, this issue).
2. J. M. Hoke, P. W. Bond, T. Lo, F. S. Pidala, and G. Steinbrueck, "Self-Timed Interface for S/390 I/O Subsystem Interconnection," *IBM J. Res. & Dev.* **43**, No. 5/6, 829–846 (1999).
3. J. Probst, B. D. Valentine, C. Axnix, and K. Kuehl, "Flexible Configuration and Concurrent Upgrade for the IBM eServer z900," *IBM J. Res. & Dev.* **46**, No. 4/5, 551–558 (2002).
4. L. Spainhower and T. A. Gregg, "IBM S/390 Parallel Enterprise Server G5 Fault Tolerance: A Historical Perspective," *IBM J. Res. & Dev.* **43**, No. 5/6, 863–873 (1999).
5. D. J. Stigliani, Jr., T. E. Bubb, D. F. Casper, J. H. Chin, S. G. Glassen, J. M. Hoke, V. A. Minassian, J. H. Quick, and C. H. Whitehead, "IBM eServer z900 I/O Subsystem," *IBM J. Res. & Dev.* **46**, No. 4/5, 421–445 (2002).
6. L. W. Wyman, H. M. Yudenfriend, J. S. Trotter, and K. J. Oakes, "Multiple-Logical-Channel Subsystems: Increasing zSeries I/O Scalability and Connectivity," *IBM J. Res. & Dev.* **48**, No. 3/4, 489–505 (2004).
7. T. A. Gregg, "S/390 CMOS Server I/O: The Continuing Evolution," *IBM J. Res. & Dev.* **41**, No. 4/5, 449–462 (1997).
8. E. W. Chencinski, M. J. Becht, T. E. Bubb, C. G. Burwick, J. Haess, M. M. Helms, J. M. Hoke, et al., "The Structure of Chips and Links Comprising the IBM eServer z990 I/O Subsystem," *IBM J. Res. & Dev.* **48**, No. 3/4, 449–459 (2004).
9. M. Mueller, L. C. Alves, W. Fischer, M. L. Fair, and I. Modi, "RAS Strategy for IBM S/390 G5 and G6," *IBM J. Res. & Dev.* **43**, No. 5/6, 875–888 (1999).
10. A. Bieswanger, F. Hardt, A. Kreissig, H. Osterndorf, G. Stark, and H. Weber, "Hardware Configuration Framework for the IBM eServer z900," *IBM J. Res. & Dev.* **46**, No. 4/5, 537–550 (2002).
11. F. Baitinger, H. Elfering, G. Kreissig, D. Metz, J. Saalmueller, and F. Scholz, "System Control Structure of the IBM eServer z900," *IBM J. Res. & Dev.* **46**, No. 4/5, 523–535 (2002).

Received March 10, 2006; accepted for publication June 13, 2006; Internet publication December 5, 2006

Ulrich Helmich *IBM Systems and Technology Group, IBM Deutschland Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (helmich@de.ibm.com).* Mr. Helmich received an M.S. degree in physics from the University of Sussex and a Dipl. Phys. degree from the University of Tuebingen, Germany. His current responsibilities are zSeries I/O microcode development, including first error data collection (FEDC) and error-recovery code.

Michael Becht *IBM Systems and Technology Group, IBM Deutschland Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (bechtm@de.ibm.com).* Mr. Becht received a Dipl.-Inform. degree from the University of Stuttgart, Germany. He is currently working on the hardware object model in zSeries firmware development.

Michael J. Becht *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (becht@us.ibm.com).* Mr. Becht is a Staff Engineer in IBM eServer I/O hardware development. He received his B.S. degree in electrical engineering from the University of Delaware. Mr. Becht is currently engaged in the development of next-generation I/O for zSeries supercomputers.

Janet R. Easton *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (jeaston@us.ibm.com).* Ms. Easton is a Senior Software Engineer in processor firmware design and development. She received a B.S. degree in computer science from the City University of New York. Ms. Easton is working on current and future design of the zSeries I/O subsystem.

Richard K. Errickson *IBMS Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (rerricks@us.ibm.com).* Mr. Errickson received a B.S. degree in computer engineering from Lehigh University. He is a Senior Engineer, working on the development of microcode for the I/O subsystem for IBM eServer processors.

Tobias Gehrmann *IBM Systems and Technology Group, IBM Deutschland Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (gehrmann@de.ibm.com).* Mr. Gehrmann received a Dipl.-Ing. degree in technical computer science from the University of Ravensburg-Weingarten, Germany. Mr. Gehrmann is the team leader of the reset and recovery team.

Steven G. Glassen *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (glassen@us.ibm.com).*

Seth R. Greenspan *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (sgreensp@us.ibm.com).* Mr. Greenspan received a B.S. degree in electrical engineering from the University of Connecticut. He is a Senior Engineer who is currently involved in I/O hardware development for the IBM Systems and Technology Group. Mr. Greenspan has received four IBM Outstanding Technical Achievement Awards.

Frank Koeble *IBM Systems and Technology Group, IBM Deutschland Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (koeble@de.ibm.com)*. Mr. Koeble received a Dipl. Ing. degree in computer science from the University of Applied Sciences, Aalen, Germany. He is responsible for the project management of the Boeblingen I/O Firmware Department.

Helge Lehmann *IBM Systems and Technology Group, IBM Deutschland Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (hhlehmann@de.ibm.com)*. Dr. Lehmann is an Advisory Engineer in hardware development. He studied mathematics and physics at the University of Cologne, where he received a Ph.D. degree for numerical solutions of partial differential equations. Dr. Lehman is working on future strategies in the I/O area for zSeries systems.

Carl Mayer *IBM Systems and Technology Group, IBM Deutschland Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (carl@de.ibm.com)*. Mr. Mayer received a Graduate Engineer degree in software engineering from the University of Applied Sciences, Esslingen, Germany. He is currently the team leader for the HOM and was the focal person for the System z9 hot-plug function.

Jonathan S. Nikfarjam *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (nikfarja@de.ibm.com)*. Mr. Nikfarjam received a B.S. degree from the Duke University Pratt School of Engineering, graduating *summa cum laude* with majors in mathematics, electrical engineering, and computer science. He is engaged in developing I/O firmware for IBM zSeries mainframes.

Forrest A. Schumacher *IBM Systems and Technology Group, 1701 North Street, Endicott, New York 13760*. Mr. Schumacher received a B.S. degree in mathematics from the State University of New York at Binghamton. He is an Advisory Engineer, working on the development of microcode for the hardware management console SE for IBM eServer processors.

Willi Storz *IBM Systems and Technology Group, IBM Deutschland Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (storz@de.ibm.com)*. Mr. Storz studied electrical engineering at the Fachhochschule Aalen, graduating in 1977. He is currently responsible for the firmware layer of error recovery for I/O adapters and I/O STI networks in zSeries systems.