

10G iSCSI Storage: Ready For Prime Time: An Introduction to the iStor iSCSI RAID Storage Processor (iRSP)

by *Carl Baldini*
Vice President of Engineering
iStor Networks, Inc.

Introduction

The arrival of 10G Ethernet at price points that would trigger the traditional adoption and ramp-up is finally here. Standards for copper twisted pair physical layers are ratified. First real implementations of 10G twisted pair PHYs have been tested. And production versions on technologies which don't melt the PCB appear to be around the corner. As shown in the past several technology upgrades of Ethernet, adoption should be rapid once the remaining hurdles are cleared.

As we've seen at each new generation of Ethernet speed, new areas of limitation were identified that were not issues at the previous rate. And 10G will be no different. Network controllers seem to have just reached the point where they can deal effectively with 1G traffic, balancing the performance of NICs, processors, memories, and storage interfaces. 10G will demand more than just incremental changes to the architectures of network engines as well and storage processing systems. This poses significant challenges to vendors vying to provide the next generation of storage controllers to exploit the benefit of 10G.

10G iSCSI Poses Significant New Challenges

Most current generation 1G iSCSI storage controllers rely on general purpose processors to handle iSCSI protocol processing tasks. In these architectures, one or more 1G Ethernet controller chips are interfaced to a high performance processor. The processor, usually a high clock rate, high power, general purpose processor handles all network protocol processing tasks, all the storage data movement, as well as the storage management.

As is well known in the industry, without some kind of protocol offload, TCP/IP protocol processing absorbs substantial processing cycles. Studies have shown that a good rule of thumb is that 1GHz of typical processing cycles are required to service a single 1G network connection. Current processors would struggle to handle a full-duplex 10G channel at even half of line rate. These kinds of solutions are simply not practical at 10G.

Most vendors will be forced to use some kind of network controller chip that provides protocol offload. These fall into two general categories – those that provide “stateless offload” and those providing “stateful offload”. Stateless offload NICs provide functions like checksum offload, and TCP segmentation offload and re-assembly. Although they do provide some benefits, they only reduce TCP processing burden by 15-25%. That still does not reduce processing overheads enough to support an effective 10G link. Stateful offload NICs move all the connection state information to the NIC which handles all connection management and termination. These controllers do a better job of offloading the processor, but suffer from other problems. They typically introduce significant latency in processing network protocols and often require additional off-chip memory for intermediate buffering to deal with retransmission, network flow control issues, and memory arbitration delays. Off-chip memories increase latency due to multiple memory accesses and transfer overheads. Some NICs also only handle protocols up through TCP, leaving the iSCSI layer to the processor.

Another problem with these multi-chip solutions is that they rarely act well as a complete system as portions of the solution may not be up to the task — and they become a bottleneck. For example, even if the network engine can support 10G operation, usable memory bandwidth often becomes the limiter. A high performance dual-redundant RAID controller that supports full duplex 10G rates will require a minimum of between 7 to 8 simultaneous memory data streams, each being 10G. This includes



full-duplex payload streams from the host network, full-duplex streams to the drive array, streams for RAID parity generation, and mirroring streams between peers for dual-redundant controllers. These are required in order to sustain 10G data flow end-to-end — i.e. all the way from the network interface to the drives. This requires a memory that can support over 70 Gb/s of sustained bandwidth. Most controllers cannot deliver this. Further, controllers that cannot support zero-copy models for data placement and management will require additional streams, taxing the memory structures even further. Many solutions also suffer from loosely integrated software and hardware components from multiple vendors. These components often are poorly optimized when integrated. Again, as a result, end-to-end performance of the complete controller suffers.

The bottom line is that general purpose processors with offload NICs bolted on, running loosely integrated firmware and applications, simply do not meet the performance requirements of an enterprise class storage controller. A better solution is needed.

iStor's 10G iSCSI Storage Processor

iStor Network's products are built around the iSCSI RAID Storage Processor (iRSP) ASIC, which provides the core data path and storage processing for iStor's integraStor™ storage systems. The iRSP was designed from inception to provide full duplex line-rate 10G iSCSI in real storage applications — including dual redundant systems.

Given the weak performance and poor scaling of iSCSI targets running on standard processors, iStor's development team came to the conclusion early on that to provide an effective 10G iSCSI target, specialized hardware was essential. A general purpose server processor, running a standard OS, and without significant hardware support, would not provide the performance necessary to handle protocol processing tasks as well as the multiple streams of data flow through a dual-redundant storage controller. iStor responded by defining a system-on-chip ASIC that combined, within one chip, a high performance low-latency network processing engine with a customized multi-processor storage engine, each with efficient access to a large shared memory (hereafter termed "buffer memory").

In order to provide a complete 10G storage controller on a chip, the solution would need to provide full-duplex line rate iSCSI protocol handling without the typical high-latency processing overheads or requirements for large intermediate buffering by the network engine. The device would need to provide an effective processing resource for handling the complexities of adaptive data caching, volume virtualization, flexible RAID algorithm handling, and could handle a variety of external drive/media interface standards. Further, the device would need to effectively handle an additional data stream being mirrored from a peer controller in a dual-redundant configuration — essentially another 10G payload stream into buffer memory. The solution is embodied in iStor's iSCSI RAID Storage Processor storage processor ASIC. This white

paper outlines some of the important aspects of that solution, and how the goals were achieved. A top level block diagram of the iRSP is shown in Figure 1.

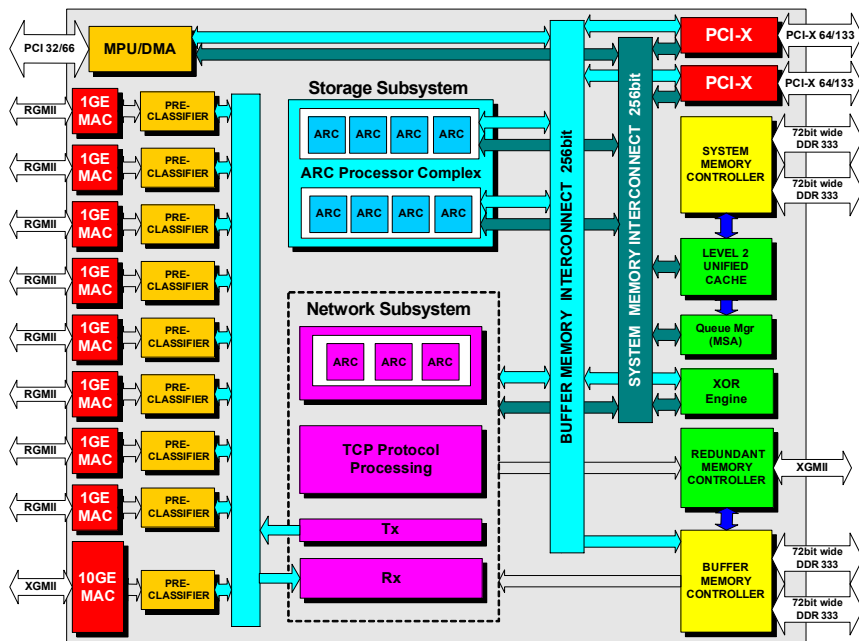


Figure 1. iRSP Block Diagram

TCP/IP Offload Alone Is Not Enough At 10G

Given the complexity of the iSCSI and TCP/IP protocol layers, and the large number of variables involved, handling the receive stream is a very complex task. One of the key targets for the iSCSI RAID Storage Processor's network engine were for iSCSI receive PDU payloads (data_out commands) to be transferred directly into their final locations in cache lines within buffer memory, ready to be managed



by the storage subsystem firmware, without additional movement or re-assembly and with very low latency. This was a very challenging goal. The goal is further complicated by the need to merge the byte-stream oriented network world and the block-oriented storage world. Data alignments and boundaries must be compatible for both the network side and the storage side. The storage side manages 512-byte blocks and 32K-byte cache lines. The network engine is stream oriented, and heavily layered, with many opportunities not only to break up streams into small frames, but to reorder frames or even drop them.

Although the iRSP would have embedded RISC processors within the same chip as the network engine, these processors would still not be fast enough to deal with the protocol handling requirements at 10G. The solution would require high speed dedicated hardware for protocol handling, implementing essentially a full stateful offload from the internal storage processing subsystem. iStor's hardware solution would need to address not just TCP/IP layers, but effectively

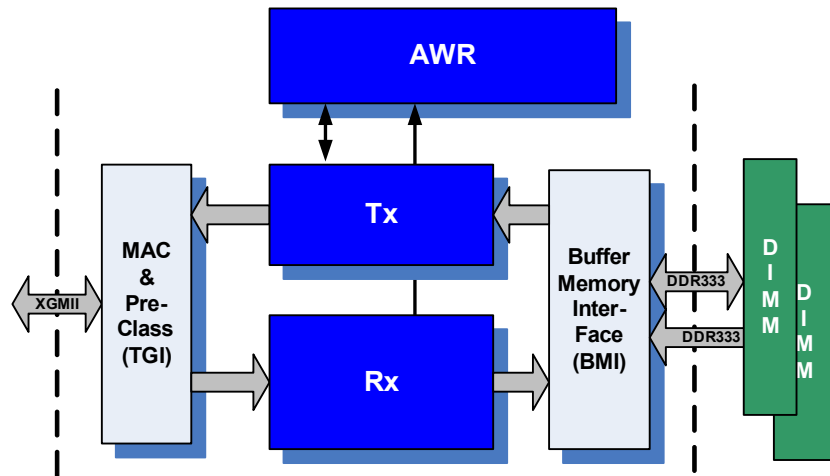


Figure 2. iRSP Network Engine

handle iSCSI layer processing as well. iSCSI PDUs would need to be decomposed, with SCSI CDBs queued up for SCSI firmware, with data payloads delivered into their final location for firmware management. This would be the key design criteria.

The iSCSI RAID Storage Processor Network Engine

The iSCSI RAID Storage Processor meets all these targets. The top level structure of the network engine is shown in Figure 2. It is divided into three major blocks: Receive path (Rx), AWR, and Transmit path (Tx).

Receive (Rx)

The Rx engine has the enormous job of dealing with the highly complex multi-layer protocol stream that arrives from the network. There are so many variations to the flow, it is beyond the scope of this document to cover. But, we can describe the most common case of receive traffic — an in-sequence iSCSI “data out” PDU. It’s instructive to follow an example of an iSCSI PDU reception all the way through the receive path. A greatly simplified description of the basic steps of the flow follows.

Step 1: Frames enter the ASIC and initially are handled by the TGI, which contains the 10G MAC and handles some pre-classification and formatting tasks. The incoming dword stream (two 32-bit words, or eight bytes) is collected into qwords (four words, or sixteen bytes). Refer to Figure 3.

Step 2: Pre-classified and formatted frames enter the Parser which parses and further classifies the data, storing it into buffers in the ingress FIFO. At frame end, the parser pushes a job entry describing the frame and how it should be handled into the job FIFO. This includes conducting a CAM search to determine the connection ID if it is an iSCSI packet.

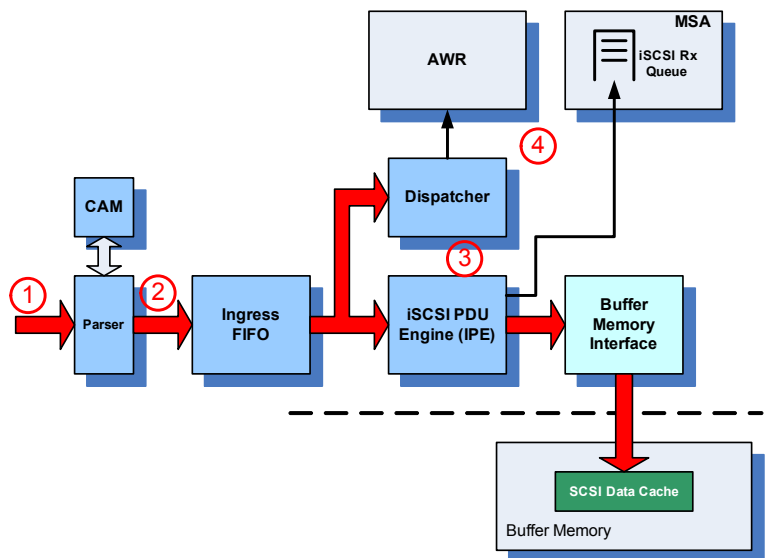


Figure 3. Receive Path Flow



Step 3: The Dispatcher dispatches frames. The payload portions of the frames are dispatched from the ingress FIFO to the iSCSI PDU Engine (or IPE). Recall that the TCP/IP layers break up iSCSI PDUs at arbitrary boundaries. So a TCP segment could hold an entire PDU, multiple PDUs, or just a portion of a PDU. And the segment could begin or end anywhere within a PDU, including partway through an iSCSI header, and on any byte boundary. The number of permutations here are huge, and the IPE needs to deal with all of them. To do so, it walks each TCP segment and determines PDU header and data segment boundaries. It decodes the iSCSI headers and performs special checks for command and SCSI data PDUs; and if necessary the remainder of a PDU may be discarded. When this process is completed, the iSCSI payload is written directly into the targeted data cache area in Buffer memory, as specified in the data pointer table.

Step 4: In addition to dispatching frames to the IPE for final processing, the Dispatcher sends the iSCSI header portion of the frame to the iSCSI Rx queue associated with the connection to which this data belongs. This notifies the iSCSI firmware, running on an ARC processor in the storage subsystem (discussed later in this document) of the PDU arrival and status. Also TCP flow management information is passed to the AWR via a hardware FIFO (discussed in the next section).

These steps describe the flow for the most common and high frequency cases of receive traffic – referred to as “fast path” traffic. In the case where frames are not in sequence, are fragmented, are not iSCSI frames, or have one or more of the many possible corner cases, they are handled by another mechanism within the network engine. That mechanism, called “slow path”, decomposes frames in a more stepwise fashion, handling all these corner cases, albeit more slowly. Frames may also be discarded if errors are detected, such as parity errors, or FIFO overflow conditions. In these cases, the actions already taken by the Rx must be carefully rewound so as to restore state and maintain proper housekeeping for each receive stream.

Because of the uniquely effective design of the Rx engine, it can handle complex iSCSI traffic at full 10G line rate — even for very small frames or difficult alignments. And this includes payload delivery directly into the storage cache in buffer memory, where it can be managed by the storage subsystem — with no additional protocol processing required. And this full line rate can be achieved for up to 1024 concurrent connections. Clearly, with this design, overhead for handling iSCSI PDUs is very low. This overcomes the industry debate over whether the iSCSI protocol has unacceptable latencies due to high protocol processing overheads. For controllers based on the iSCSI RAID Storage Processor, front-end latencies are comparable to other high performance interfaces like Fibre Channel.

AWR

The AWR block handles acknowledgement, windowing, and retransmission — hence the name AWR. It’s a control oriented block within the network subsystem focused mostly on handling the TCP layer in the protocol stack — flow control and error recovery. The TCP window and congestion management algorithms required for an effective iSCSI target are complex and compute intensive, and at 10G rates cannot be processed fast enough on general purpose processors — even the tuned ARC RISC processors embedded within the iRSP. Although the RFCs and algorithms are well known, the implementation can often require tuning and adaptation. Programmability was therefore a strong requirement for the ASIC design. To meet these requirements iStor developed a custom processing engine that can handle multiple parallel operations in a single cycle, but allowing for fine grain control. This engine is the heart of the AWR, and is called the Protocol Accelerator, or PA. The PA is a very long instruction word (VLIW) processor, which consists of four parallel arithmetic elements and a highly specialized instruction set, designed specifically to accelerate the TCP connection management algorithms. A block diagram of the AWR is shown in Figure 4.

In simple terms, AWR operations begin with messages arriving at the Master Scheduler from Rx (as described earlier), the event machine, and the pending queue (which holds jobs waiting to execute). The master scheduler determines the scheduling of VLIW routines that the PA is to execute. Before a VLIW routine begins executing, the VLIW register set, containing the local context for the connection, is automatically loaded by hardware from the TCPCB (TCP control block) cache for the specific connection. The TCPCB contains the complete detailed status of a connection. This rapid loading of TCP contexts supports



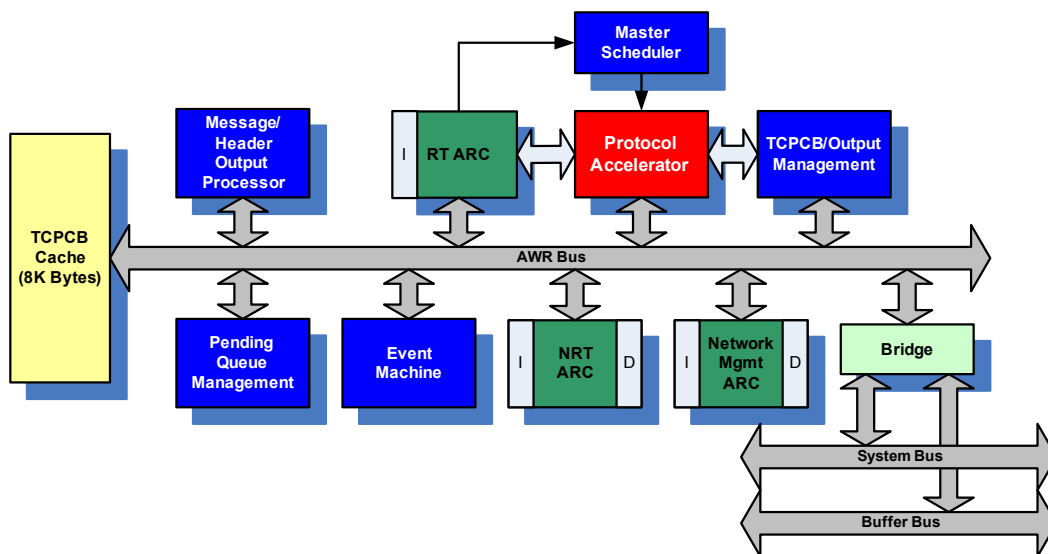


Figure 4. AWR

multiple simultaneous connections to be managed without the overheads of loading and storing the large connection contexts. A deeper discussion of these mechanisms is beyond the scope of this document – but suffice it to say, the PA is the basis for the high segment handling rates of iSCSI RAID Storage Processor. Because of the performance of the PA, the AWR can handle over 20 million frames per second.

Within the AWR block, there is an Event Machine and three dedicated ARC RISC processors. These processors are similar to the other ARC processors in the iRSP, but focused on a fixed function. Each has a different set of tasks. One ARC is responsible for working in real time with the PA. This ARC is called the Real Time ARC, or rtARC. It works closely with the PA engine, managing messages from the various network subsystem entities and sharing register sets with the PA. Two other ARCs handle a variety of other network functions, such as higher level network management tasks, such as network configuration and connection management, as well as some tasks for lower layers, such as IP fragment reassembly. Lastly, the Event Machine manages all the active timers required for TCP connection management, again freeing the ARC processors from these tasks.

All these components of AWR work together to efficiently handle TCP flow control and congestion management, as well as connection management duties. Standard sliding window management algorithms (RFCs 1122), including silly-window syndrome avoidance are implemented. AWR further implements congestion management algorithms, including slow start and fast retransmit capability (RFC 2001), and calculates round-trip-time (RTT) and retransmit timeout (RTO) based on Karn’s algorithm. AWR handles TCP re-sequencing, TCP options, keep alive mechanisms, as well as IP path MTU discovery, route establishment, ICMP, IP reassembly and options.

Transmit path (Tx)

Since we are in control of transmit traffic, it’s a simpler and more predictable process than receive traffic. The Tx block is responsible for handling two basic types of transmit traffic: 1) iSCSI Tx queue traffic, and 2) Raw Tx queue traffic. iSCSI Tx traffic requests come from the AWR and are associated with iSCSI PDUs, and there is a transmit queue per active connection. Raw Tx traffic requests are associated with non-iSCSI frames, and there are two queues – high and low priority. The Tx module is focused on achieving efficient flow of data from buffer memory through Tx in order to maintain full 10G line rate transmit.

The network subsystem also provides an extensive array of MAC and IP layer hardware statistics counters for reporting against various RFCs and a standard SNMP MIB.

The network subsystem of the iSCSI RAID Storage Processor represents a major advancement in the landscape of iSCSI target implementations, opening the door for multiple high performance and low latency 10G storage solutions. It’s a flexible and scalable architecture that will be the basis for several iterations of 10G network solutions.

Dual Controller Data Mirroring Requires A New Solution At 10G

High performance caching storage controllers will acknowledge the receipt of data from the host prior to writing it to the disks. This can expose the data, of which there would be only one copy, to



potential controller failures. Dual redundant controllers typically solve this problem by mirroring data to the peer controller which keeps a copy of exposed data until no longer needed. On these storage controllers, this mirroring operation is managed by the controller firmware, and usually requires specific data transfer operations to be set up and executed through some data channel between controllers which is often already busy with other data movement operations. For example, mirroring might be done at the back end of the controller through the fibre channel chips/interfaces used for drives. This has two disadvantages: 1) significant processing cycles are wasted managing these mirroring operations, creating additional overhead for each controller in a redundant pair, and 2) causes traffic conflict and congestion over the channels used for normal data operations. This can be true even if there are separate physical paths between the controllers, as the internal paths and drive interface parts are often burdened with handling the transfers. These issues usually result in controllers with poor performance scaling for the dual controller pairs, particularly with regards to write throughput.

iStor engineers recognized that this would be even more serious in a 10G product. At 10G receive rates, the burden on internal processors would be much more significant, and would seriously limit the end-to-end performance of a dual-redundant system. Early on, iStor established the objective of achieving effectively linear scaling when controllers are run as a dual-redundant pair. That is, a dual-redundant controller pair would achieve nearly 2x the throughput of a single controller. This would simply not be possible using conventional mirroring methods.

iStor again relied on its purpose-designed ASIC to address this system level problem. The iSCSI RAID Storage Processor incorporates a patented mechanism to handle data mirroring called Redundant Memory Channel, or RMC. RMC allows data to be mirrored between two iRSP based controllers without processor management or intervention, and without further utilizing main data movement paths within the controller. On integraStor storage systems, data mirroring is done over a dedicated full-duplex 10G data channel emanating from each ASIC, allowing controllers to mirror receive data at full 10G rates.

Inside the ASIC, RMC hardware automatically forwards network receive payloads being written into buffer memory (see earlier network engine discussion) to the peer controller over this RMC channel. See Figure 5.

The RMC channel uses a proprietary low-overhead link layer protocol on top of a standard IEEE 802.3ae MAC sublayer and 10 gigabit media independent interface (XGMII) PCS layer. The link protocol employs CRC and other protocol features to provide reliable delivery of mirrored frames to the peer controller. The standard XGMII interface at the ASIC pins can use off-the-shelf Ethernet PHYs for various physical layer interfaces, such as XAUI which is employed on the dual-redundant controllers in integraStor

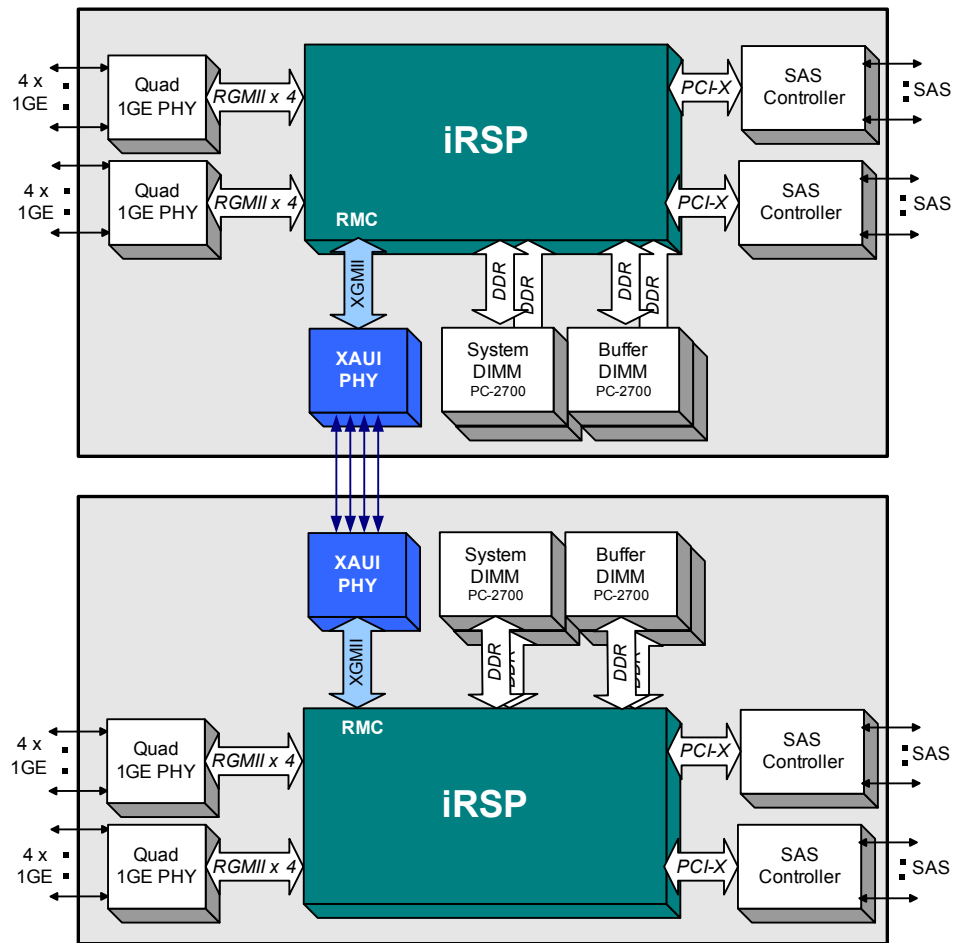


Figure 5. RMC Between Dual Controllers

storage systems for effective transfer over a backplane. Data transferred over this channel is placed into battery-backed buffer memory on the peer controller. Hardware mechanisms ensure that the data has safely arrived and has been written into buffer memory before acknowledging transfers to the originator.

The ASIC provides flexible means for setup of the RMC by providing a number of address ranges and criteria for qualifying data to be mirrored, allowing great firmware flexibility in using this channel for data, and even for some control communications. Because of the RMC, the overhead of managing data mirroring on processing resources is quite small, allowing processors to focus on doing their primary task of data flow and storage processing functions. Once configured for operation, as new receive data arrives it is automatically mirrored to the peer controller without disruption to the buses and resources used for normal data movement. This is one of the primary reasons why the dual-redundant integraStor solution can achieve nearly linear scaling of performance – unparalleled in the industry.

The Storage Processing Flow

A high powered network engine must be complemented by an equally high powered storage engine. The storage side must handle all storage related functionality, including caching, virtualization, and RAID operations. On the iSCSI RAID Storage Processor, this is handled by the Storage Subsystem.

The Storage Subsystem

This subsystem must handle a variety of storage related tasks including SCSI termination, adaptive storage cache management, volume virtualization, RAID algorithm implementation, and interfacing with external silicon for drive interfacing. In the iRSP, these functions are implemented primarily in firmware running on an array of ARC processors. Flow of firmware operations for typical iSCSI operations is notionally a pipeline as shown in Figure 6.

But in order to maintain 10G end-to-end performance, some stages in the pipeline require more performance than a single ARC processor can provide. In this case, two processors are employed, working in a symmetric multiprocessing model. This varying operation flow required a compute complex that could be effective at supporting both a pipelined compute model

as well as a multiprocessing model. This dictated that facilities for effective message passing and inter-processor communication, fast context switching, hardware

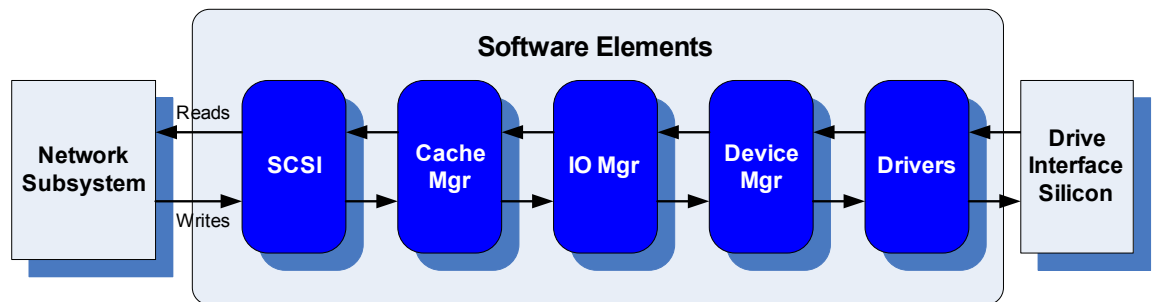


Figure 6. integraStor Software Pipeline

semaphores, and support for mechanisms to maintain cache coherency all be provided.

Inside the storage subsystem, there are 8 identical ARC RISC processors, implemented as two groups of 4 processors (called Lego4). These processors are based on ARC International's configurable ARctangent-A4 processor core, each running at 156MHz. They are highly optimized 32-bit Harvard architecture RISC processors with 4-stage pipelines and a register oriented instruction set that allows most instructions to execute in a single clock cycle. The ARC processors were initially configured to have a large working register set, and a barrel shifter. iStor further customized the processors by adding additional instructions, a scratchpad memory with a special DMA channel to buffer memory, along with separate data and instruction caches (see below). Several custom instructions were added, including divide, load/store multiple, hardware semaphore instructions, and instructions for usage of the special hardware queue accelerator (see MSA below). The block diagram of the Storage Subsystem is shown in Figure 7.

These processors would be ineffective without a cache/memory hierarchy that could move instructions and data at the required rates. To this end, the iSCSI RAID Storage Processor provides a very robust solution, including first and second level caches, high performance bus



structures, and a large dedicated system memory separate from the buffer memory.

Cache/Memory Hierarchy

To support a cluster of RISC processors with very low cycles-per-instruction, an effective first level cache is critical. The caches must be big enough to support the working set of code for that SW stage, but also be able to deal with some level of context switching. This dictated that each ARC processor have their own dedicated first level instruction and data caches. Although ARC international provided a baseline cache capability, these were insufficient to support the required fill rates and bus interfaces provided in the chip. So iStor developed its own first level caches. Each ARC is configured with separate 16K byte data and instruction caches. The caches are organized as 4-way set associative with 64-byte lines for the instruction caches, and 16-byte for the data caches. The caches use a wrap-around fill mechanism for optimizing utilization of the cache RAMs. For the data caches, a write-back replacement algorithm is used. The instruction caches can also evict lines (to the second level cache) to support an exclusion model (see below).

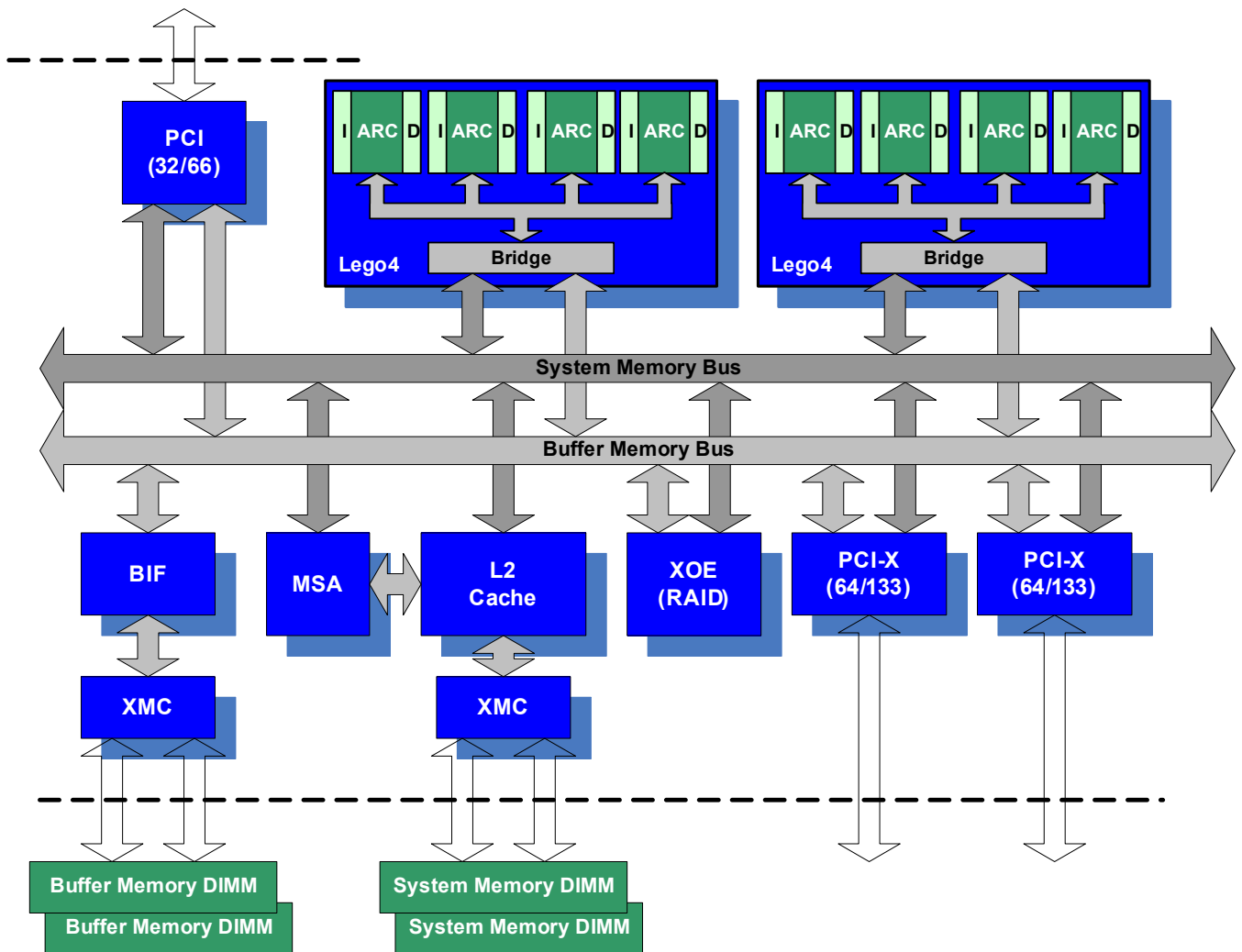


Figure 7. Storage Subsystem

A second level cache (L2C) is also provided. It structurally sits between the internal system memory bus and the System memory, and all system memory accesses go through the L2 cache. This is shown in Figure 8. This avoids the problem of cache coherency for IO transfers, and has the effect of speeding up other operations besides just ARC first level cache misses. The L2 cache is 128K bytes and is organized as 8-way set associative with 64-byte line sizes. The L2 Cache supports operations from



a single byte to sixty-four byte read/write/read-and-clear capability. It services L1 instruction and data cache misses from the ARC processors as well as non-L1-cacheable requests and DMAs. Given a first level cache line size of 16 bytes, one L1 transaction is a single cycle access to the L2 Cache block. The L2 cache employs a very flexible replacement mechanism that allows a variety of replacement policies to be configured. An efficient interface to memory also provides optimal scheduling and utilization of the DRAM ranks residing on the two system memory DIMM interfaces.

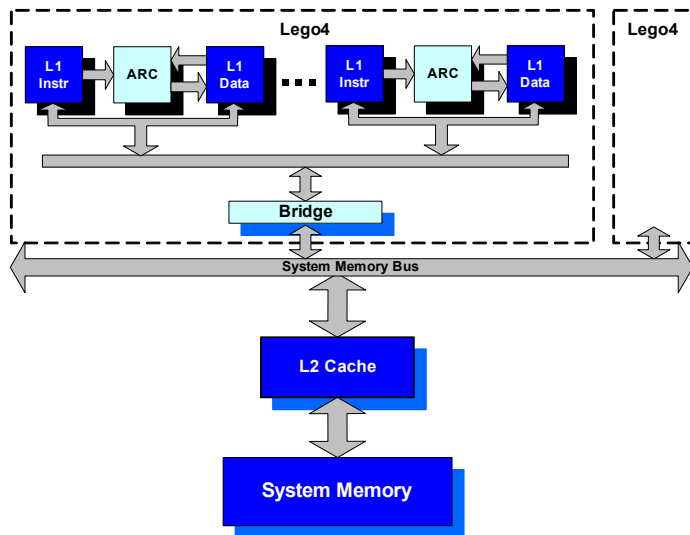


Figure 8. Cache Memory Hierarchy

regards to locality of reference. Some processors are running small code segments, and in these cases where locality is high, instruction cache hit rates approach 100%. For others which execute a more broad code base or which see some level of context switching, hit rates averaged from 95-98%. First level data caches correlated in similar ways, with hit rates averaging from 90-95%. The second level cache, being that it services processors, MSA, and other initiators, sees a much lower general locality. But hit rates were still very respectable at 85-93% on average depending on workload.

Like the Buffer memory described earlier, System memory supports two full 64-bit external memory interfaces, each with their own ECC. The system memory controller optimizes accesses to memory to maximize bandwidth to/from memory. The chip provides two full DDR DIMM interfaces off chip, expecting to see one DIMM on each interface. Each interface can support DIMMs up to 2GB in size, for a total of 4GB. For most applications though, system memory requirements do not exceed 256MB. The interface runs at 156.25/312.5 MHz, and so interface to DDR333 speed DRAMs — or standard PC2700 DIMMs. A single DIMM can be configured for smaller applications that require fewer network ports. The memory interface supports background scrubbing to further reduce the probability of double bit soft errors.

With 8 processors running at 156 MHz, and an effective memory hierarchy, the ARC RISC processors achieve a very low cycles-per-instruction (CPI) measure, thereby providing an aggregate of over 1000 MIPs of processing power in just the storage subsystem alone.

The Backbone — A2B Bus

Shared bus structures can have severe limitations if not carefully designed and applied. Available ASIC bus technology, such as the AMBA bus, simply cannot handle the high sustained bandwidth and utilization requirements in the iSCSI RAID Storage Processor. iStor required a much more robust solution for interconnect. To solve the problem, iStor licensed as a base the A2B bus from Advanced Architectures Inc, a provider of high end IP for ASIC development. The A2B bus is a fully synchronous and scalable design. It provides for split-transactions, simultaneous burst read and write transfers on separate read and write paths, with busy target arbitration to allow maximum utilization. The iStor implementation provides read and write paths that are each 128 bits wide, providing an aggregate burst bandwidth of 5 GBytes/second to both buffer and system memory simultaneously.

There are two A2B bus structures in the chip, one primarily for data path traffic, and on which the

The caches together also support an exclusive model for data and instruction allocation in the caches. When this mode is enabled, data or instructions from a particular address will only reside in one place within the cache hierarchy. In non-exclusive cache hierarchies, a data element can exist in multiple levels of the cache, potentially wasting the limited on-chip cache resource. The exclusivity algorithm required some enhancement to the replacement algorithms, and particularly for the first level instruction caches, which must now evict lines to L2 on replacement versus simply being overwritten.

During initial product testing, cache efficiency was measured with real workloads using the extensive performance counters implemented throughout the chip. It was found that cache hit rates varied per processor in the complex depending on which software component was being executed and its behavior with

Buffer Memory interfaces reside. The other is mostly associated with the storage subsystem of the part, and traffic in and out of the System Memory. These buses have shown to be very effective as the backbones of the ASIC.

RAID Algorithm Support

As should be clear by now, an important premise of the design was to keep payload data from flowing through the ARC processors and their cache hierarchy — allowing them to focus on effective system processing. In a storage controller that provides RAID protection, parity must be created against each byte of data. Running this data through the ARC processors would be cumbersome and would seriously degrade the effectiveness of the caches. To avoid this a dedicated RAID parity generation engine, called XOE, was added onto the buffer memory bus. It can generate parity for a variety of RAID algorithms, as well as concurrently generate CRCs to support disk lines (used for data and seek-error protection for low-end SATA drives). This engine can queue streams for multiple RAID extents, one at a time through the engine, moving accumulated parity and CRC results to either buffer or system memory. XOE leverages the high bandwidth read and write paths of the A2B bus for fast parity generation without heavy ARC processor intervention.

Flexible Drive Infrastructure Interface

In order to make the iSCSI RAID Storage Processor device applicable to a variety of drive interface types (e.g. SATA or SAS), the chip provides two full 64-bit PCI-X buses on which drive interface silicon resides. In initial system implementations, SATA controller chips reside on these buses, providing the direct interface to drives. Drivers running on ARC processors control these devices. The PCI-X buses run at 133 MHz, and support Dual Address Cycle operations. Internally, these buses can access both buffer and system memory, and can handle DMA operations to either. They provide a very flexible interface to backend drive infrastructures.

Effective Communication – MSA

As mentioned earlier, in order for the processors and network subsystem to work effectively together, an efficient means of communication is required. This was an important theme from the outset of the development, and in keeping with the theme of this solution, specialized hardware was developed to accelerate these crucial communications.

Much of the communication that happens in a complex system is by means of queues and stacks. They provide the basic constructs for efficient operation when implemented effectively. This is particularly true in a pipelined flow where operations are completed by an element or stage and are then queued at the next for execution. Rather than leave this purely to software, the ASIC hardware provides a memory structure accelerator (MSA) that provides high speed queue and stack constructs. But in addition the hardware provides a very powerful set of status and event signaling mechanisms. The MSA is a target on the A2B bus, and can be used by all initiators on the bus (both hardware and software driven). Up to 8096 simultaneous queues or stacks can be maintained by the MSA, and each can be up to 64K bytes in size. Queue entries can be small elements such as pointers, or even larger elements that contain data structures. Commands for these queues are presented on the A2B system bus, are processed by the MSA, and then are applied to the queues which are stored in system memory (and are usually cached in the L2 cache). Each queue or stack has a descriptor and descriptor extension which are located within system memory. These descriptors contain information about the structure's size, data location, thresholds, and signaling methods when significant events occur.

The MSA supports an extensive set of commands defining operations to queues and stacks, including Push, Pop, Peek, Increment, Decrement, Checkpoint, Rewind, as well as combinations of these. This set covers all the various operations that software or hardware within the chip requires. Further, when execution of a command generates a previously identified event, such as crossing a water mark, a “signal” may be generated. The “signal” can be either a hardware signal (see below) or a status pushed into another queue/stack (called Report to Queue) to allow for queue aggregation. An extensive set of event types are provided, including going empty, going not empty, underflow, overflow, crossing a high threshold, crossing a low threshold, and report-to-queue as mentioned earlier. Each event type is individually programmable per queue in order to provide maximum flexibility. Events are reported to various hardware components including the interrupt controller, which can distribute interrupts to any processor in the chip.



The commands and signaling mechanisms provided by the MSA offer a powerful way of detecting and reporting key inflection points to the software – offloading the processors from the task of managing these often complex structures and events themselves. And the MSA has ample bandwidth to handle the requests rates of all the initiators without becoming a bottleneck to the system.

Summary

Providing storage controllers than can take advantage of the emerging 10G Ethernet infrastructure will require more than just bolting 10G NICs onto existing 1G architectures. Solutions that absorb nearly all the controller's resources just to process the network protocol, or that have high overheads for data movement, will not deliver on the promise of 10G.

The iSCSI RAID Storage Processor has met the goals established at its inception, that of providing a robust 10G end-to-end storage solution. And because of its high level of physical integration, can be the basis for effective products spanning the range from SATA appliances for small companies up to high performance, highly available dual-controller rack-mount storage systems for mid-range enterprise applications.

The iRSP ASIC is shipping in volume today. It is the basis for iStor's integraStor family of storage systems, offering an un-paralleled hardware platform to leverage all the benefits of 10G storage now.

For more information about iStor Networks or its products, visit the website at: www.istor.com or send email to info@istor.com.

About iStor

iStor Networks, Inc. is an innovative technology leader that provides leading edge network storage solutions that are scalable, flexible, and high performing. iStor has professional storage solutions for small to medium business enterprise environments that are recognized for bringing leading edge iSCSI technology to the marketplace, setting new standards for cost effectiveness, world class performance, and ease of use. iSCSI IP-based Storage Area Networks (SAN) represent the fastest growing segment of the storage industry and iStor is uniquely positioned to provide the next generation of network storage technology to the marketplace.

The iStor integraStor family of products addresses customers' broad range of business-critical needs for flexibility, affordability, and high performance iSCSI Storage. The cost-effective iStor iSCSI storage includes entry level SATA appliances and scalable storage supporting SATA/SAS tiered storage for high availability primary storage applications.

integraStor technology is built on the foundation of a purpose-built ASIC designed for high performance and low energy consumption and complemented by a comprehensive storage management firmware stack featuring volume virtualization. Volume virtualization brings compelling features to the data center, including online capacity expansion, RAID level migration, free space defragmentation, read-only volumes, and ACL security on the volume levels, in addition to efficiently managing user data stored on the media.

Headquartered in Irvine, California, iStor has established a global presence with offices in the United States, UK, and operational facilities in Asia. The iStor team is dedicated to solving business storage needs today with award winning market leading iSCSI solutions.



iStor Networks, Inc.®

Corporate Headquarters

iStor Networks, Inc.
7595 Irvine Center Drive Suite 100
Irvine, CA 92618
Phone: (949) 753-8999
Toll Free: (888) 98iStor
(888) 984-7867
Fax: (949) 753-1068
Email: info@istor.com

Worldwide Sales Information
Phone: (949) 753-8999
Fax: (949) 753-1068
Email: sales@istor.com

Customer Support
Phone: (949) 753-8999 x159
Toll Free: (888) 98istor x159
(888) 984-7867 x159
Email: support@istor.com

Sales Management

Leo Kameya
Director of Sales - Americas
Office: (949) 753-8999 x181
Fax: (949) 753-1068
Mobile: (760) 707-6247
Corporate: (949) 753-8999 x181
Email: lkameya@istor.com

Peter Cmaylo
VP, NA & EMEA Sales & Business Development
Office: (508) 543-7966
Fax: (508) 543-7954
Mobile: (508) 208-8835
Corporate: (949) 753-8999 x105
Email: cmaylo@istor.com

International Offices

Europe

Tim Beck
Director of Sales EMEA
7 North Avenue
Goring By Sea
Worthing
BN12 4DA
United Kingdom
Office: +44 (0) 1903-502800
Fax: +44 (0) 5600-492410
Mobile: +44 (0) 7939-310522
Email: tbeck@istor.com

Asia Pacific

Dennis Lin
VP, Asia Sales and Business Development
7F., No. 46, Lane 10, Kee Hu Road, Nei Hu,
Taipei(114), Taiwan, R.O.C.
Office: +886-2-2799-9198 x8228
Mobile: +886-9-8771-9526
Fax: +886-2-2799-9398
Email: dlin@istor.com