CHAPTER 16

HIGH-SPEED ROUTER CHIP SET

The previous chapters provide a general discussion of design issues for the next generation of routers, including IP address lookup, packet classification, traffic management, and packet switching techniques. This chapter describes several commercially-available chips that are destined for next-generation router implementation. These chips include: (1) network processors for various packet processing; (2) co-processors for route lookup and packet classification; (3) traffic managers; and (4) switch fabrics.

16.1 NETWORK PROCESSORS (NPs)

16.1.1 Overview

The first generation routers were based on software with general-purpose processors (GPP) performing both the data-plane and control-plane functionalities. It was easy to add new features and additional interfaces through system and software upgrading. This architecture was sufficient for bandwidth need during that period and still persists in today's low-end products, for example, Cisco 2500 Series routers.

With exponential increase of Internet traffic, however, processor-centric general-purpose computing architecture was unable to satisfy today's bandwidth demand, and application specific integrated circuits (ASICs) were introduced to cope with the problem. This brought about the advent of the second-generation routers, where performance-critical data-plane functionalities such as packet-forwarding are separated from other functions and performed by dedicated hardware [1]. Because of especially high performance of ASICs and the stability of middle-layer network protocols, hardware-based architecture has been widely adopted in many vendors' designs, including the Juniper M- and T-series.

High Performance Switches and Routers, by H. Jonathan Chao and Bin Liu
Copyright © 2007 John Wiley & Sons, Inc.

In order to meet various quality of service (QoS) requirement, routers with the ability to control latency and jitter for packet flows and assure guaranteed priorities, in cases of congestion, are required. Networks are also having to accommodate complicated protocols and applications such as IPv6, VoIP, IPsec, VPN, and so on. Moreover, digital subscriber line (DSL) and packet over SONET (POS) are added to legacy ATM (asynchronous transfer mode), IP, and Frame Relay. In order to meet these new transmission and switching demands, carriers needing a powerful and flexible solution offering a short time-to-market and long time-in-market solution, have turned to a new breed of packet processor – the network processor.

In spite of no official definition, a network processor is a programmable processor specific for network applications. As being the two architectures dominating the first two generations of routers, ASICs provide better performance but have higher costs, longer development cycles and less flexibility. GPPs, on the other hand, require too many cycles to perform the same task and thus cannot provide wire-speed processing. The focus of today's router vendors is to seek proper ways of combining the flexibility of GPPs with the performance of ASICs. Most mainstream vendors' products can be categorized as application specific instruction processors (ASIPs), that is, a programmable processor with an instruction set optimized for a particular application domain [2]. The network processing forum (NPF) was founded in 2001, aimed at speeding-up the standardizing process. A series of common specifications have been established since then, ranging from hardware interface, and application programming interface to benchmarks [3].

16.1.2 Design Issues for Network Processors

Functional Requirements. The basic design methodology follows the "functional profiling to architecture exploration" procedure [4]. Network processors typically sit between the physical devices and the switch fabric along with the subsidiary static random access memory (SRAM), dynamic random access memory (DRAM), and in some cases, content addressable memory (CAM) and external co-processors, as shown in Figure 16.1. Network processors are dedicated to packet processing. Some network processors have to be able to

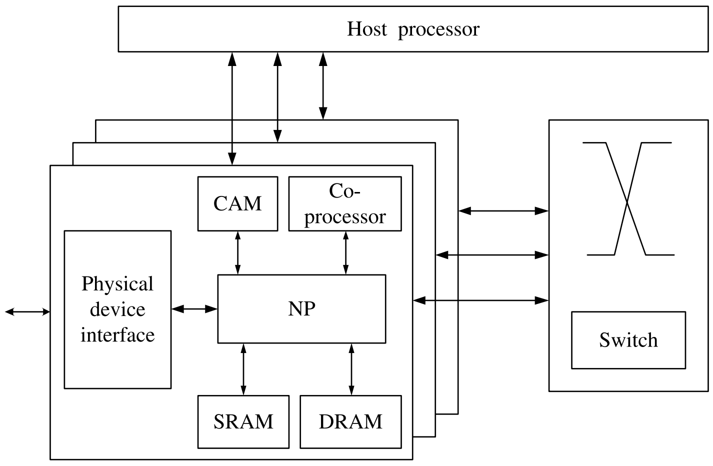


Figure 16.1 Network processor in a router application.

do general purpose processing with the absence of a host processor. Their tasks typically include:

Media Access Control. Many network processors can handle the raw data stream coming from the physical links. Low-layer protocols like Ethernet framing and ATM cell processing will be realized directly in this kind of network processor.

Packet Processing. Both header processing (such as field extraction) and payload processing (string searching and encryption), fall into this category.

Packet Classification. Packet classification is the foundation for functions such as QoS, policing and traffic management. For example, a service provider may provide a tiered service where mission-critical traffic like E-commerce has a guaranteed delay of <80 ms, while all the other traffic is delivered with best effort. To enforce this policy, traffic must be identified and classified according to the QoS requirement, as it enters the service provider's network. Network processors can be used to perform this function.

Policing and Traffic Management. Some vendors provide policing and traffic management functions in their network processor products. These operations are rather crucial to core routers and are related to QoS implementation.

Routers that are located at different locations on the Internet have different requirements of network processors. Core routers or backbone routers emphasize high-speed and high-throughput and should provide an advanced traffic management ability but do not need to offer feature-rich applications. Access routers only need limited traffic management capabilities, but huge processing power for the various applications running on them is desirable. The requirement for edge routers lies somewhere between the previous two. Furthermore, core routers typically have much larger routing tables than the other two. Therefore, it is obvious that this kind of router must be able to handle large amounts of looking-up operations with low latency. Interfaces needed by different kinds of routers are also different. An access router may be equipped with just a number of physical layer interfaces of a single type. On the other hand, the other two kinds of routers need to support many of the prevailing transmission formats, namely POS, Frame Relay, ATM, high-speed Ethernet, and so on [5, 6].

Design Considerations for NPs

Architecture Selection. Some obvious, but important, characteristics of network processing are the simplicity and the weak dependence of floating point operations, even in the higher-layer functionalities. Therefore, most NPs adopt the reduced instruction set computing (RISC)-like fixed-point architecture. Kernel operations such as pattern matching, table lookup, and queue management comprise the majority of the functionalities that affect the architecture selection in many aspects [4].

Instruction Set. Although the core frequency of today's processors has reached 3 GHz, for most vendors, instruction set optimization is still needed since it can dramatically reduce the number of cycles required for the same task [7]. These tasks include: data transfer operations, bit-manipulations, finite state machines needed by protocols and services, and the routing/switching operations [8]. Some test benches aiming to facilitate this optimization are also devised to further exploit the program architecture [9]. Yet the optimized instruction set is a double-edged sword. Since many vendors have developed their own instruction set, corresponding software development becomes a key-point of success in the market with a lack of third-party support. If the original equipment manufacturers (OEMs) cannot obtain enough programming tools, there would be a long way for network processors to go from merely a hardware platform to a running engine in the router.

Hardware Function Unit. Special functional units are often provided for common networking operations that are usually cumbersome and error-prone to implement in software, yet easy to implement in hardware, such as cyclic redundancy check (CRC) and checksum operations. One type of these hardware function units is used for simple operations. The units produce one result within one pipeline stage of a processing element and typically cannot be shared among multiple processing elements.

Another extensively used form is a co-processor. As opposed to special functional units, co-processors often perform more complicated tasks and can be accessed by multiple processing elements. To achieve an actual increase in performance, co-processors must have the ability to work with other processing units simultaneously and collaboratively. They can be triggered by special instructions and can offload some complex functions from main processing units. Some network processor products have external co-processors, typically packet classification co-processors, security co-processors, and traffic managers [2].

Parallelism. Packet processing inherently has a large amount of data parallelism that can be exploited. Parallelism exists at both inter-packet and intra-packet levels. If we consider "packet flow", most processing is performed on a per packet basis with each packet going through similar but independent processing. This phenomenon is the main aspect of inter-packet parallelism. Moreover, within the many tasks to be performed on each packet, some of them are also independent of each other. That is what we call intra-packet parallelism. If we assign individual tasks for different packets into multiple threads, both of these two classes of parallelism can be viewed as thread-level parallelism (TLP). Instruction-level parallelism (ILP) refers to the phenomenon that part of instructions within one program can be executed simultaneously and still produce the right result. This is more fine-grained and more difficult to exploit, typically involving compiler cooperation and a multiple-issue mechanism for instructions [10]. While some control-plane processors still use traditional single-threaded processors, most mainstream NP manufacturers have developed various multi-threaded schemes making use of different levels of parallelism [11].

Buffer and Memory. Network processor design focuses on providing sufficient processing power to accomplish complex functionalities in addition to the elementary packet forwarding. However, even if processing capability was sufficient, the processor would not perform well without adequate memory bandwidth.

Under today's store-process-forward scheme, storage and buffer size dominates the efforts devoted to NP design. Packets might be buffered inside an NP waiting for processing elements to become available or might be stored inside or outside an NP before and after switching. A lookup table is also stored in memory on-chip or off-chip. What makes the design difficult is the tremendous memory bandwidth needed by both the packet processing and the lookup operation. In general, a packet will go through the memory interface up to four times in the ingress direction if deep packet processing is performed and involves being: (1) copied to proper buffers when received from the incoming port; (2) read out (header and some portions of the payload) for the purpose of classification, route lookup and/or modification; (3) copied back with the modified information; and (4) read out again for transmission. In some architectures the first two data transfers can be omitted with the help of internal buffers. It still leads to enormous memory bandwidth demands [12]. Given a 60 percent efficiency of DRAM utilization and an available double data rate synchronous dynamic random access memory (DDR SDRAM) speed of 266 MHz, several hundreds of pins are needed to satisfy the OC-48 line rate for the memory alone. One possible solution is on-chip, ultra-wide DRAM technology [13]. Some chip designers have followed this approach, for example, EZchip.

Route-Lookup and Packet Classification Implementation. Quite a few vendors have implemented lookup and classification operations in separate chips functioning as co-processors or as a component of a chip-set. These are the fundamental functions of both the basic forwarding-decision making and high-level services such as QoS and IP packet filtering. Most developers employ software solutions and some of them provide interfaces to ternary content addressable memory (TCAM) devices as an option. Researchers have made great efforts in the related fields, ranging from look-up table management to operation complexity, and from algorithms to implementations [12]. Manufacturers tend to use their own proprietary algorithms and solutions. In contrast to the complex software schemes or dedicated hardware search engines, TCAMs are easy to use and have the advantage of deterministic searching delay and high performance derived from the parallel and pipelined operation of TCAM devices. Its major weaknesses include high cost, large power consumption, and limited look-up width, which severely constrains its applications in long-item and/or high-volume classification tables, for example, the look-up table of the future IPv6 protocol with IP addresses of 128 bits [14].

Software Support. The prevalence of network processors heavily depends on third-party software and support to provide an easy-to-program environment. This in turn reduces time-to-market and increases time-in-market. The process is more difficult than that of other communication products because of the use of nonstandard application–optimized instruction set and the parallel architecture of network processors.

16.1.3 Architecture of Network Processors

Two major architectures of today's NP products are as follows: (1) "dedicated approach", where most of the necessary network processing (e.g., packet classification, header checking, and so on) is performed by specialized hardware or components with a highly optimized instruction set; (2) "brute force approach", where a multitude of RISC-based simplified processing elements (PEs) are integrated onto a single chip to exploit the parallelism of packet processing with only marginally specialized hardware support.

A typical architecture of a brute-force duplex/simplex network processor is shown in Figure 16.2. Let us examine the ingress packet flow, that is, port to switch direction. Packets are received by the physical layer (PHY) device interface and transmitted by the switch fabric interface. Before being processed by the processing elements array or being scheduled for transmission, they are buffered under the control of the buffer and memory management module. The two interface modules (PHY device interface and switch fabric interface) can also do some bit-manipulation operations for the implementation of layer 2 or upper layer functions such as frame header processing or CRC.

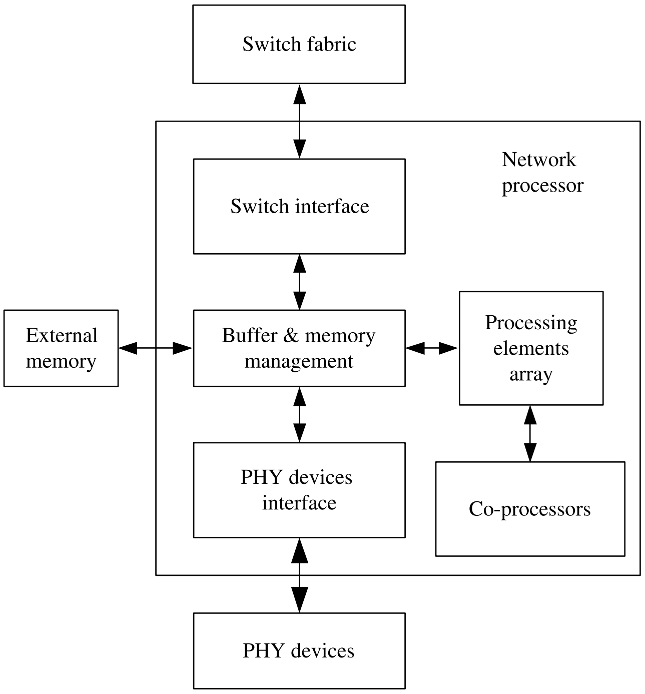


Figure 16.2 Simple block diagram of a brute-force network processor.

In order to exploit the massive parallelism available in packet processing, especially the TLP, the processing unit may be constituted by a number of PEs with an optimized instruction set for network processing and a pipelined or even a super-pipelined architecture. Some implementations utilize very long instruction words (VLIW) or superscalar multiple-issue techniques to further exploit the ILP [15]. But the issue width of the instruction is often constrained to just two because of the limited ILP existing in integer non-scientific applications, that is, at most two instructions can be read out from instruction code memory and prepared for execution. Manufacturers like Intel and IBM use a network processor with a general-purpose processor core embedded in it, which is responsible for system initialization and part of packet processing workload, whereas others such as AMCC and Motorola just provide an external host central processing unit (CPU) interface.

16.1.4 Examples of Network Processors – Dedicated Approach

IBM NP4GS3. The basic architecture is similar to that of Figure 16.2 but most network processing related functions are implemented in dedicated programmable coprocessors. NP4GS3 is a 2.5 Gbps duplex network processor. Its network interface can be configured either as Ethernet medium access control (MAC) or POS. The Ethernet mode can support up to four gigabit Ethernet or 40 fast Ethernet ports while the POS mode can support 16 OC-3c, four OC-12, four OC-12c, one OC-48, or one OC-48c POS framers. It has almost a symmetric architecture for the two packet flow directions that share the same embedded processor complex, as shown in Figure 16.3.

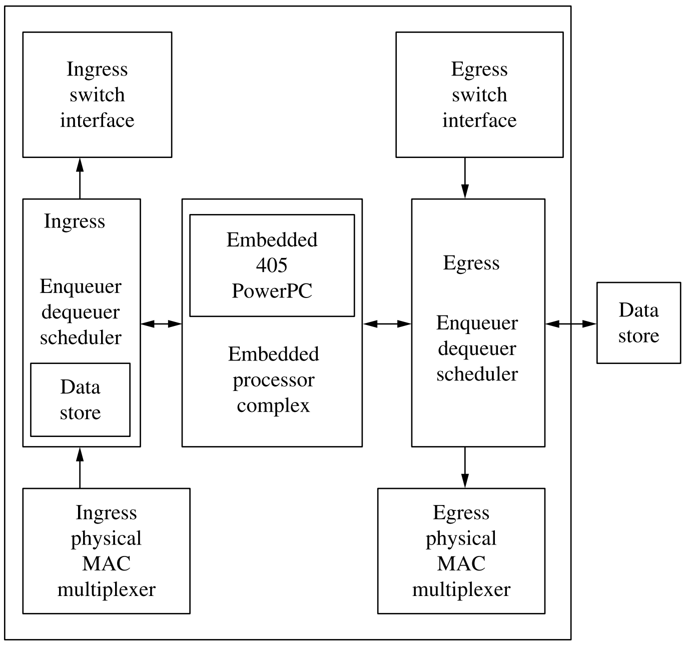

Ingress switch interface
Egress switch interface
Ingress
Enqueuer dequeuer scheduler
Data store
Embedded 405 PowerPC
Embedded processor complex
Egress
Enqueuer dequeuer scheduler
Data store
Ingress physical MAC multiplexer
Egress physical MAC multiplexer

Figure 16.3 Simple function block of NP4GS3.

The physical MAC multiplexer moves the data between the physical layer devices and network processor, and keeps 36 Ethernet statistic counters per MAC, enabling support for many standard management information bases (MIBs) at wire speed. Ingress enqueuer/dequeuer/scheduler (EDS) buffers the incoming frames in the data store and extracts the frame header information for the embedded processor complex (EPC) to process. It also enqueues the frames to await transmission or discards them according to the processing results returned from the EPC with the help of hardware configured flow control mechanisms. Egress EDS is mainly responsible for frame reassembly and external buffer handling. The NP4GS3 provides two data-aligned synchronous link (DASL) ports at the switch side, which can also be used to cascade several NP4GS3s.

The NP4GS3 incorporates an embedded PowerPC subsystem that contains a 133 MHz PPC405 processor core with 16 KB of instruction cache and 16 KB of data cache. A peripheral component interconnect (PCI) bus is included to connect the NP4GS3 to an external host processor. EPC uses interrupts to communicate with PowerPC, appointing a particular DRAM location as a mailbox.

The processing core of the EPC consists of eight dyadic protocol processor units (DPPUs). Each DPPU incorporates two core language processors (CLPs), 10 shared co-processors, two communication buses and one shared memory pool, as shown in Figure 16.4. CLPs execute the so-called picocode that is optimized for network applications. Each CLP can support two threads and the two threads can switchover with zero overhead. Although 32 independent threads can exist simultaneously, at any one time, only 16 of them are executed for frame forwarding, route table building, and network processor maintaining because each CLP has only one set of execution unit. Some hardware accelerators are shared among the DPPUs, facilitating the operations of thread allocation, packet protocol

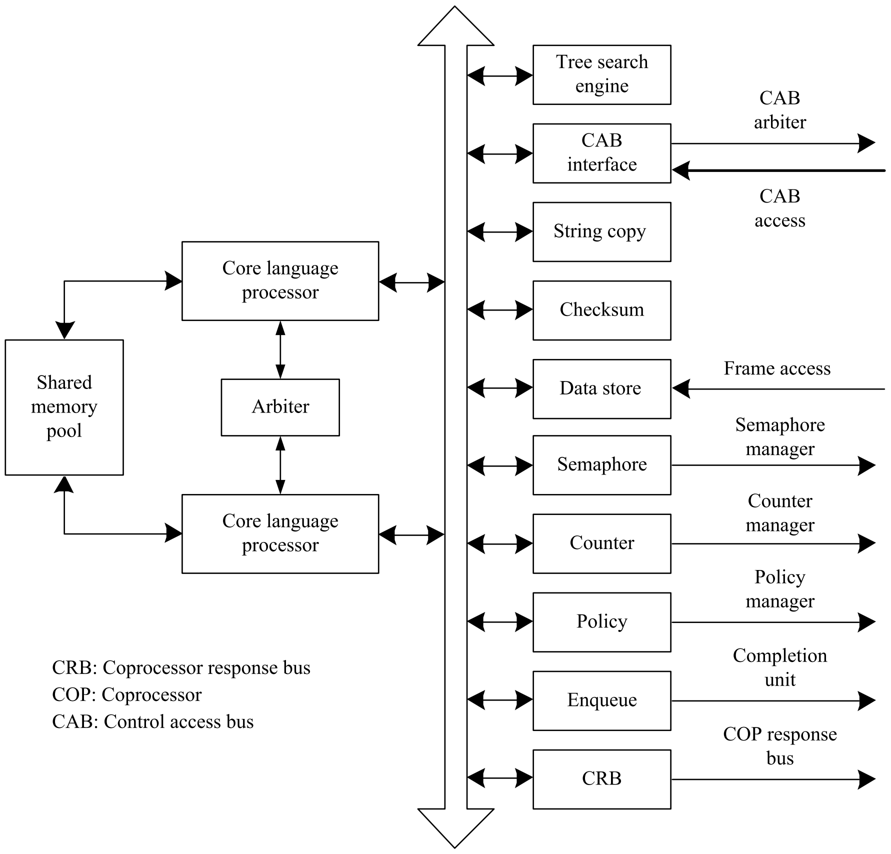


Figure 16.4 Function block of DPPU.

recognition, and packet order preservation. Of the ten co-processors, some act as the commonly seen checksum computing unit, string copier and tree search engine while the others provide interfaces to shared hardware assists. They include a policy manager that is used to determine the character of the incoming data stream, a counter manager that can contribute to the statistics, flow control and policy management, and a semaphore manager that assures the mutual exclusion access to a shared memory pool. These co-processors are called by a special kind of picocode and can work in parallel with each other and the CLPs.

The tree search engine (TSE) co-processor performs table search with the table maintained as Patricia trees. TSE supports three search modes: full match (FM) for layer 2 ARP address lookup, longest prefix match (LPM) for layer 3 IP address routing, and software managed table (SMT). Hardware supported geometric hash functions can yield lower collision rates than conventional bit-scrambling methods [16].

Brute-Force Approach. Among the diversity of brute-force network processors, the main architectural difference lies in the multi-PE organization, that is, how multiple processing elements are organized into an array. Two basic organizations are parallel mode and pipeline mode. The former is widely adopted by most NP products while the latter often appears in some start-ups' design. Different variations can be made upon these two basic modes.

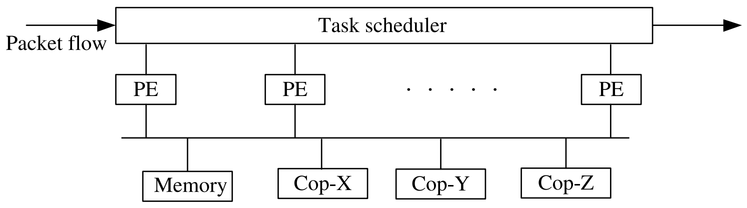


Figure 16.5 Parallel mode of PE organization.

Example of Parallel Mode. As is shown in Figure 16.5, a task scheduler is needed to assign incoming packets to available PEs. Processing elements of this type are typically not coupled very tightly and are usually identical. Interconnections among them and communication methods between them vary from different solutions. What functionalities each PE will perform are totally determined by the programmer.

Multi-threading is typically supported by each PE to exploit more TLP; thus, if one thread has been suspended waiting for the result of a co-processor, another one can get into the processing element and be executed. To take full advantage of this mechanism, hardware-facilitated thread-switching should be employed to hide latency.

Although this arrangement efficiently exploits both the inter-packet and intra-packet parallelism, when a programmer manually partitions the tasks among processing elements, extra burden is added to load balancing and this increases inter-PE communication overhead. So some programmers still stick to the "run to completion" programming model or just do very simple task partitioning.

Multi-threading also introduces new problems. Since the processing element is not dedicated to one packet, a new packet will be placed into a processing element when the suspended packet awaits its result to be returned from the co-processor. Thus deterministic packet processing performance cannot be determined and the packet order within the same flow may be distorted, which is usually unacceptable in a network node. This can be avoided by allocating packets of the same flow into the same thread. This method, however, will decrease the utilization when there are no available packets from other flows for a PE with suspended threads and new packets of the same flow keep coming in.

AMCC nP7510. AMCC nP7510 OC-192c network processor is capable of providing layer 2 and layer 3 cell/packet processing at wire speed for multi-service POS, ATM, and gigabit Ethernet systems. A simple functional block diagram is shown in Figure 16.6.

Two packet input/output (I/O) interface blocks connect nP7510 to the framer or switch fabric. In addition to the interface-specific functions, they also provide layer 2 functions such as AAL5 CRC and length checking, header sequence error checking, byte and cell length generation, OAM CRC checking and generation. The packet transform engine provides data paths and buffering between the framer interfaces. It performs special frame commands issued by the nPcores. These functions include add or delete information from a packet as it leaves the channel, or to create a new packet. The nP7510 contains six nPcores, which are microcontrollers with a network-optimized instruction set called network instruction set computing (NISC). Each nPcore can handle 24 tasks and up to 144 packets can be processed simultaneously. Incoming packets are allocated to the next available task in one of the microcontrollers, which is transparent to the microcode programmer. The nPcores also control the policy engine, which is a packet classification module differentiating among

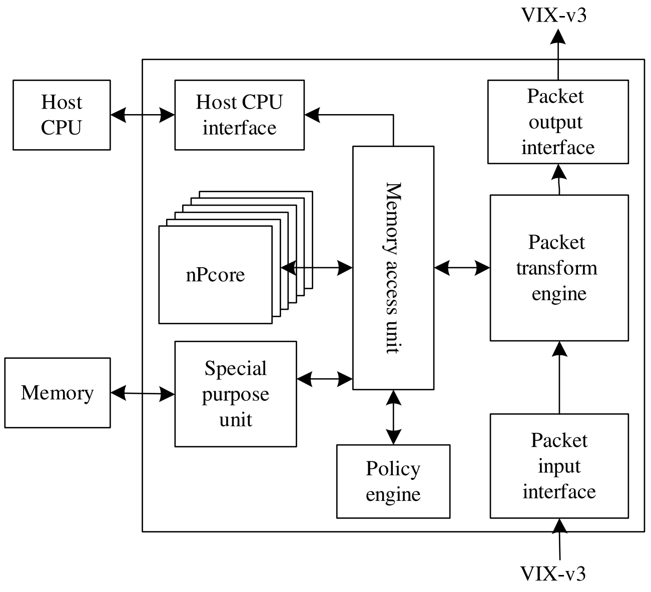


Figure 16.6 Block diagram of AMCC nP7510.

different kinds of packets/cells. It contains a 512 × 68-bit ternary policy database with a configurable number of entries, and entry width.

Example of Pipeline Mode. As is shown in Figure 16.7, in pipelined mode the processing procedure is divided into multiple stages with each stage designed to handle a certain category of tasks. Once a packet enters such a "PE-pipeline," it has a dedicated data path to go from one PE stage to another without rewinding or skipping. The functionality of each PE is also affected by the hardware designer. Since PEs are allocated to different tasks in advance, each one can be optimized for dedicated tasks, resulting in different kinds of PEs in a single processor. For example, each kind of PE can have its own optimized instruction set. Software should cooperate with an underlying hardware platform to make the partitioned tasks balanced among processing elements. Otherwise the over-loaded stage will become the performance bottleneck. Such a requirement makes software modification a difficult job.

Agere PayloadPlus. The PayloadPlus network processor family includes the fast pattern processor (FPP), the routing switch processor (RSP), and the Agere system interface

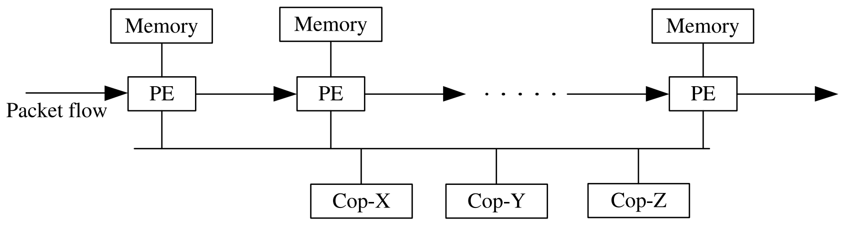


Figure 16.7 Pipeline mode of PE organization.

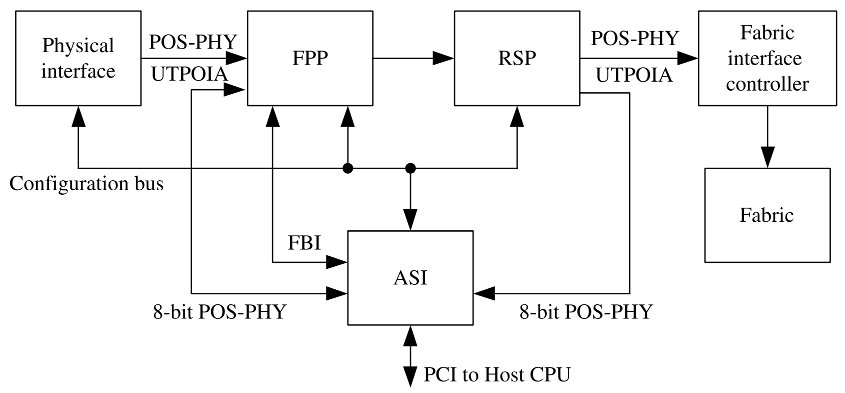


Figure 16.8 Line card structure using Agere PayloadPlus.

(ASI). The programmable FPP performs wire-speed classification and analysis for multiple protocols at layer 2 through 7. The RSP performs various queuing, traffic management, traffic shaping, and packet modification functions on traffic flows in a fully programmable way. The ASI provides a PCI interface to a host processor for control and management functions, including routing table and virtual circuit updates, hardware configuration, and exception handling. It also assists the FPP in policing both cell and packet-based traffic, maintaining state information on data flows and capturing statistics. Figure 16.8 shows a line card system using Agere PayloadPlus. The FPP and RSP, together with Physical Interface and Switch Fabric, form a pipelined-based, communications-focused architecture.

The internal structure of FPP is shown in Figure 16.9. The *Input framer* receives packet/cell data from PHY devices or management information from the ASI. The data

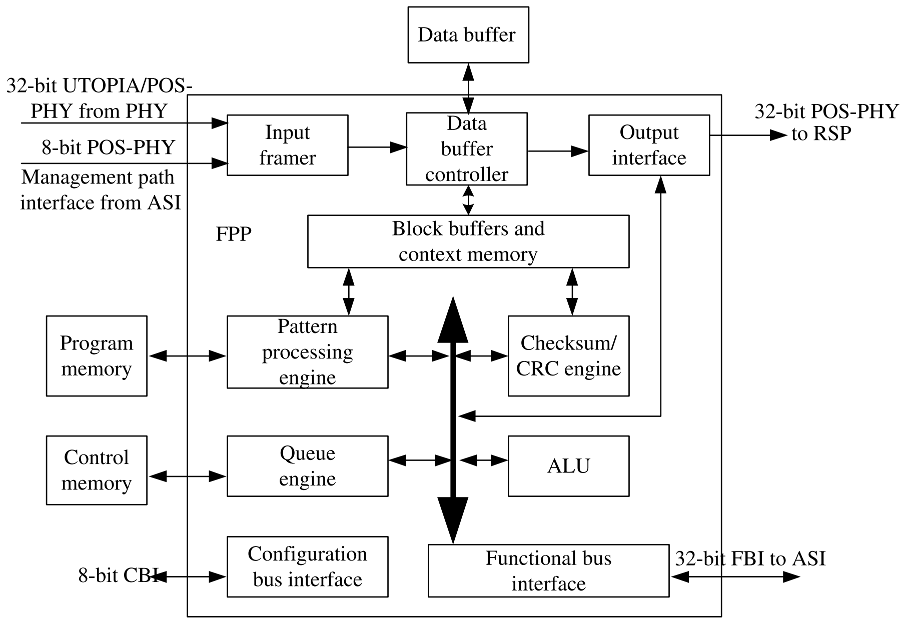


Figure 16.9 Agere FPP internal architecture.

buffer controller manages movement of data blocks between the data buffer, input framer, block buffers, and output interface. The output interface transmits Protocol Data Units (PDUs) and their classification results to the RSP or other downstream logic. The block buffers and context memory stores the blocks currently being processed, and the configuration information for the context. The pattern-processing engine (PPE) is programmed with a simple protocol processing language, the functional programming language (FPL), to recognize and classify incoming packets based on a set of patterns. The checksum/CRC engine performs a generic checksum, IPv4 checksum, CRC-10, or CRC-32, as requested by the FPL program. The arithmetic logic unit (ALU) performs arithmetic functions such as addition, subtraction, shifting, and Boolean operations. The two bus interfaces, configuration bus interface and functional bus interface, are responsible for communication between the FPP and the ASI. The FPP divides the processing into two passes: the first pass stores data as 64-byte blocks and computes data offset for each block, creating a linked-list data structure that defines the reassembled PDU; the second pass processes the whole PDU. It simultaneously performs pattern matching and transmits the reassembled PDU and the results of the pattern-matching process for that PDU.

The RSP accepts PDUs and their classification information on up to 128 logical input ports. It queues them on up to 64 K programmable queues, then outputs the modified PDUs on up to 256 logical output ports. The RSP uses programmable VLIW compute-engines to process PDUs. The internal architecture of the RSP is shown in Figure 16.10. The input interface sends the incoming PDU blocks to the PDU assembler for reassembling into PDUs, and sends the FPP classification results for PDUs to the transmit queue logic. The stream editor compute engine performs PDU modification for each queue. These modifications include encapsulating PDUs into AAL5, segmenting into ATM cells with appropriate headers, and implementing IP operations (e.g., decrementing time-to-live counts and updating checksums). The traffic manager compute-engine enforces discard policies and keeps the queue statistics. The traffic shaper compute-engine ensures QoS and class of service (CoS) for each queue [17]. The output interface accepts blocks from the stream editor and sends them to the appropriate port manager and output port.

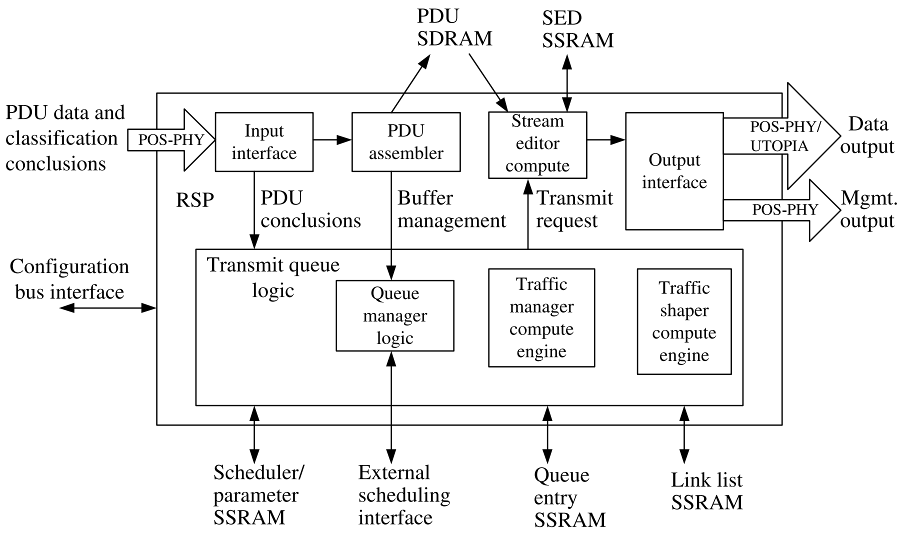


Figure 16.10 RSP internal architecture.

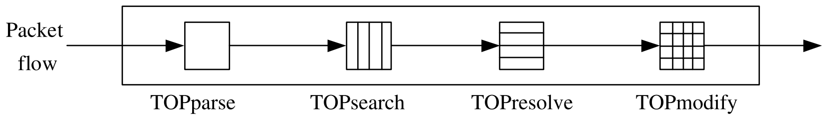


Figure 16.11 Data path for each packet-processing task in EZchip NP-1.

EZchip NP-1 and NP-2. NP-1 is the first duplex 10-Gbit NPU introduced to the market. As shown in Figure 16.11, NP-1 incorporates a four-stage packet processing procedure. This is as follows: (1) parse: extract different packet headers and do the corresponding analysis; (2) search: perform table look-ups through layer 2 to layer 7; (3) resolve: assign the packet to the proper port according to the destination and QoS requirements; (4) modify: modify certain fields of the packet on the basis of classification result.

As illustrated in Figure 16.12, each stage is implemented by a number of superscalar and superpipelined task optimized processing cores (TOPcores) with their own customized instruction set and data path for the specific packet processing operation. The memory bandwidth problem is resolved by a rather complicated embedded-DRAM system-on-chip technique. On-chip DRAMs, together with the dedicated search algorithms, replaces the expensive CAM. All of these efforts have dramatically improved the single-chip performance and reduced the system chip-count, power consumption, and cost [7].

NP-2 is EZchip's newer generation family of network processors that builds on EZchip's NP-1 family and further its integration. NP-2 integrates a 10-Gbit or 5-Gbit full-duplex NPU, classification engines, traffic managers, 1-Gbit and 10-Gbit MACs, and SPI4.2 interfaces in a single chip. The integration allows system vendors to deliver solutions that are cost effective as well as power and board-space efficient.

Example of Hybrid Architecture. Two variations of the basic modes are shown in Figures 16.13 and 16.14, respectively. Such structures can be designed to be more adaptable to varying application requirements. But they typically make the programming task more challenging.

Intel IXP2800. The Intel IXP2800 network processor [18] is a member of Intel's second-generation network processor family. It delivers 10 Gbps packet forwarding and traffic management on a single chip.

Figure 16.15 shows the internal architecture of IXP2800. The Intel XScale^{TM} core is a general-purpose 32-bit RISC processor used to initialize and manage the network processor, to handle exceptions, and to perform slow path processing and other control plane tasks.

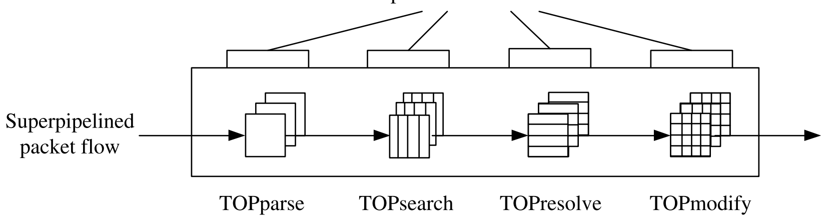


Figure 16.12 Superpipeline and superscalar organization of processor-array for massive processing power in NP-1.

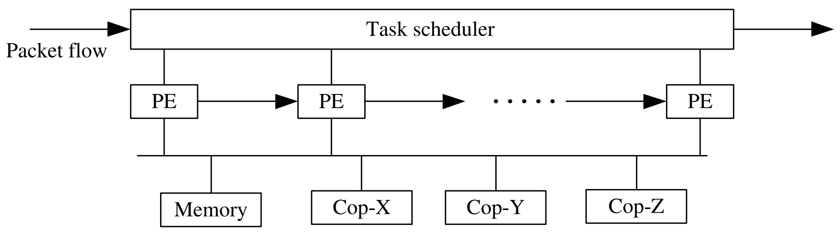


Figure 16.13 Parallel mode with pipeline capability.

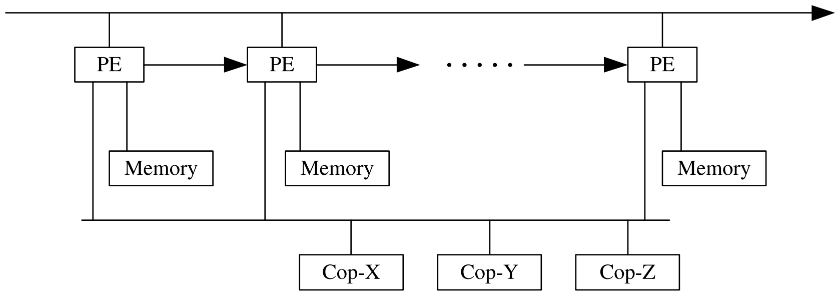


Figure 16.14 Pipeline mode with parallel capability.

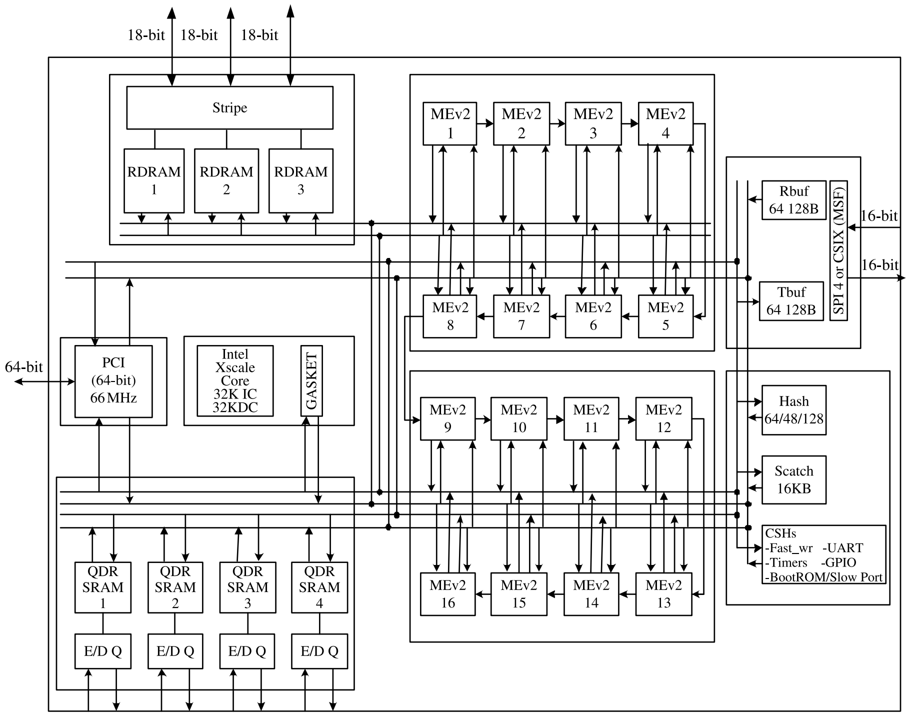


Figure 16.15 IXP2800 block diagram.

The media and switch fabric (MSF) interface is used to connect the IXP2800 network processor to a PHY device and/or to a switch fabric. Each of the receive and transmit interfaces can be separately configured for either SPI-4 phase 2 (System Packet Interface) for PHY devices or CSIX-L1 protocol for switch fabric interfaces. The receive and transmit ports are unidirectional and independent of each other. Each port has a clock, a control signal, a parity signal, and 16 data signals. Rbuf is a RAM that holds received data. It stores received data in sub-blocks, and is accessed by Microengines or the Intel XScale^{TM} core reading the received information. Tbuf is a RAM that holds data and status to be transmitted. All elements within a Tbuf partition are transmitted in order.

The IXP2800 network processor has controllers for three Rambus DRAM (RDRAM) channels, supporting up to 2 GB of DRAM. It also has four independent SRAM controllers, with each supporting pipelined quad data rate (QDR) synchronous static RAM (SSRAM) and/or a coprocessor that adheres to QDR signaling. A 64-bit PCI bus is used either to connect to a host processor or to attach PCI-compliant peripheral devices.

There are 16 microengines (MEs) used for most of the programmable per-packet processing. They are connected into two clusters of eight MEs. They have access to all shared resources (SRAM, DRAM, MSF, etc.) as well as private connections between adjacent MEs. The ME provides support for software controlled multi-threaded operation. Given the disparity between processor cycle times and external memory times, a single thread of execution will often block, waiting for memory operations to complete. Having multiple threads allows for threads to interleave operation – there is often at least one thread ready to run while others are blocked. In addition to various kinds of local memories, the MEs also provide some special hardware blocks to assist certain packet processing tasks such as CRC, pseudo random number generation, multiplication, and so on.

The scratchpad unit provides 16 KB of on-chip SRAM memory that can be used for general-purpose operations by the Intel XScale^{TM} core and the MEs. IXP2800 also provides 16 hardware rings that can be used for communication between MEs and the core. The hash unit provides a polynomial hash accelerator. The Intel XScale^{TM} core and MEs can use it to offload hash calculations in applications such as ATM virtual channel/virtual path (VC/VP) lookup and IP 5-tuple classification.

Cisco Toaster2. The Toaster2 network processor is an example of a parallel, pipelined multiprocessor system, shown in Figure 16.16. The processor is a 4×4 matrix of custom-designed PEs. The PEs are laid out to form four parallel data flow pipelines. Each pipeline has four stages and corresponds to a row in the PE matrix. The Toaster PE is a VLIW core with independent data cache, 12 KB instruction RAM (IRAM), and two four-stage instruction pipelines. The instruction set is optimized for network processing, including fast lookups, atomic memory operations, preset tasks, and bit-level operations. Each instruction is 64 bits in length and is segmented, with each part being tied to a pipeline.

The PEs in each column share a hierarchical memory system. Local memory of each PE is in the form of a register file, a 64-byte data cache, and space for context (Toaster unit of data for packet processing) storage. Large data structures are stored in the column memory devices. The on-chip (internal) column memory (ICM) is typically a 16 KB SRAM device. The off-chip (external) column memory (XCM) is typically a 32 MB SDRAM device [19]. The packet interface consists of two subblocks called the input header buffer (IHB), which is used to receive Toaster contexts, and the output header buffer (OHB), which is used to transmit contexts. The route processor interface (RPI) provides access to all resources

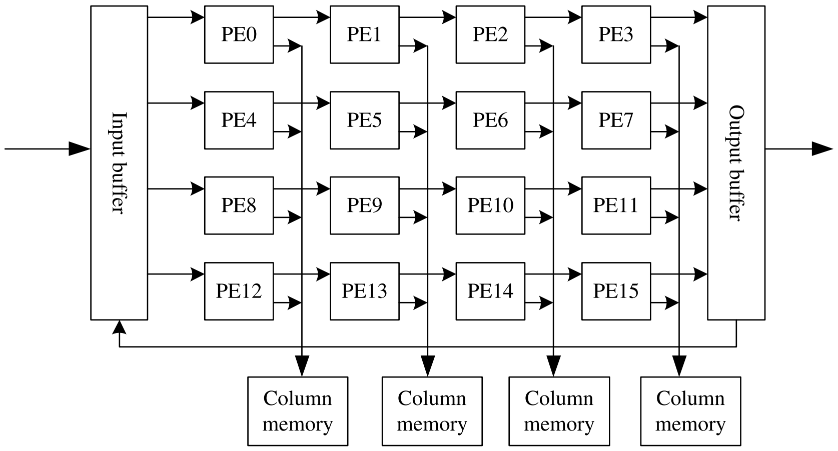


Figure 16.16 Architecture of Cisco's Toaster2 network processor.

internal to the Toaster2 as well as the XCM. In addition to resource access, the RPI block also maintains a set of synchronized timers that are accessed via individual processors.

Toaster2 has been designed to support either centralized or distributed configurations. An example of a system of centralized configuration is shown in Figure 16.17. Packets are received or transmitted through media-specific interfaces. Valid packets are stored in a packet buffer while the corresponding context is sent to Toaster2 for processing. A context includes the first *n* bytes of each packet, and a packet descriptor carrying the necessary information about the packet, such as the length of the received packet and a handle that will be used to associate the forwarding result with the body of the packet stored in the packet buffer. A forwarding task can be divided into eight software pipeline stages. Each packet is processed by one of the four pipelines. After being processed, the context is transferred to

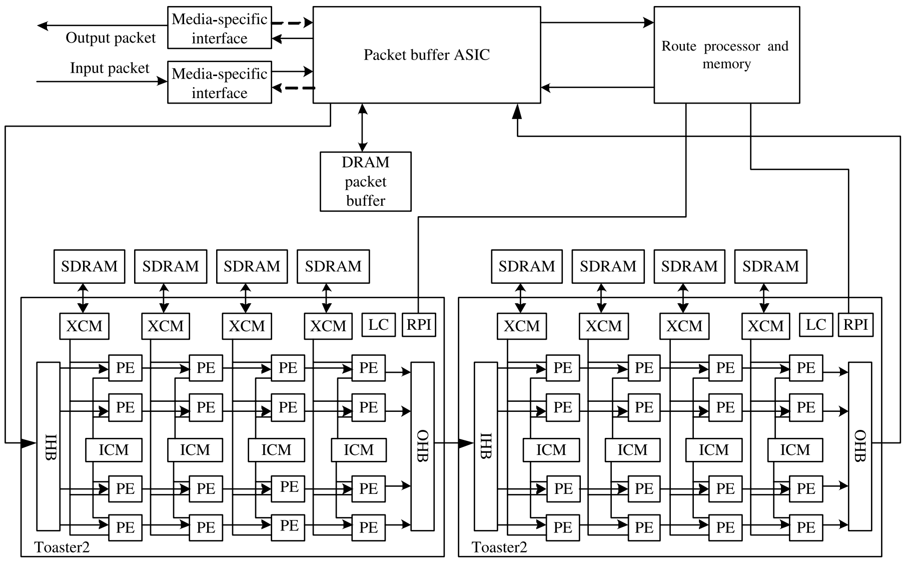


Figure 16.17 Example of a system using Toaster2 in centralized configuration.

the packet buffer ASIC and merged with the packet body, waiting for transmission under the control of the traffic manager [19].

Juniper's Internet Processor II. The architecture of Juniper router is based on a "Switch fabric–Internet Processor II–JUNOS software" hierarchy. Switch fabric is dedicated to performance-eager and stable functions whereas JUNOS software provides full flexibility for routing protocol and network management. The Internet Processor II, derived from its former Internet Processor ASIC, sits between the fixed-function ASIC and the general-purpose microprocessor. It can support packet classification, packet filtering, policing, traffic shaping, statistics, and accounting without significant penalties on the forwarding rate.

Functions of Internet Processor II are broken down into "primitives". The following three primitives are provided as part of Internet Processor II: tree lookup, table lookup, and filter program. The order and basic structure of the execution of the three primitives are under of the control of the JUNOS software. New releases of the software will bring newer features to the Internet Processor II [20].

16.2 CO-PROCESSORS FOR PACKET CLASSIFICATION

The classification processing is an important functionality and generally comprises the bottleneck of the network processing unit (NPU) performance. Typically, co-processors are provided to offload this function. These devices can be divided into two categories, that is, TCAM-based and algorithm-based. This section first describes an interface standard for a co-processor recommended by the network processing forum (NPF), that is, LA-1 bus (look aside interface), followed by the introduction of several typical chips of the two types of co-processors.

16.2.1 LA-1 Bus

The NPF hardware working group has two task groups to standardize the interfaces for NPs:

- A streaming interface for complete packet data path.
- A look-aside interface that does not need to be in the direct data path.

As can be seen in Figure 16.18, NPU and co-processors may operate in either look-aside (LA) or streaming mode. The LA interface is intended for devices located adjacent to a NPU that offloads certain tasks from the NPU. In this architecture, the co-processor sits beside the NPU outside of the datapath, enabling packet co-processing in parallel with the NPU operation, increasing the overall throughput. It may not receive the full payload of all packets passing through the NPU, but can make the design more complex and add latency to the system. In the streaming interface architecture, the datapath includes both the NPU and the co-processor. Every packet passes through the co-processor, so it can act on packets as required and pass them directly to the NPU. This architecture requires the co-processor to continuously process the packets at wire speed, or it may become overwhelmed and drop packets.

The LA-1 interface is targeted to support the lookup requirements for OC-48 through OC-192 line rates. It is currently defined to have a bandwidth of about 6.4 Gbps per direction, which is sufficient for lookups at 10 Gbps (OC-192) packet rate. The packet count

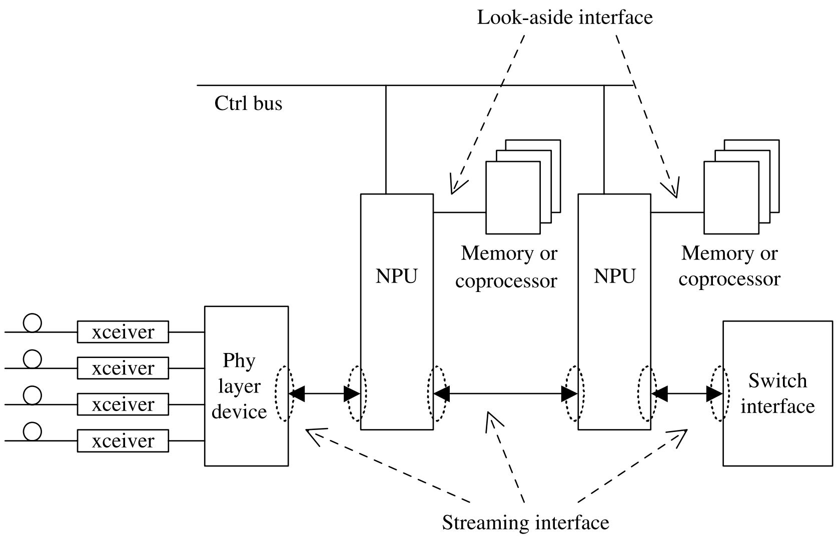


Figure 16.18 Linecard system block diagram.

assumptions are based on line-rate performance using 40-byte packets and 144-bit search keys [21].

LA-1 is a memory-mapped interface modeled after a synchronous separate I/O DDR SRAM interface and QDR is the basis for it. Using a DDR interface, data is clocked on the rising and falling edges of the clock signals. This effectively doubles the bandwidth of the interface without increasing the clock speed or the bus width. Overall, the LA-1 interface operates at clock speeds between 133 and 250 MHz. It supports unidirectional 18-bit read and write interfaces. These data inputs and outputs operate simultaneously, thus eliminating the need for high-speed bus turnarounds (i.e., no dead cycles are present).

The main bus signals are listed in Figure 16.19. Access to each port is accomplished using a common 24-bit address bus. Addresses for reads and writes are latched on rising edges of K and K# input clocks, respectively. Each address location is associated with two 16-bit data words that burst sequentially into or out of the device. Since data can be transferred into and

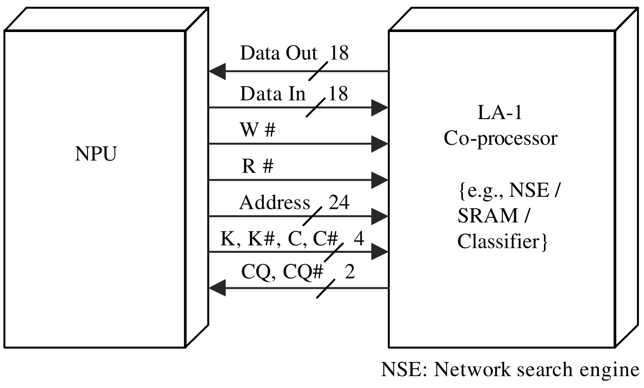


Figure 16.19 LA-1 bus interface signals.

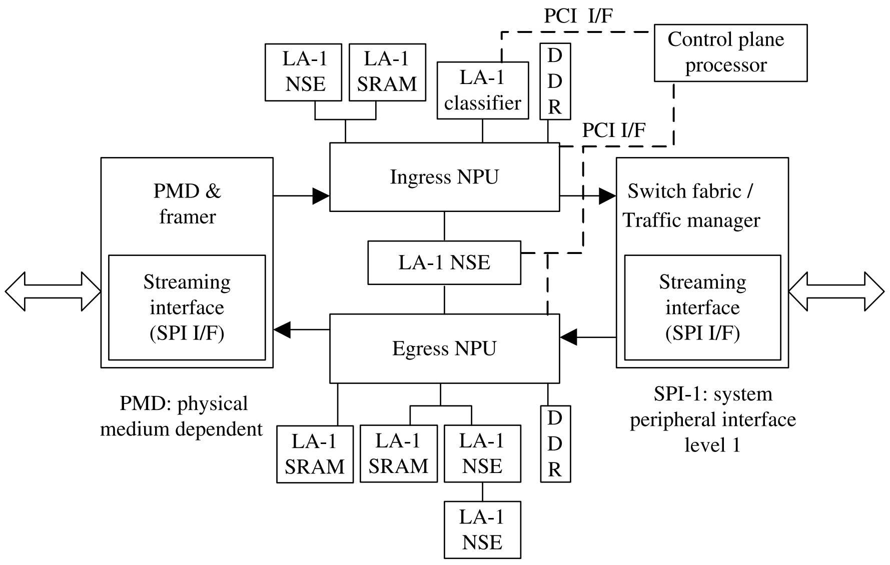

PCI I/F
LA-1 NSE
LA-1 SRAM
LA-1 classifier
DDR
Control plane processor
PCI I/F
PMD & framer
Ingress NPU
Switch fabric / Traffic manager
LA-1 NSE
Streaming interface (SPI I/F)
Streaming interface (SPI I/F)
Egress NPU
PMD: physical medium dependent
SPI-1: system peripheral interface level 1
LA-1 SRAM
LA-1 SRAM
LA-1 NSE
DDR
LA-1 NSE

Figure 16.20 LA-1 co-processor configurations and connections.

out of the device on every rising edge of K, K#, and CQ, CQ# (or C, C#) clocks respectively, memory bandwidth is maximized while simplifying overall design through the elimination of bus turnarounds. As the LA-1 interface uses a memory-mapped structure, a network processor can use the LA-1 address bus to initiate co-processor actions by reading and writing to memory mapped registers. Reads and writes to these memory-mapped registers in an LA-1 network search engine (NSE) allow the packet processor to issue search instructions to the NSE, retrieve returned results, and provide in-band management for NSE databases. NSEs are discussed in Section 16.2.2.

Figure 16.20 shows different LA-1 configurations for a typical line card application. One of the biggest advantages of using the LA-1 interface is the capability to support devices in a multi-drop configuration (i.e., multiple devices connected to the same interface). Hence system designers can support an NSE, SRAM, classifier and/or other LA-1 compatible devices on the same bus at the same time.

In some scenarios, two different types of devices can be placed on the same bus (i.e., NSE and SRAM). NSEs can be used to maintain the databases for different applications that need to be searched, and SRAM can be used for storing associated data. NSE searches require more write bandwidth whereas SRAMs are generally read intensive, so having two separate data buses allows the network-processing element to write to an NSE and read from an SRAM simultaneously without wasting any LA-1 bandwidth. Multiple configurations can be supported by adding more than one LA-1 interface to a co-processor. As shown in Figure 16.20, an NSE has two LA-1 interfaces that can be used to share databases between ingress and egress NPUs [22].

16.2.2 TCAM-Based Classification Co-Processor

IDT Network Search Engines and Search Accelerators. IDT has developed NSEs and search accelerators that utilize IDT's TCAM technology and high-performance logic technology. The NSE portfolio includes a pin-compatible family with a high performance interface available in 64K × 36 to 512 × 36 configurations as well as products that gluelessly interface to many leading packet processing silicon solutions such as those supporting an LA-1 interface. IDT also recently introduced a family of search accelerator products that supports core search speeds up to 250 MSPS with multi-search capability of 1 billion searches per second, an innovating 80-bit interface with optimized supports for complex IPv6 policy lookups, and integrated error correction code for improved data integrity.

An NSE with a TCAM core and an LA-1 (QDRII) interface may be connected directly to the QDR LA-1 interface of an NPU. Within a single clock cycle, the NPU may initiate an NSE search via the QDR LA-1 write data bus and obtain a search result via the QDR LA-1 read data bus. To accommodate the complex multi-threaded execution environment of today's NPUs, LA-1 NSE incorporates schedulers and mailboxes so that requests, results, and exceptions associated with various NPU threads can be properly handled. Beyond the basic search operation, additional NSE performance may be gained through architectural features such as reducing the number of NPU off-chip accesses by initiating multiple database searches using a single key as well as returning multiple match results from a single-database search. Further performance gains are possible through better implementation of address caches and flow tables by using intelligent learning operations, activity monitoring, and the notification of stale or aged database entries. The remainder of this section focuses on IDT's NSEs with QDR a single LA-1 interface: the IDT75K62134 (9 Mbit, 128 K × 72-bit) and IDT75K52134 (4 Mbit, 64 K × 72-bit) devices [23–25].

The IDT75K62134 and IDT75K52134 NSEs are designed to seamlessly connect to NPUs with LA-1-compliant interfaces operating up to 250 MHz. Up to four NSE devices can be directly connected to one QDR LA-1 interface, and up to eight devices can be cascaded from each of the four NSEs connected to the QDR LA-1 bus using point-to-point cascading to form on bank, also known as a search machine. In this way, up to 32 devices can interface with one NPU, as can be seen in Figure 16.21. Each bank allows a single database size of up to one million 72-bit entries while operating at maximum frequency. Each NSE supports 16 databases, 128 contexts, and 64 72-bit global mask registers (GMRs) that are shared across all contexts. The width of each database can be selected to be 36, 72, or 144 bits wide or can be programmed to be from 32 to 576 bits wide in increments of 32 bits.

Figure 16.22 illustrates the internal architecture of the IDT NSE with a single LA-1 interface. The NSE can perform a number of operations to support database maintenance

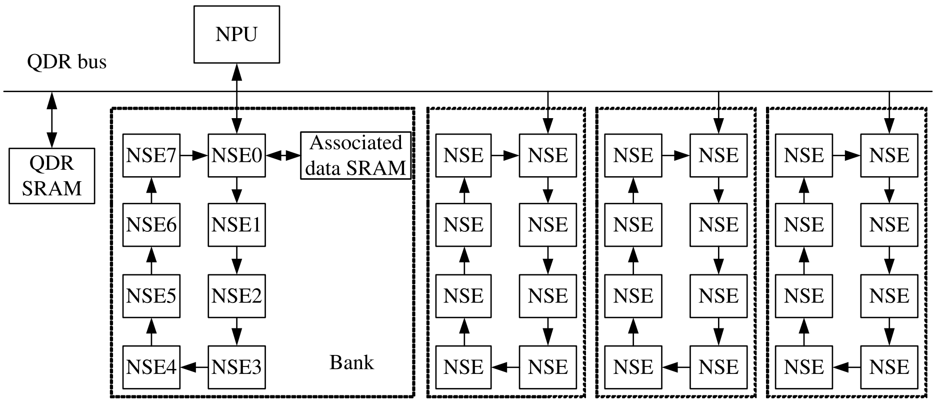


Figure 16.21 NSE configuration.

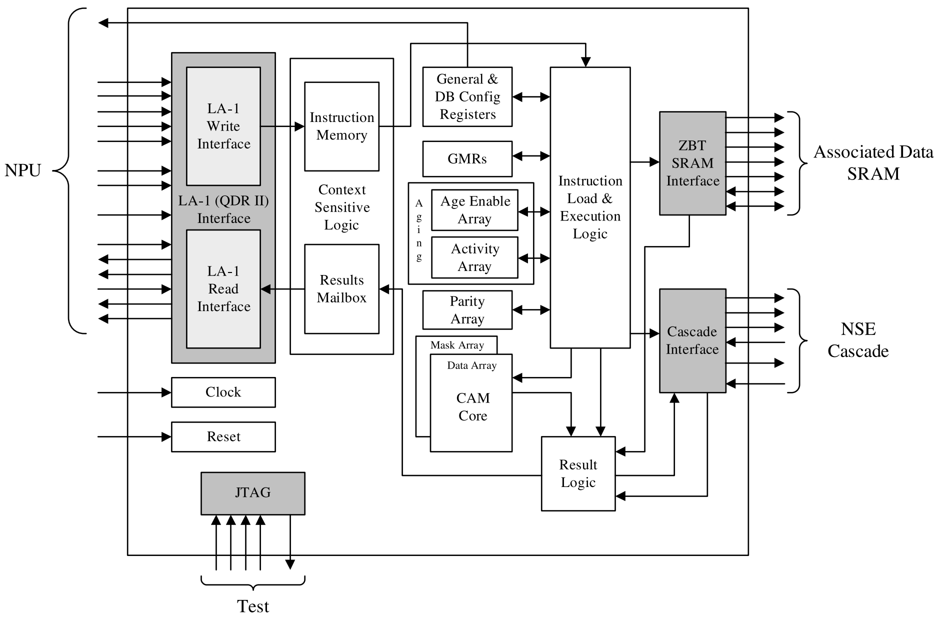


Figure 16.22 Architecture block diagram of the IDT NSE.

and searches. These operations include lookup, multi-hit lookup, multi-database lookup, re-issue multi-database lookup, learn, and multi-hit invalidate. Each data entry in the NSE has a valid bit that can be set, cleared, or left unchanged by using the appropriate instruction. Additional database maintenance support includes aging with age enable and activity arrays that provide independent aging control per data entry. A separate aging count interval is supported per database.

The NSE search and maintenance operations are initiated by writing to the NSE using the LA-1 interface. The operation command is encoded on the address lines, and the search key is presented on the data bus. The operation is placed in the instruction memory, where it is associated with a specified context that is often connected to a specified thread of execution in the NPU. Once a thread places a request in the instruction memory, the execution logic block presents the search operation, consisting of the global mask, the search key, and a CAM instruction to the TCAM core. The index results are stored in a mailbox for fetching by the associated thread of execution in the NPU. When a search is performed, the NSE can be optionally configured to fetch a 32-bit entry from the associated data SRAM. Results are read from the mailbox using the LA-1 interface. When the execution thread receives the result word, a “done” bit indicates if the rest of the result is valid. If the result’s read operation occurs before the NSE operation is completed, the done bit is not set, and the application must reissue the read command. Hit and multi-hit bits further qualify the result.

The execution block is also responsible for monitoring activities and indicating the lack of activities of entries being monitored. In addition, the execution unit maintains parity on the TCAM array, which can be interrogated through a background scrub operation using the parity check instruction.

The internal core of the NSE is divided into 4 K $\times$ 72-bit segments. There are 32 segments in the 75K62134 and 16 segments in the 75K52134. Databases are composed of a subset of these segments. To conserve power, only the segments that belong to the database being

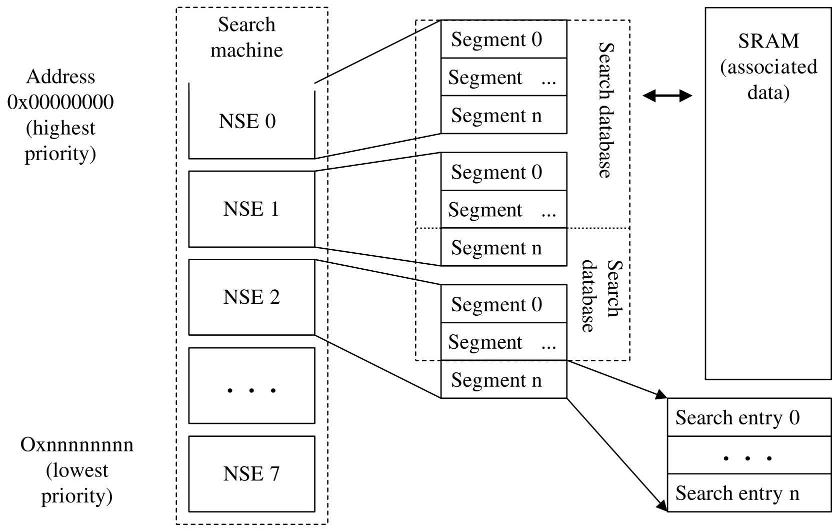


Figure 16.23 Search machine and logical database structure.

searched are powered up. The NSE instruction specifies a unique database on each operation. Figure 16.23 shows the logic structure for the database.

The IDT NSE also provides a set of comprehensive software packages for development support that includes system-level architectural models (SLAMs), data plane macros, complete product development diagnostics, and operational example applications, as well as a robust, production-quality control plane software library called the initialization, management, and search (IMS) library for runtime support.

IDT provides the IDT75KTA062134-200 development board to further accelerate application software development, as can be seen in Figure 16.24. The 75KTA062134-200

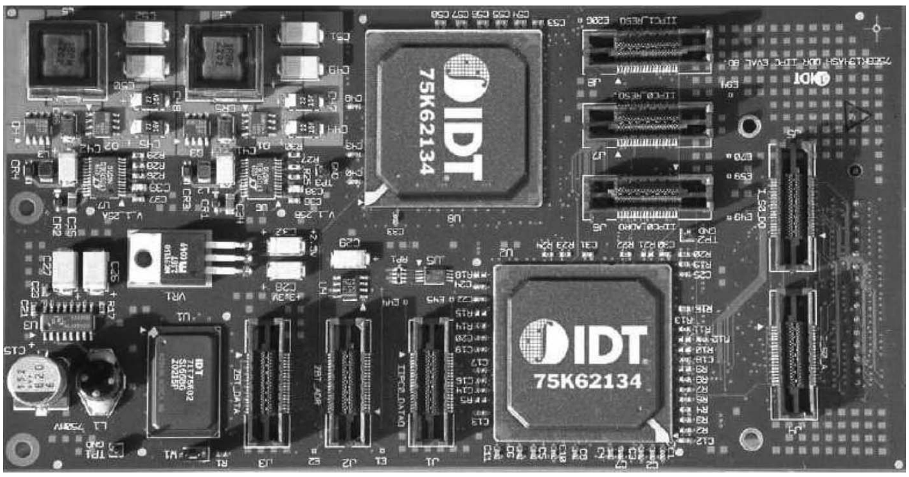


Figure 16.24 75KTA062134-200 development board photograph.

compliments the Intel IXDP2400 Development Platform to provide a complete hardware search accelerated network processing development platform. Utilizing the IDT 9M NSE with QDR LA-1 interface (IDT75K62134), the 75KTA062134-200 enables Intel IXDP2400 Development Platform to achieve 100 million searches per second (MSPS) using a 200 MHz QDR II bus frequency. Additionally, resources on the module include an on-board zero bus turnaround (ZBT) SRAM that allows for the storage of NSE associated data, and a QDR SRAM that is integrated for auxiliary NPU memory.

Cypress Ayama^{TM} 20000 Network Search Engine. Cypress Ayama^{TM} 20000 is a high-performance, pipelined, ternary NSE that provides a seamless interface to commercial network processors via the LA-1 interface. An Ayama 20000 device can interface with up to two NPUs using two separate LA-1 interfaces. This family consists of devices of multiple densities, including 512 K (18-Mbit) and 256 K (9-Mbit) entries. It includes a full T-CAM array with per-bit masking, enabling efficient searches for the applications including flow classification, lookup and packet forwarding.

Figure 16.25 shows the Ayama 20000 connected in a system. Ayama 20000 device supports up to two NPU LA-1 interfaces, one to an ingress NPU and another to an egress NPU. Both the NPUs are able to perform searches on the same database, eliminating the need for the redundant memory storage devices or NSEs. The Fast Link^{TM} NSE expansion interface connects Ayama 20000 to a cascade of multiple NSEs. In Figure 16.25, two different classes of NSEs are simultaneously connected to the Ayama 20000 device. A class

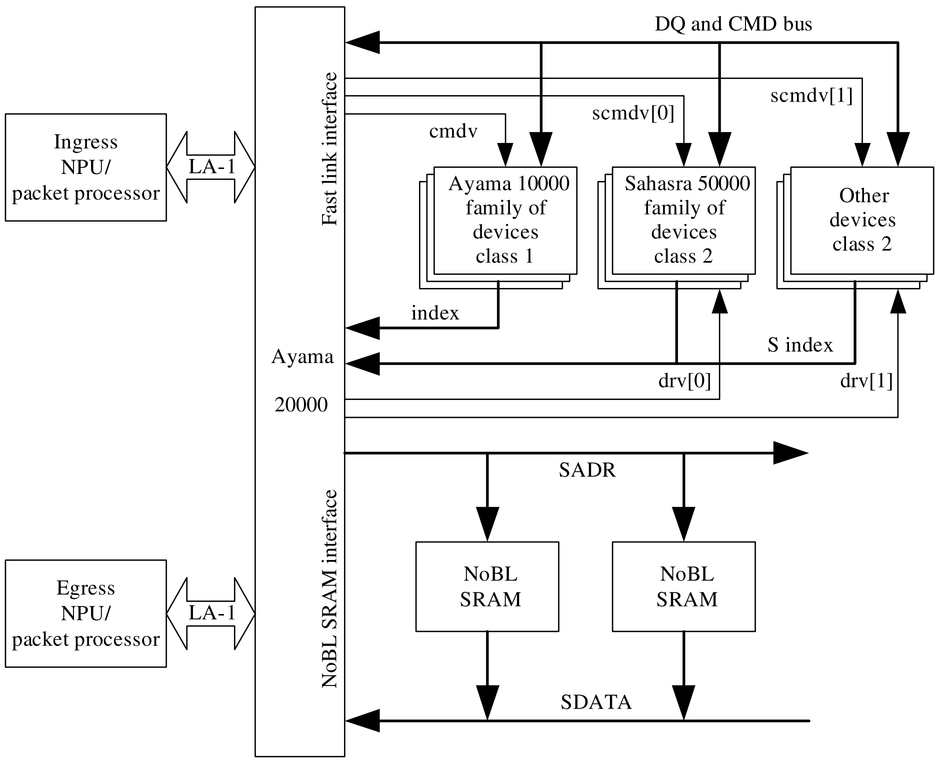


Figure 16.25 Ayama 20000 connected to heterogeneous mix of NSEs.

is defined as a group of NSEs sharing the same latency. The cascading of NSE devices allows NPUs to perform searches on different databases via the same LA-1 interface. The NPU can use the LA-1 port to access network policy information, QoS information and the destination port information for a data packet. The cascaded NSE devices share the same data queue (DQ) and command (CMD) bus, which supplies the search data and search parameters. The search results for each class are returned via individual result bus. This prevents contention due to different latencies that the NSE devices may have. The Ayama 20000 has an internal SRAM memory controller that resolves latency issues of the cascaded NSE devices. The search results returned from the NSEs are indices of the associated data in the SRAM. A SRAM lookup can be performed using their indices. The Ayama 20000 offloads SRAM lookup overhead from the NPUs by automatically performing associated data lookups, NSE lookups and associated data management as an atomic instruction. The Ayama 20000 communicates with the external SRAM via the No Bus Latency^{TM} (NoBL^{TM}) SRAM interface.

Figure 16.26 shows the internal block diagram of the Ayama 20000. The Ayama 20000 device has an internal NSE, an age memory block, one or two LA-1 interfaces (depending on the configuration), a NoBL SRAM interface, an external NSE interface and a control and arbitration block. The internal NSE performs database lookups, supporting up to 266 MSPS. The LA-1 interfaces connect directly to the commercial NPU. It also supports the SRAM interface standard QDR-I, running a burst of 2, at 133, 167, or 200 MHz and QDR-II, running a burst of 2 or a burst of 4, at 250 MHz. The SRAM interface connects to the external SRAM for associated data lookups. The age memory block keeps track of the frequency of usage for each database entry. If a database entry has not been used for a certain amount of time, the entry will be marked in the age memory. The information stored in the age memory can be used as the criteria for replacing or filtering out the older database entries. The control and arbitration block commands the rest of the circuitry inside the Ayama 20000 device.

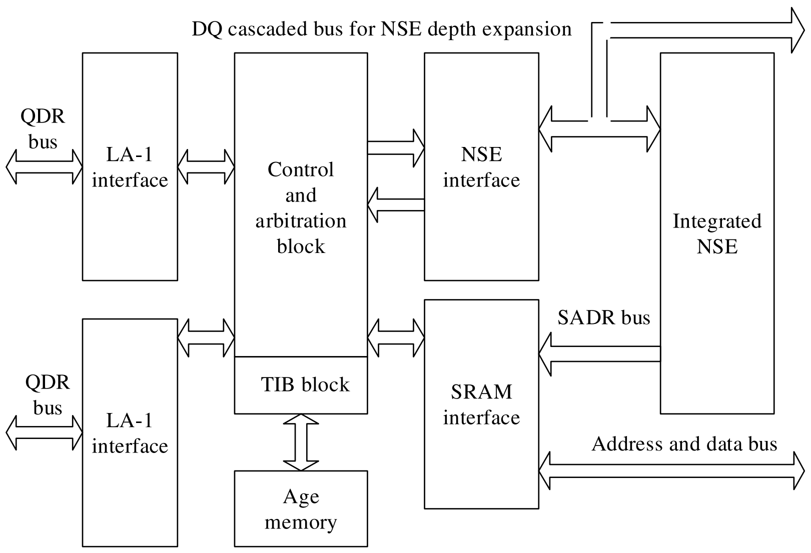


Figure 16.26 Ayama 20000 internal block diagram.

16.2.3 Algorithm-Based Classification Co-Processor

IDT PAX.port 2500™ Content Inspection Engine. The PAX.port 2500™ is an application specific standard product (ASSP) content inspection engine (CIE) with streaming interfaces. It is optimized for full layer 7 content inspection and regular expression matching at OC-48 in content aware networking equipment, web switches/load balancers, multi-service switches, IP service gateways, cable modem termination systems, edge routers, network security devices, and broadband wireless networking equipment [26, 27].

The PAX.port 2500 is optimized for use in flow-through applications with in-band classification at OC-48 or multi-gigabit Ethernet line rates. The PAX.port 2500 can be connected in a system either as a flow-through pre-processor or as a look-aside co-processor.

Figure 16.27 shows a PAX.port 2500 connected as a flow-through pre-processor in a line card. The PAX.port 2500 receives each packet from the input interface and prepends the classification results on the output interface. The input and output buses are identical. Up to three dedicated SRAM pattern memories contain the classification policies. A separate PCI host interface is used for programming and configuration. An IDT SRAM and/or NSE subsystem can be used by the ingress NPU to maintain stateful information for traffic flows. The classification results can be programmed to pre-extract a key, such as an IP 5-tuple, for efficient search operations to an NSE.

Figure 16.28 shows how a PAX.port 2500 can be connected as a look-aside co-processor in a line card. Typically a switch or bridge device is required to convert the SPI-3 (system peripheral interface level 3) input packet interface and output packet interface of the PAX.port 2500. This processing model allows the NPU to maintain control over which packets are passed to the PAX.port 2500. It also allows for modifications to be made to the packet prior to passing the data to the PAX.port 2500, such as trimming (i.e., stripping data that is not of interest to the PAX.port 2500), decryption, TCP termination and other functions to be performed prior to classification.

Figure 16.29 shows a simple block diagram of a PAX.port 2500. The PAX.port 2500 supports one physical SPI-3 path (input and output) configurable as one to 16 logical channels and performs wire-speed classification at a 2.5 Gbps rate for packet sizes of 40 to 1600 bytes. There are no per-channel throughput limitations. Typically, this

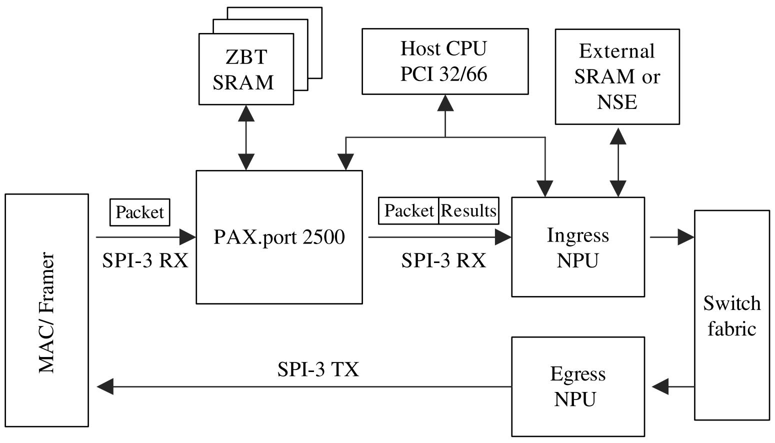


Figure 16.27 PAX.port 2500 used as a pre-processor in a line card.

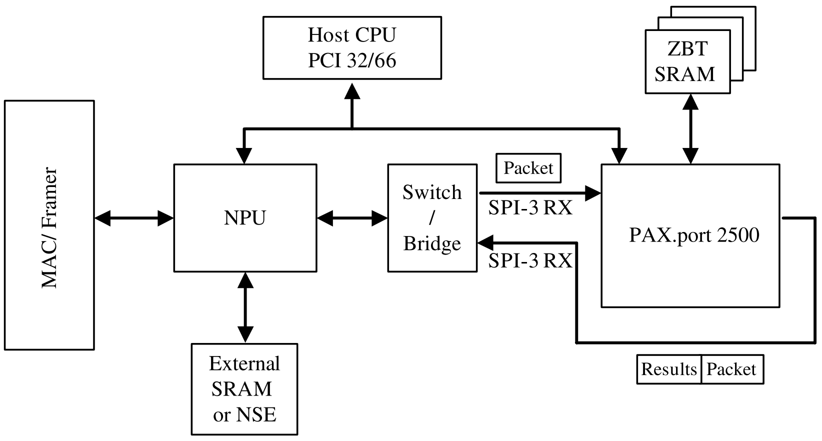


Figure 16.28 PAX.port 2500 used as a look-aside co-processor in a line card.

channelization maps to a single OC-48 channel, four OC-12 channels, or 16 OC-3 channels in POS applications, and various mixes of 10/100 and gigabit Ethernet channels in Ethernet applications.

The PAX.port 2500 has 15 classification cores. Each set of five classification cores has its own external memory called the pattern memory. Classification cores accept packet data and control signals from a dedicated, dynamically allocated and managed internal FIFO buffer. After classifying a packet, the classification core produces a classification result for that packet consisting of a tag (up to 64 bits) and a digest (up to 192 bits), which form a classification result together. The tag contents are programmable in a bitwise fashion and a digest is a parsed subset of the packet. Both of them can be prepended to the packet after

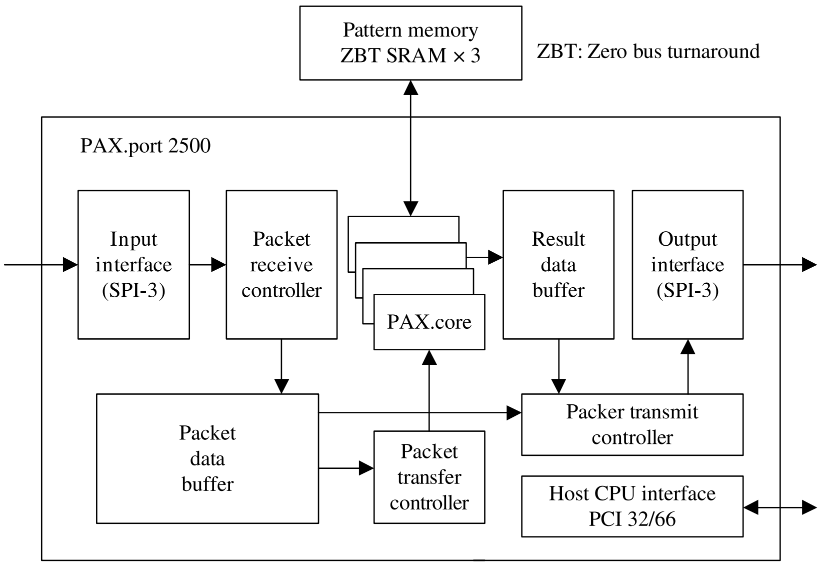


Figure 16.29 PAX.port 2500 block diagram.

each packet is classified and the packet is then transmitted from the PAX.port 2500 to the downstream device in the data path. The digest can also be used to store up to nine offset pointers (16 bits each).

The output interface can be overclocked by as much as 25 percent to support additional bandwidth contributed by the combined tag and digest. Classification results can also be prepended with a variable offset to overwrite up to 15 bytes at the beginning of the packet and thereby this mechanism reduces the aggregate bandwidth on the SPI-3 output interface.

The internal packet transmit controller (PTC) ensures that when classification is complete, all packets on a given virtual channel leave the device over the SPI-3 output interface in the same order in which the packets entered.

The PAX.port 2500 has three pattern memory interfaces, each providing up to 833 Mbps of classification performance for a total of 2.5 Gbps. It uses external 72-bit-wide or 36-bit-wide ZBT SRAMs to store classification policies. The number of policies depends on the complexity of policy rules for the particular application. Each of the three pattern memory interfaces is shared among five PAX.cores on a fair, round-robin basis. Each pattern memory interface can address up to 32 Mbyte of ZBT SRAM to support very large tables of complex policies.

The PAX.port 2500 has a PCI 32/66 host processor interface that provides an interface to any common central processing unit (CPU). The host processor interface provides access to option registers for configuration of the PAX.port 2500. This interface also provides the path for writing to and reading from the pattern memory.

The PAX.port 2500 classifies each packet using fully programmable classification criteria and results. Users can define any set of protocols and data patterns using IDT's high-level, non-procedural, PAX pattern description language (PDL), IDT's extensive protocol library, and application programming interfaces (APIs). Policies are loaded into external pattern memory, which the PAX.port 2500 accesses to classify the bit stream at wire speed. It is programmed using the PAX.works software development kit (SDK), which is a suite of software tools for both the development environment and the run-time execution environment. The SDK creates the pattern memory program that represents the classification rules for the PAX.port 2500.

PMC-Sierra ClassiPI^{TM} Network Classification Processor. The ClassiPI (classification by packet inspection) device assists network processors by offloading the complex and time-consuming task of packet classification [28]. It enables analysis and classification for all layers of a packet, including payload, at wire speed from 100 Mbps to gigabit LANs, as well as OC-48 and higher speed wide area networks (WANs). The ClassiPI PM2329 is the latest member in the ClassiPI^{TM} family [29].

The target applications of PM2329 include: simple switching and forwarding applications that require an exact match of L2 MAC addresses (or labels as in MPLS label switching routers) and a longest prefix match of L3 addresses (for MPLS label edge routers and IP forwarding), packet filtering applications such as a firewall, virtual private network (VPN), and QoS that require match, prefix, and range checks for L2, L3, and L4 addresses, and higher-layer applications such as server-load balancing, Web caching and intrusion detection that require lookups in the payload (L7) portion of the packet.

The general architecture of a PM2329-based system is shown in Figure 16.30. The network processor or customer ASIC sits in the data path and buffers the packets upon

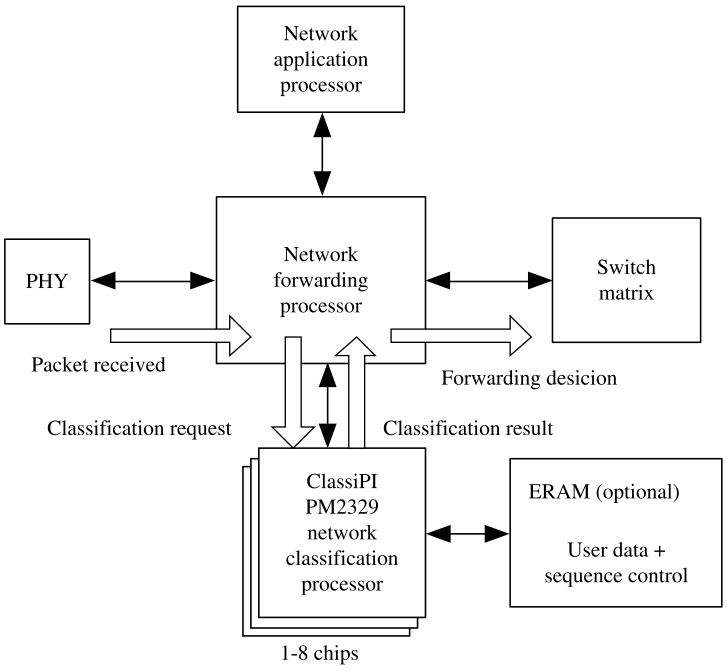

Network application processor
PHY
Network forwarding processor
Switch matrix
Packet received
Forwarding desicion
Classification request
Classification result
ClassiPI PM2329 network classification processor
ERAM (optional)
User data + sequence control
1-8 chips

Figure 16.30 ClassiPI-based network equipment architecture.

arrival. The ClassiPI device is attached to the generic search machine interface as a co-processor. The network processor or customer ASIC transfers the appropriate payload to the ClassiPI device depending on the type of lookup that needs to be performed. Based on the preprogrammed rule set in the ClassiPI device, the result of classification is returned to the processor that can further perform appropriate editing and forwarding of the packet. The ClassiPI supports up to two optional external SSRAMs connected using a dedicated bus, which can be used to implement more complex classification lookup sequences and to store packet parameters and user data associated with rules.

The internal architecture of PM2329 is shown in Figure 16.31. It features an efficiently pipelined architecture, enabling a continuous stream of packets to be fed into the device, while it continues to perform packet parsing, key formation, and lookup operations.

The system interface is a general-purpose synchronous SRAM interface (configurable as 32 or 64 bits wide operating in SyncBurst or ZBT mode) for connecting to a processor, packet source, or DMA (direct memory access) device. This interface is used to send packets or pre-extracted payload for classification. The results of a classification operation are placed in the result FIFO and accessed by the processor through this interface. This interface can also access the control registers that are used to configure classification operations, key selections, and the rule database. It supports up to 32 independent channels, each of which has a separate area in the packet input buffer and result FIFO, to simplify the interaction with external multi-context processors.

The field extraction engine (FEE) includes an 8 kbyte packet input buffer that can store up to 32 packets simultaneously. Whenever the external device (processor or other packet source) sends packet data to the PM2329, the packet is first deposited in the packet input buffer. The FEE can parse and extract Ethernet from layer 2 (including Ethernet II, 802.3,

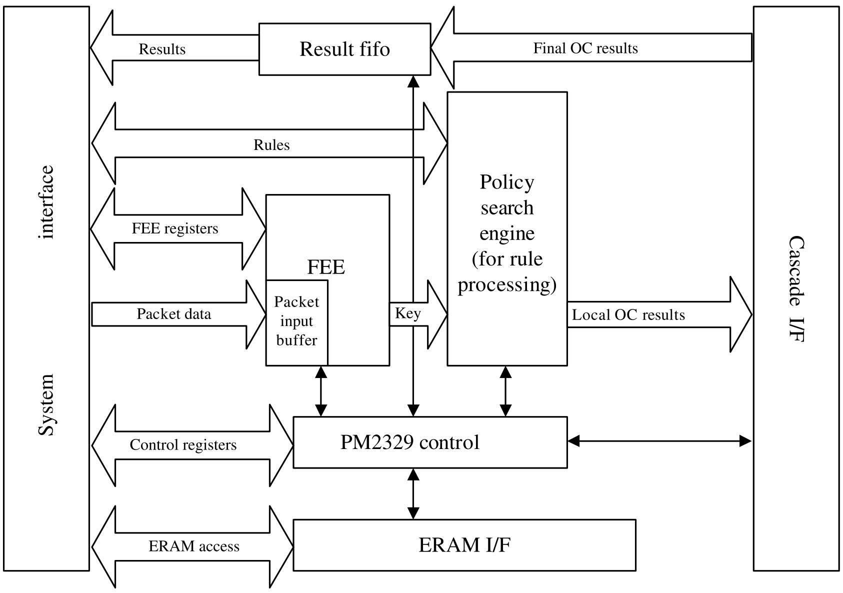


Figure 16.31 PM2329 block diagram.

and 802.1p/q headers), IP at layer 3, and TCP/UDP at layer 4 as well as payload data at any offset.

The policy search engine implements a set of classification functions. It performs policy-based search operation sequences, applying rules from the 16 K-deep rule memory and comparing a specified sequence of rules with the extracted key information from the FEE. It generates detailed results of those searches and returns those results (under control of the control unit) through the cascade interface to the result FIFO. The PSE performs operation cycles (OC) on packets in the order that their field extraction is complete. The processing of any one search using one key constitutes a single OC. As there could be multiple keys extracted from a packet, and as each key can be subjected to multiple searches, a packet can utilize many OCs for full processing. An OC is complete when the key is compared against a specific rule, and the results are queued in the result FIFO. The time duration of an OC depends therefore on the availability of space in the result FIFO; if the OC must wait for space, it remains active until space is available.

The control unit orchestrates the operation of the internal blocks to perform packet classification. It consists of registers required to control these operations, and state machines to perform the control functions.

The external RAM (ERAM) stores: (1) the classification program in the command RAM (C-RAM section), and (2) the user programmable data associated with every rule in the User section; (3) packet count, byte count, and timestamp statistics maintained by the device on a per-rule basis in the Stats section. The ERAM interface has a shared address bus and two data buses – one for command and the other for statistics and user data. As a consequence, two types of cycles can run on this interface. When fetching the command word, the data read from the command section of the ERAM is delivered to all devices in the cascade, whereas accesses to the statistics and user section of the ERAM are targeted between the specific PM2329 and ERAM as is determined by local on-chip ERAM configuration register in each PM2329 device.

The cascade interface is used to connect up to eight PM2329 devices to increase the number of rules in the rule database to a maximum of 128 K. Each of the devices in the cascade receives the same key but operates on different rules in parallel. The cascade interface implements a handshake mechanism to make the cascade appear as a single large device to the network processor.

16.3 TRAFFIC MANAGEMENT CHIPS

16.3.1 Overview

Traffic management is defined as the ability to control data flow across the network. A traffic management chip handles queuing and scheduling. It applies different buffering strategies so that flows can share limited buffers according to the traffic requirements. When a traffic management chip schedules cells or packets, it must make sure flows can share the limited bandwidth properly. At the same time, it will also take different protocols and QoS requirements into consideration. Nowadays, a network processor can handle flows at OC-48 or OC-192 line rate or even faster. As a workmate component of network processors, the performance of traffic management chips is becoming increasingly more critical due to the factors described above.

Traffic management chips bridge the network processor and the switch fabric. It handles numerous flows of a variety of protocols. The ingress chips process flows from the network processor before being sent to a switch fabric to resolve bandwidth congestions. It applies buffering and scheduling algorithms to these flows. This chip is also responsible for multicasting. The egress chips process data from switch fabric before data is sent to the network processor. If packets are segmented into cells in the ingress chip, reassembling the cells into an original packet is one of the important tasks of the egress chips.

There are different traffic management chips from the vendors, based on different commercial platforms and system architectures. Agere, ZettaCom, and AMCC are some of the pioneers in this field. In this section, we will discuss traffic management chips called traffic manager (TM) in more detail.

16.3.2 Agere's TM Chip Set

In this subsection, we will go through Agere's high-end traffic management chip set called the TM10 [30–32]. The TM10 is designed to work with the NP10, which is a network processor by Agere, to provide a complete 10 Gbps traffic management solution.

The TM10 is a high-speed, high-performance, programmable traffic management engine providing protocol data unit modification, queuing, buffer management, scheduling, and shaping.

Architecture and Implementation. Figure 16.32 illustrates the data flow and major functional components of TM10. Details of the data flow and major components in TM10 are described in the following.

As shown in Figure 16.32, first, the TM10 receives PDU and its control information from a network processor or from a switch fabric. This work is done by the input interface. The PDU is the formatted data unit including protocol information, QoS parameters, and data payload. Control information is received from the NP10 through the SPI-4 interface, which

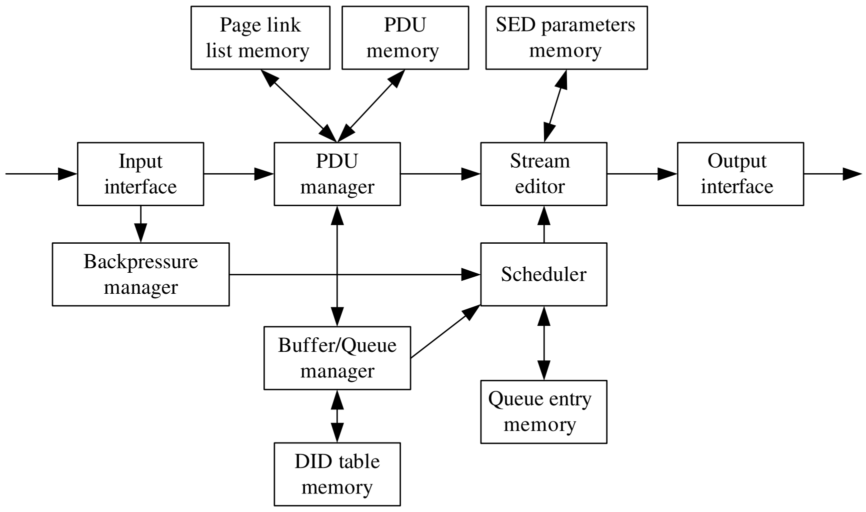


Figure 16.32 TM10 block diagram.

is in the form of a transmit command, while control information is received from switch fabric in the form of backpressure signals for ports[1] and queues.

After that, the TM10 stores the PDUs and the related information into PDU memory. The PDU manager allocates memory pages for the incoming data. Because one packet may be segmented into multiple PDUs, these PDUs occupy more than one block and the blocks for these PDUs are linked together in the page linked list memory. The transmit commands associated with the PDUs are forwarded to the buffer and queue manager. The PDUs are segmented before being switched so that they may be interleaved while being switched. The PDU manager is in charge of reassembling egress PDUs into packets in the egress side of the switch fabric.

The buffer management process begins on the arrival of PDU and its transmit command. The buffer and queue manager checks transmit command to access the destination ID (DID) table memory. Based on the information both in the transmit command and in the DID table memory, it determines whether to queue or to discard arriving PDU. If accepted, corresponding enqueue operation will be finished by the PDU manager.

Finally, the TM10 schedules, modifies and transmits the PDUs. The scheduler determines which queue is to be serviced for the next scheduling slot. When a scheduling result is ready, the scheduled PDU will be read out from the memory and modified by the stream editor. Then the modified data block is sent to the output interface.

Buffer and Queue Management. The TM10 provides sophisticated and highly configurable mechanisms for buffer and queue management. It supports four types of memory management, as shown in Figure 16.33: (1) Reassembly buffer management keeps track of the number of memory pages used to reassemble PDUs as they arrive, PDUs will be enqueued after reassembly; (2) PDU buffer management keeps track of the number of memory pages that are used by enqueued PDUs; (3) Multicast staging buffer management keeps track of the control information for multicast PDUs, allows prioritizing of multicast

[1]These ports are not the ones when describing a router or switch. In this case, they mean queues in the structure of VOQs.

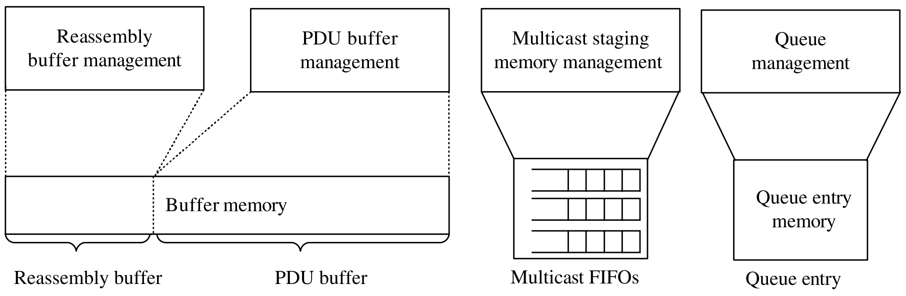


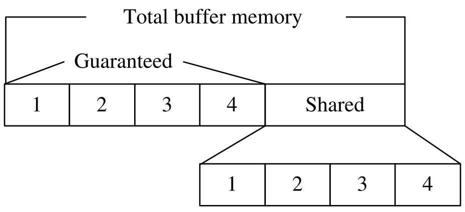
Figure 16.33 Four types of memory management in TM10.

PDUs; (4) Queue management keeps track of the number of enqueued PDUs that belong to a particular queue or group of queues.

The first two mechanisms described above are used to protect and manage the TM10 PDU buffer. At configuration stage, the user can partition the PDU buffer into two parts: memory for reassembly and memory for enqueued PDUs.

The memory for reassembly is to allow the customization of the TM10 for specific applications or traffic conditions. The TM10 uses a set of external data buffers for PDU reassembly purposes. The reassembly space is partitioned into eight regions, corresponding to eight reassembly classes. Each reassembly class is configured with a minimum guaranteed reassembly space, as well as with a maximum usable reassembly space. This scheme allows the dynamic sharing of the reassembly space among classes. As a result of the classification process in NP10, each PDU is assigned a priority. These eight priorities are mapped into the eight priority class buffers in the reassembly buffer, respectively. These eight prioritized class buffers provide space isolation for each class. Reassembly buffer management uses a portion-sharing strategy. Each priority class buffer has a minimum and maximum threshold. The minimum threshold is the guaranteed buffer space for a certain priority class. A PDU will be accepted for reassembly if the occupation is below the minimum threshold. The maximum threshold is based on the usage of shared space and the sum of all the maximum thresholds can exceed the entire buffer space. If there is some excess shared buffer space and the occupation is below the maximum threshold, the maximum threshold can be guaranteed and a PDU will be accepted. In order to ensure that the maximum amount of reassembly memory is not exceeded, the global threshold is applied.

The TM10 uses another part of the external data buffer to store PDUs while they wait for transmission. The memory pages are tracked by the PDU buffer management process. A PDU occupies its buffer space from the time of being admitted into the PDU buffer until being transmitted out of the buffer. The TM10 provides a programmable pool-level buffer management mechanism. A buffer pool is a set of PDUs that either shares some common attributes or can be viewed as a single entity for buffer management. The TM10 supports up to 4096 such pools. For example, all flows with the same level of priority and destined for the same port can be assigned to one pool. Each of the TM10's 4096 pools is assigned a minimum (guaranteed) number of memory pages, and a maximum number of memory pages. The sum of the minimum thresholds is the guaranteed memory assigned to the pools, and the rest of the memory is shared. This means the memory space can be assigned to the pools dynamically based on their maximum thresholds, as shown in Figure 16.34. The PDU buffer management supports two forms of weighted random early detection (WRED): pool-based WRED, to protect the pool's shared memory and global shared buffer space



Each pool can have a minimum and maximum page threshold. The minimum threshold is guaranteed.

Figure 16.34 Minimum and maximum pool thresholds.

WRED, to protect all of the shared memory. These WRED mechanisms can be enabled and disabled at discretion.

Multicast staging buffer management uses a complete partition strategy. The multicast staging buffers, which are organized in the form of FIFOs, are used to temporarily hold multicast PDU information before it is added to the queue. These FIFOs are used to prioritize multicast PDUs. After a multicast PDU has been admitted into the PDU buffer space, its control information is placed in one of the eight multicast staging FIFOs. The control information remains in the FIFO while the TM10 attempts to enqueue the PDU into all the queues corresponding to the multicast group membership. The TM10 determines which of the eight FIFOs to use based on the information in the DID entry. This allows different levels of multicast priority for different multicast flows. The multicast staging FIFOs can hold the control information for up to 1 K multicast PDUs. This entire space is statically partitioned into eight FIFOs. Unlike the other managed spaces, no sharing of multicast staging FIFO space is permitted. A multicast PDU will only be admitted into the requested FIFO if its occupancy is lower than the maximum allowed value for that FIFO. When the requested FIFO is full, the multicast PDU is discarded.

Queue management is designed to protect the queue entry memory, which maintains entries for each PDU and is named QRAM by Agere. A PDU may occupy one or more buffer pages; an entry is associated with a buffer page. For unicast traffic, since the number of possible entries in the QRAM is the same as the number of PDU buffer pages, it is not possible to run out of QRAM, even when the buffer is filled with PDUs that occupy a single page. However, with the multicast case, one PDU in the PDU buffer memory can be scheduled into multiple queues, increasing the number of entries in the table. The TM10 supports up to 16 K scheduling queues. When the TM10 is ready to enqueue a PDU into a queue, it first performs the queue entry space management function, making sure that no single pool of queues uses a disproportionate amount of queue entry space. The process for queue entry space management is very similar to the process performed for PDU buffer space management. The main difference is that the queue-entry space management deals with fixed size allocations, where each allocation unit corresponds to a single queue entry. The PDU buffer space management, in contrast, deals with variable size allocation, accommodating different-sized PDUs. The queue entry space supports up to 4 K pools, with minimum and maximum thresholds per pool. Each PDU has flow-based and profile-based thresholds, as is the case with PDU buffer space management. The enqueueing decisions are made using WRED, and are based on current occupancy levels of the pools, as well as on the flow and profile of the PDU being enqueued.

These four types of memory management protect the entry buffer and payload buffer based on priority and QoS parameters. It can solve the buffer congestions in different levels.

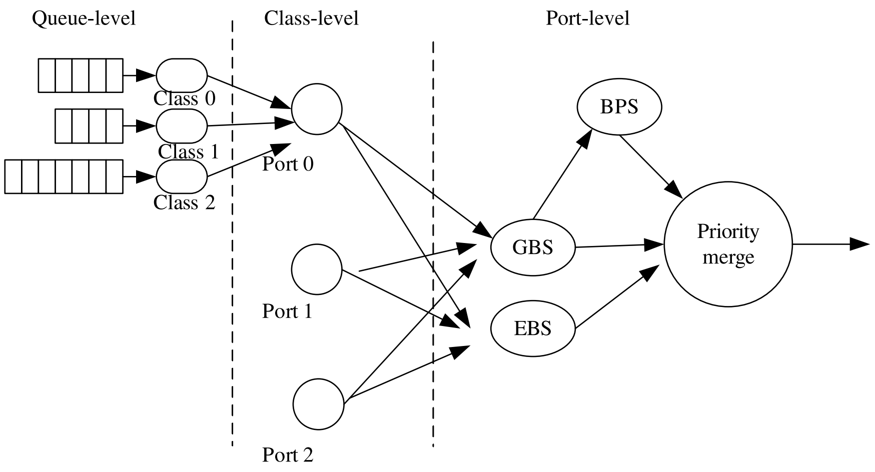


Figure 16.35 TM10 scheduler hierarchy.

Scheduling and Shaping. The TM10 device uses a hierarchical scheduling structure that provides bandwidth guarantee and traffic isolation at three levels. A port-level scheduler is the first level of the hierarchical scheduling structure. The port-level scheduler arbitrates among traffic flows, which are destined for different ports. The port-level scheduler's structures for both the ingress and the egress directions are similar, and are configured separately for each case. A class-level scheduler is the second level, which can provide bandwidth and delay guarantees, fair sharing of bandwidth, and weighting among four classes. A queue-level scheduler is the bottom level of the structure. The queue-level scheduler provides fair bandwidth sharing and minimum bandwidth guarantees among the 16 K queues. The scheduling hierarchy is shown in Figure 16.35.

The port-level scheduler selects a port for service using three different kinds of schedulers as shown in Figure 16.35, including the guaranteed bandwidth scheduler (GBS), the excess bandwidth scheduler (EBS) and the backpressure scheduler (BPS). These three schedulers are serviced in a strict priority order: GBS, BPS then EBS. If the GBS has a PDU ready to transmit, it is serviced first, if there is no GBS PDU ready, the BPS scheduler is serviced next. If there are no GBS or BPS pages to transmit, the EBS scheduler will then be serviced. BPS and EBS schedule the ports in a round robin (RR) manner.

The GBS is the highest priority port scheduler. This scheduler uses a shaped virtual clock algorithm that assigns a minimum bandwidth to each port and keeps track of the next service time needed to maintain the minimum bandwidth. The virtual clock algorithm provides accurate rate-based scheduling in a form of weighted fair queuing (WFQ). It also provides a tighter delay bound than a weighted round robin scheme. An ending timestamp is calculated for each backlogged port, based on the current clock value and the length of the PDU. The scheduling mechanism works to service the PDUs with the earliest timestamps, while maintaining assigned port rates, using a set of rate groups.

At the second scheduling level, service priorities among different scheduling classes within each port are managed by a scheduler, which runs in a class-level. Within each managed port, traffic is divided into four priority classes. Schedulers of this level run a strategy called smoothed deficit weighted round robin (SDWRR) or serve the classes in strict priorities. There are four classes for each scheduler: one priority class plus three

SDWRR classes. The TM10 can be configured to use one of the scheduling schemes to serve the traffic among different classes.

Traditional implementations of deficit weighted round robin (DWRR) scheduling employ a single FIFO queue. At each scheduling event, the entire list is examined without regard to the previous scheduling event, until the allocated bandwidth is exhausted. This approach induces extremely poor performance in terms of latency and service burst, since in some cases some of the round robin members can be serviced twice before other members are serviced once. The TM10 employs a smoothed version of DWRR. This approach interleaves the transmission of packets from different scheduling events, in a two-level mechanism that ensures fairness and reduces burst and latency. In SDWRR, service is distributed on a PDU basis. PDUs from different sessions are served in a round robin fashion. The bandwidth sharing is fulfilled in a collective way that prevents one member from being accessed twice before others are accessed only once. The SDWRR scheduling algorithm is implemented in both the class- and the queue-level schedulers.

At the third scheduling level, service priorities are managed among different queues within a given class on a given port. A queue in TM10 can be viewed as a collection of virtual connections or IP flows. The virtual connections or IP flows are treated as a single entity from the traffic management perspective and addressed by queue identification (QID). The TM10 supports 16 K queues. SDWRR, as described previously, is used to serve the different queues in the class.

Modification. This work is finished by the stream editor (SED). The SED allows for PDU modifications to be programmed for each DID. The PDU modifications can be defined at system initialization or can be dynamically assigned during operation. PDU modifications are performed by compute engines with specifically assigned parameter values and script instruction sequences. These scripts and parameter variables are part of the programmable nature of each destination ID definition. Typical applications for PDU modifications include: encapsulating and tunneling PDUs into appropriate IP and other protocols, implementing IP operations such as decrementing time-to-live counts, IP packet fragmentation when required by the outgoing link and performing various MPLS operations such as swapping labels or pushing tags.

Chip Set Features. Based on the architecture we described above and strategy used in buffering and scheduling, the TM10 can achieve the following features and benefits.

TM10 can get a full 10 Gbps line rate performance with any PDU size (40 bytes or higher), supporting collaborations with both the cell-based and the frame-based switch fabrics. The TM10 has good support for multicast traffic too. By employing four types of memory management process, the TM10 can provide advanced, programmable buffer management and discarding policies. The TM10 provides up to 4096 buffer pools, each with programmable guaranteed and maximal space sizes. RED and WRED discarding policies are applied for each buffer pool. The TM10 uses per-flow buffer space thresholds for making a discard decision and per-flow and per-profile RED weights, allowing for better traffic isolation. These features enable direct support of DiffServ AF (assured forwarding), EF (expedited forwarding) and PHBs (per-hop behaviors), predictable PDU drop probability, and support for a large number of classes of service.

Hierarchical scheduling enhances advanced, programmable scheduling policies. The TM10 provides bounded delay, guaranteed bandwidth services down to byte-level granularity, as well as excellent fairness between traffic classes. Shaped virtual clock (SVC)-based shaping achieves guaranteed packet delay bounds. The TM10 supports SDWRR and strict priority policies. It can schedule up to 256 switch fabric ports, with four priority classes per port. Each traffic class within a port has a separate backpressure assertion. These scheduling capabilities can be easily programmed for different applications.

16.3.3 IDT TM Chip Set

A typical traffic management chip provided by IDT is called the TTM552 [33–35]. The TTM552 can be used as a standalone flow-aggregated traffic management device. Another extended chip provided by IDT is called the TTM553, which cooperates with the TTM552 and enhances the capacity of the TTM552.

The TTM552 traffic manager can perform flow-based traffic management for up to one million flows, when it is used in conjunction with the TTM552, including queuing, scheduling and buffer management. In this subsection, we will describe how the IDT TTM performs these operations.

Architecture and Implementation. As shown in Figure 16.36, the TTM552 receives packets or cells channeled on the Rx traffic interface. Then it assembles the packets in the arrival reassemble queues (ARQs). Before data is stored in the buffer memory, the TTM552 performs congestion management checks based on the packet's discarding preference, forwarding label, configured thresholds, and on the measured levels of congestion. These checks may result in partial or full packet discard. If not discarded, packets or cells are stored into the buffer memory. The TTM552 places the packets in appropriate aggregated flow queues (AFQs). If the TTM552 is present, packets are placed in the appropriate flow queues (FLQ) controlled by the TTM553. At the same time, the TTM552 updates the statistics and scheduling database via the stat port.

After buffering operations, the TTM552 makes bandwidth management (scheduling) based on the configured parameters, the selected scheduling algorithms, and the received flow control and backpressure information. When a cell or a packet is scheduled, the TTM552 updates the statistics and scheduling database. Then the cell or packet is retrieved

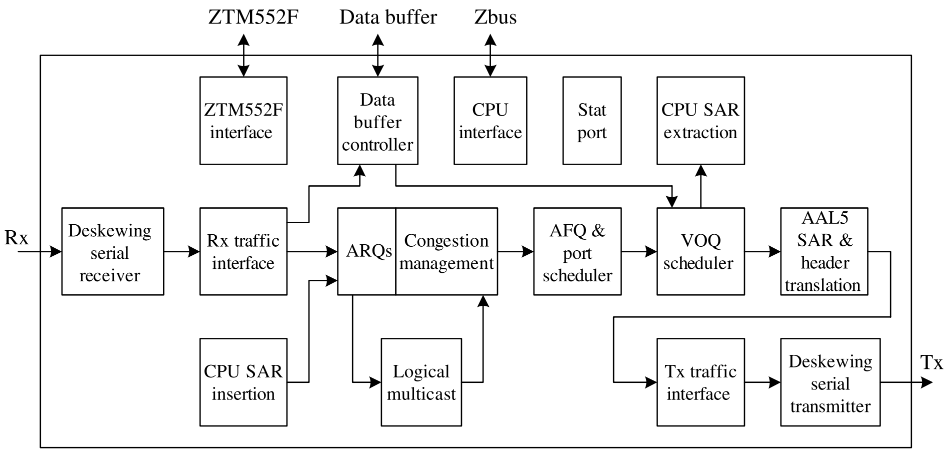


Figure 16.36 IDT TTM block diagram.

from the memory. After header translation is applied to the PDUs, the cell or packet is transmitted through the Tx traffic interface.

There are other functions provided by the TTM552. In the TTM552 system, multicast is supported and multicast cells are replicated by the logical multicast module. After replication, cells will be treated as unicast cells. Another function provided is that control messages from or to the CPU can be inserted or extracted over the CPU interface via the integrated segmentation and reassembly (SAR).

Congestion Management. Cells or packets from the network processor or switch fabric are received via the Rx traffic interface. The interface consists of an input data interface and an out-of-band flow control interface. Packets arrive at the TTM552 over the Rx traffic interface using one of the standards such as SPI-4.2, CSIX or NPSI specifications. Protocol headers of the cells or packets are examined to generate protocol independent cell (PIC) headers, and payloads of the cells or packets are extracted from the frames. If needed, they are segmented into multiple internal fixed-length cells. A loss-of-syn signal transmitted from up-stream is detected at the same time. Receipt of such a signal can either automatically trigger a training pattern transmission on the status interface that is used to communicate backpressure information to the upstream device, or simply generate an interrupt to allow software handling of the event.

The TTM552 uses ARQs to reassemble packet segments that have been interleaved by the NPU, switch fabric, CPU-SAR-generated traffic, or through logical multicast replication. The segments cannot be scheduled for departure or for next operations before they are reassembled.

Both in the ingress direction and in the egress direction, after the reassembling operations, the TTM552 should apply congestion management before cells or packets are stored into the memory, called buffering. The TTM552 performs congestion management by means of discarding threshold or congestion avoidance. When the TTM553 is present, it enables congestion management on every individual flow.

The TTM552 makes decisions to accept or discard every arriving packet based on these two types of information: current congestion conditions in the data buffer space and the thresholds configured for each queue. Each queue threshold and the memory usage are checked before the cell is accepted into the ARQ. The system can configure each of the queue levels with thresholds independently to achieve maximum flexibility in the allocation of data buffer resources. There are four levels of thresholds employed by the IDT TTM, including global data buffer maximum threshold, per port queue (PQ) maximum threshold, per AFQ maximum threshold, and per flow identification number (FIN) maximum threshold. All levels of these thresholds can be enabled or disabled. Multi-level thresholds make congestion management more flexible. The relationship of these thresholds and their operation order is shown in Figure 16.37.

There is a single, configurable global data buffer threshold. This threshold represents the entire data storage capacity that the TTM552 can sustain. If the entire data storage reaches the defined global threshold, the TTM552 discards all the arriving cells.

Similarly, there is a single, configurable discard threshold for each destination PQ. A unique maximum threshold can be set for each PQ. Same as global threshold, if the occupation of the PQ exceeds this threshold, the arriving data will be discarded.

Each AFQ can be configured with up to a group of thresholds called a set. For each AFQ, we can choose one of the 256 AFQ threshold sets. Also there is one configurable reference maximum threshold (RMT), which is always applicable to traffic arriving at the AFQ.

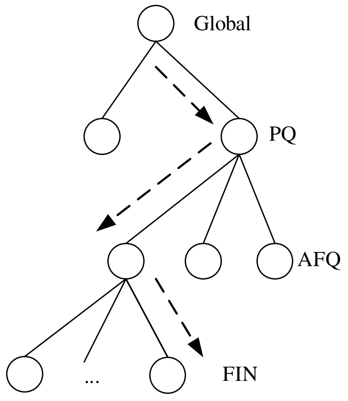


Figure 16.37 Congestion management of the TTM552.

Each of the 4 K AFQs has its own RMT that represents all of the memory allocated to it. When the usage exceeds the RMT value, packets or cells will be discarded. Each threshold set consists of eight configurable and selectable threshold multipliers and a configurable SOP (start of packet) flag associated with each threshold multiplier. A threshold multiplier determines the actual threshold to be applied to a packet. The actual threshold is a percentage of the AFQ's configured reference maximum threshold. The multiplier is chosen according to the priorities. If the SOP flag is active, then the result is only applied to the SOP cells.

If the TTM553 is connected, there is a single, configurable discard threshold for each destination FLQ called FIN maximum threshold. If the occupation of the FLQ exceeds this threshold, the packet belonging to this flow will be discarded.

In addition to the static threshold, the TTM552 supports congestion avoidance. It supports WRED for AFQs. The TTM552 also employs an explicit forward congestion indication (FCI) feedback mechanism for congestion avoidance. They function as indicators of local congestion that can be mapped by a downstream network processor or other protocol-aware service engines into a flow control message in the native protocol.

After performing the congestion management operations, the TTM552 stores accepted packets or cells into the memory and discards others. The TTM552 supports full packet discard (FPD) to reduce local memory congestion. When the SOP cell is discarded, the TTM552 will discard the entire packet. This method increases the efficiency of cell memory usage. When a non-SOP cell is discarded, the IDT TTM supports a version of FPD that is similar to partial packet discard (PPD). The discarding of a non-SOP cell arrival will result in all subsequent cell arrivals for the packet being discarded. A partial packet that was assembled in the ARQ or FLQ prior to the first discard is also deleted by forwarding this partial packet to the designated "trash queue."

Queuing and Scheduling. After congestion management, acceptable cells will be buffered into the memory in the form of queues. There are five types of queues in which data can be stored while being processed by the TTM552. They are ARQs, FLQs, AFQs, PQs, and output queues (OQs). The FLQs are only supported when the TTM553 chip is connected. In the following, we will describe the queuing structure of the TTM552.

As shown in Figure 16.38, ARQs are the first-stage of queuing for arriving packets. The TTM552 supports 2048 ARQs. The ARQs accumulate an entire packet before moving it to the next stage queuing. The next stage queuing could be FLQs if the TTM553 is connected

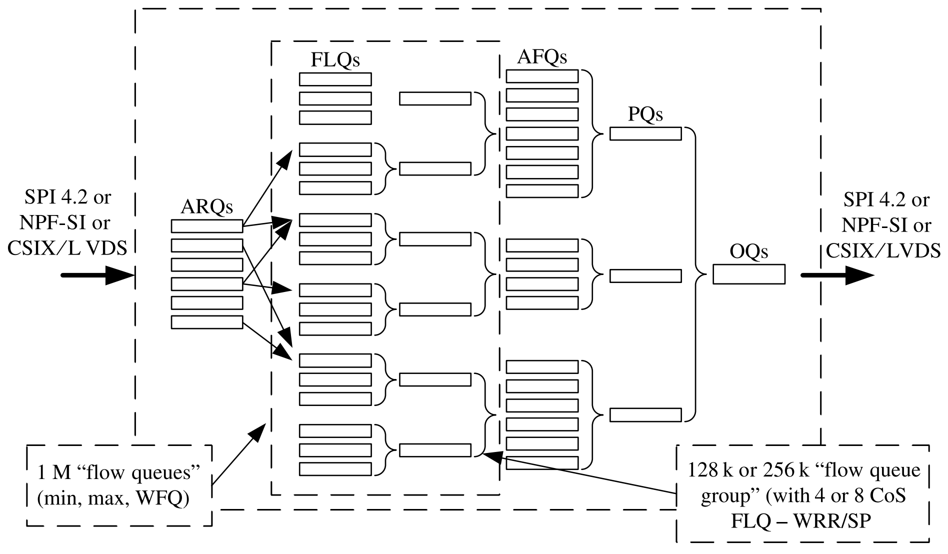

FLQs
AFQs
PQs
ARQs
OQs
SPI 4.2 or NPF-SI or CSIX/L VDS
SPI 4.2 or NPF-SI or CSIX/LVDS
1 M "flow queues" (min, max, WFQ)
128 k or 256 k "flow queue group" (with 4 or 8 CoS FLQ – WRR/SP

Figure 16.38 IDT TTM queue structure.

or AFQs if the TTM553 is absent. As mentioned in the previous part, the TTM552 starts packet accumulation when it receives an SOP cell. It moves the accumulated packet to FLQ or ARQ when it receives an EOP (end of packet) flag.

FLQs are supported when a TTM553 is connected. The TTM553 can support one million FLQs, corresponding to one million unique FINs. The FLQs are maintained by the TTM553 in its external SRAM. The TTM553 supports two modes: one-level FLQ mode and two-level FLQ mode in both the ingress and the egress directions. The one-level mode allows all one million FLQs to be scheduled directly into the AFQs. In two-level mode, four or eight FLQs are combined together and then mapped into 128 K or 256 K FLQ groups, respectively. The four or eight FLQs in each group can be scheduled before the scheduling of the FLQs. This mode adds immense flexibility.

The TTM553 standalone supports 4096 AFQs. One or more FLQs can be mapped into an AFQ. Arriving cells or packets are accumulated in ARQs before being moved into the ARQs when the TTM553 is absent. When the TTM553 is connected, any FLQs can be mapped into any AFQs based on configuration.

In the following stage, one or more AFQs can be mapped into one PQ. PQs can be used as physical ports or logical ports (virtual pipes) within an output queue. Ports can be mapped to any OQ and are "maximum rate shaped" when the TTM552 is used in the egress direction. While in the ingress direction, only one port can be mapped into one of the OQs.

The OQs correspond to the switch fabric's virtual output queues (VOQ) when the TTM552 transmits data to the switch fabric. The OQs can be viewed as physical destinations or channels when the TTM552 transmits data to a NPU or a framer. There are 1024 OQs available. However, in SPI-4.2 interface mode, only up to 16 OQs are supported for transmitting.

According to the queuing structure described above, when we use the TTM552 as a standalone chip, it manages traffic in three stages. And when the chip is connected to the TTM553, it manages traffic in 4–5 stages. The stages of traffic managing hierarchy can offer immense flexibility in configuring bandwidth on a physical channel, logical port, class, subscriber, or per-flow basis, as well as other applications. Another question is how the TTM552 cooperates with the TTM553 to finish the scheduling operations based on this queuing structure so that flows can share the limited bandwidth fairly. There are four levels of scheduling corresponding to the four stages of queuing structure except ARQs because they are dedicated to serve the reassembly and cannot be scheduled directly.

FLQs lie in the first level of hierarchy scheduling when the TTM553 is connected. These FLQs can be structured into 128 K groups of eight CoS FLQs or as 256 K groups of four CoS FLQs. FLQs can be individually configured with the following scheduling algorithms. The minimum rate algorithm is a guaranteed minimum rate of service for the FLQ. This algorithm becomes a constant bit rate (CBR)-like constant rate scheduler when other algorithms are disabled. Maximum rate algorithm controls the maximum rate for a FLQ. The available rate for a FLQ cannot exceed this value. Weight fair queuing (WFQ) is used for distributing the excess traffic when extra bandwidth is available. Strict priority and weighted round-robin (WRR) selection are only used for the FLQ within a group.

AFQs scheduling is the next level of hierarchy scheduling. There are 4096 aggregate-flow queues that can be configured with different scheduling algorithms. Same as the FLQ scheduling, the TTM552 provides two rate algorithms in this level: the minimum rate algorithm and the maximum rate algorithm. AFQ scheduling also supports dual-rate variable bit rate (VBR) shaping. This algorithm is performed using a peak rate (PCR), a maximum sustained rate (SCR) or a maximum burst size (MBS). WFQ is also used for distribution of excess traffic. If enabled, priority round-robin (PRR) or strict priority selection allows the distribution of excess bandwidth. An AFQ can be assigned to a priority level. Within the priority level, AFQs are selected on a round-robin basis.

At the third level, the TTM552 has 1024 PQs that can be individually configured with the following scheduling algorithms. PQs can be configured with a maximum rate for shaping traffic at a logical port level. This is only used in the egress direction. Ports can optionally be shaped using a calendar-based mechanism for TDM traffic with no jitter and a high-precision calendar-based shaping can be used with byte rate shaping.

At the fourth level, the TTM552 has 1024 OQs. After traffic has been serviced for departure, it goes to the OQ into which the port is mapped. In the egress direction, an OQ corresponds to a VOQ in a switch fabric, representing a destination port and CoS. In the ingress direction, an OQ is a physical port or a TDM channel of a framer. OQ selections can be round-robin or calendar-based.

Every level of scheduling can be configured to use one of the scheduling algorithms. The operational order of scheduling is opposite to the order described above. First the scheduler chooses which OQ to serve. Second the scheduler chooses a PQ within this OQ. The next is for the AFQ in the same way. Following the AFQ is the FLQ if the TTM553 is connected. Schedulers in different levels can work in parallel so that the pipelining can improve the performance of scheduling.

When scheduling is complete, the TTM552 sends data from the selected OQ to the Tx traffic interface. Before transmitting, however, there are a few modifications that the TTM552 can apply to the protocol headers of packets. If configured to do so, the TTM552 replaces the existing FIN with a new one. This is called FIN translation. The TTM552 also modifies or removes the packet header if necessary.

Multicast. The TTM552 supports two types of multicast. One is logical multicast and the other is spatial multicast. Logical multicast means making one or more copies of an incoming packet. The TTM552 applies this operation on the traffic received from the Rx traffic interface. Spatial multicast means sending one copy of a packet from one source port in one ingress line card to more destination ports in different egress line cards. This kind of operation is always done by a switch fabric. Logical multicast is done by the egress chips and spatial multicast by ingress chips. The bandwidth requirements of the switch fabric are minimized in this way.

Logical multicast is controlled by configuring multicast trees. A multicast tree is a data structure that describes how a packet is to be replicated. A tree has a root FIN, and one or more branches, identified by branch labels. The TTM552 performs logical multicast on a packet received on a root FIN by making one copy of the packet for each branch label in the multicast tree. Each resulting packet is identical to the original root packet with the FIN replaced by the branch FIN. The TTM552 supports up to 16,000 multicast trees. But there is an upper limit of 4096 on the total number of branch labels available to user. A branch can be reused by more than one multicast tree as shown in Figure 16.39. A multicast tree cannot have more than 4096 branches because of the upper limit.

The TTM552 classifies arriving traffic at the Rx traffic interface as either unicast or logical multicast. Unicast traffic bypasses the logical multicast engine. Logical multicast traffic is processed in the logical multicast engine. The engine first assembles PICs of a logical multicast root packet in one of the multicast ARQs. As each PIC of a root packet is received, the logical multicast engine performs congestion management checks. When a packet fails to pass the check, it is discarded. After these operations, the packet can be placed in one of the four multicast class queues (MCQs). Four MCQs provide four classes of service. The logical multicast engine selects packets from the four MCQs and replicates the root packet over their configurable multicast trees. Replicated packets behave as newly arriving unicast traffic to the later processing of TTM552.

Spatial multicasting can occur only in the ingress direction and is typically performed by a switch fabric. The ingress TTM552 assists by flagging the spatial multicast traffic going to the switch fabric and identifying the target line cards to which the traffic is to be sent. This information is passed in the form of a multicast ID or label.

Chip Set Features. The TTM552 can get a full 10 Gbps line rate performance. The chip can provide per-flow queuing and scheduling. It supports up to one million configured flows when equipped with the TTM553. Scheduling can be simultaneously performed on a large number of flows, each at an individual rate of fine granularity, and with many user-configurable scheduling algorithms allowing maximum flexibility and performance.

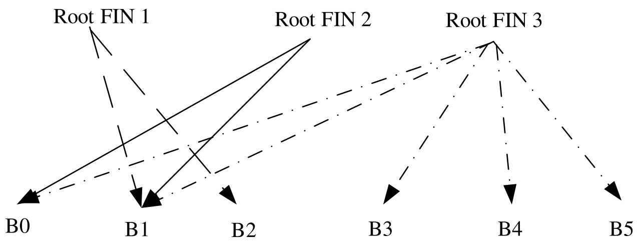


Figure 16.39 IDT TTM multicast trees.

Before queuing the packets or cells, the chip set executes sophisticated cell or packet admission control. Based on the information of thresholds, the chip set can support traditional discard algorithms such as maximum threshold, minimum threshold, and dynamic threshold in four different levels. If buffer memory approaches its limit because data arrival rate at the queue is greater than the departure rate, the TTM552 performs per-queue congestion management. The TTM552 can intelligently discard lower priority traffic as it arrives.

The TTM552 can also support spatial and logical multicast, satisfying the requirement of a 10 Gbps line rate.

16.3.4 Summary

The traffic management module plays an important role in high-performance routers. The traffic manager handles queuing and scheduling while it must consider the QoS requirements and different protocols. Because of the large number of flows managed and the very high line rate, it is difficult for the traffic manager inside a high-performance core router to complete these operations. And we cannot treat all the packets in a same way for efficiency reasons, so appropriate policing, shaping, buffering and scheduling algorithms must be developed within the TM.

From the description above, we have learned the structures, the implementations and the features of some commercial traffic management chip sets. They employ different structures and algorithms, providing different features and benefits for buffering and scheduling. They also provide configuration interfaces to users. The TM bridges the network processor and switch fabric, resolving resource congestion. They must make sure that different flows can share the memory and switch bandwidth adequately and pass through a router with lower latency.

16.4 SWITCHING FABRIC CHIPS

16.4.1 Overview

General switching principles, including various switching architectures and scheduling algorithms, have been discussed in Chapters 5 to 15. This section describes how to apply these theories to commercial applications, that is, the switching fabric chips. Generally speaking, most of today's switching fabrics beyond multi-gigabit adopt the crossbar with combined input and output queuing (CIOQ) structure.[2] In this architecture, VOQ structure is implemented in ingress ports and output queues are used in egress ports to deal with the speedup of the switching fabric. Vitesse's 872/882 and Agere's Pi40X/C are typical switching fabric chipsets using this architecture.

An alternative is to build a switching fabric using a single chip, such as Agere's Pi40SAX. The solution greatly reduces the design complexity and is often deployed in relatively lower throughput routers, for example, 40 Gbps.

In the core of the switch fabric, three different structures are used: the crossbar, the buffered crossbar, and the shared memory. The first two are space-division switching and the last one is time-division switching. Most commercial solutions support parallel and

[2]Share-memory is an exception as its single stage switch can only achieve 40 Gbps throughput.

multistage configurations of switching chips to achieve higher line speed and larger port numbers.

In the following section, three series of switching fabrics from Vitesse are presented. Section 16.4.3 describes the flexibly configurable cyclone switching fabric from AMCC. In Section 16.4.4, the detailed architecture of the IBM PowerPRS switching fabric is explained.[3] Finally, in Section 16.4.5, two different switching fabric solutions from Agere are described.

16.4.2 Switch Fabric Chip Set from Vitesse

Vitesse's [36] switch fabrics can be classified into three generations: CrossStream [37, 38], GigaStream [39], and TeraStream [40]. CrossStream, composed of a VSC870/VSC880 chip set, can perform variable-size packet switching. GigaStream is composed of VSC872/882 chip sets and cells are scheduled and switched internally. TeraStream (including two chips: VSC871 and VSC881) targets terabit core routers and uses a buffered crossbar architecture. These three chip sets represent different switching architectures and are described in detail as follows.

CrossStream: VSC870/880 Chip Set. For small capacity routers, parallel bus is usually used on their backplane to support modularity and simplicity. An example would be a PCI bus-based architecture. The parallel bus architecture, however, does not scale well. A new generation of large capacity routers use high-speed serial links (HSSL) for signal transmission between the line card and the switching card on the backplane. The VSC870 and VSC880 chip set are designed based on the new generation architecture.

Figure 16.40 shows the connections between VSC870 and VSC880. Vitesse designed the CrossStream chip set that performs serial data switching between the VSC870 and the VSC880. Each serial link is a 2.125 Gbps differential signal pair. Connection request (CRQ), data, acknowledgment (ACK), and other flow control information are all multiplexed into and transferred through these serial links. The VSC870 is built into the line cards whereas the VSC880 is built into switching cards.

The VSC870 also performs bit alignment and word alignment to synchronize both itself and the VSC880. The VSC880 acts as the master, generating the bit clock. The VSC870 recovers a bit clock from the receiving HSSL and then locks to it through a phase-lock-loop (PLL). In this way, the VSC870 and VSC880 are frequency-locked to one clock source provided by the VSC880. The word alignment is achieved by shifting the word clock one bit at a time until the VSC870 detects a proper predefined word pattern.

The VSC880 is a 16×16 synchronous serial crossbar switch with each high speed I/O link running at 2.125 Gbps (2.0 Gbps for data and 0.125 Gbps for control information). The aggregate data bandwidth is 32 Gbps. The internal block diagram of VSC880 is shown in Figure 16.41, which is a typical crossbar switch fabric. CMU (clock multiplex unit) is used to generate a high-speed bit clock. DRU (data recovery unit) is designed as a delay lock loop and remains phase locked to the incoming data stream. Port logic performs the necessary logic functions, such as separating CRQs and packets, inserting acknowledgments, and so on. Switch matrix is a crossbar switch fabric and performs data transferring from input ports to output ports. The scheduling algorithm is executed in the arbitration logic block.

[3]Note that this series has been purchased by AMCC.

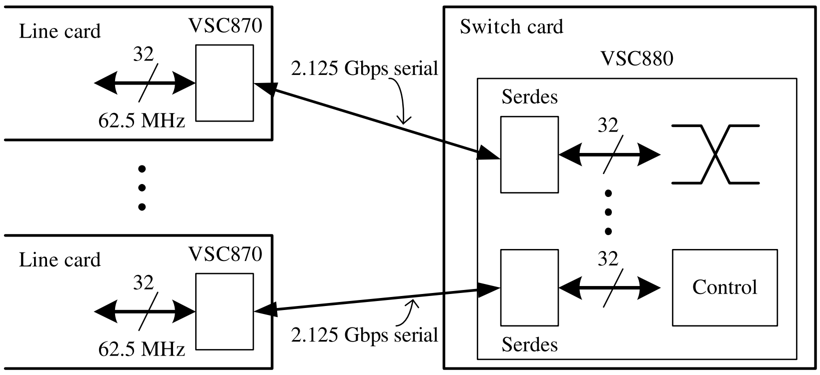


Figure 16.40 CrossStream chip set, VSC870 and VSC880, connected by SERDES (serializer/deserializer).

In packet switching, the variable-size packet scheduling between the VSC870 and VSC880 is executed in the following three steps:

Step 1. The VSC870 in each line card sends multiple CRQs to the VSC880.

Step 2. Arbitration is performed in the VSC880. CRQs reach the switch chip on each word clock and the arbitration process takes two word clock cycles: the first cycle looks for empty output ports and the second cycle makes arbitration using the round-robin scheduling. An ACK will be sent to the corresponding VSC870 if the CRQ is granted.

Step 3. The VSC870 sends out packets to the granted output port after receiving an ACK from the VSC880.

Figure 16.42 shows the format of CRQs. The CRQs are inserted at the end of each packet. The beginning of a variable-size packet is indicated with a special marking word named Header. To break a match between an input and an output port, a null CRQ is sent at the end of the current packet.

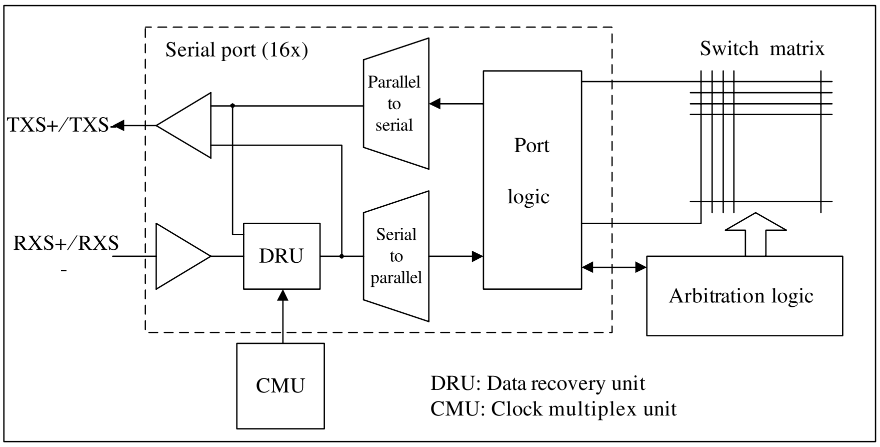


Figure 16.41 VSC880 block diagram.

10	Header	Start of packet
01	D0	
01	D1	
	• • •	
01	DN	
11	CRQ	End of packet
10	Header	Start of packet
01	D0	

Figure 16.42 VSC870 CRQ format.

To utilize the HSSL bandwidth efficiently, a CRQ may be inserted before the end of each packet. Figure 16.43 shows the format of sending a CRQ before the end of a packet. This scheme allows the central scheduler in the VSC880 to perform arbitration while transferring data. If the CRQ word is inserted into the current data packet D words before the end of the packet, ACK results will be known at the input port when the first word of the next data packet is ready for transmission. The value D is determined based on the round trip delay from the time the port submits a CRQ until an ACK is received.

The VSC880 can support a backpressure mechanism by providing a flow control channel. The flow control channel is time shared with the signaling between the switch chip and the transceiver. The VSC880 does not take any action to prevent congestion, and only passes the state information from the output port to the input port. During the flow control, the input port stops transferring packages to that port and prevents the FIFO on the receiver side from overflowing.

CrossStream has three modes to support multicast traffic. In the first mode, it copies the multicast packets and treats them as unicast packets. In the second mode, it treats a multicast packet as a group and it can be transferred only if all of its destination requests are granted. As a result, in some conditions, it may cause a dead lock as shown in Figure 16.44. Input port 1 sends CRQs to output ports 2 and 3, and input port 4 sends CRQs to output ports 3 and 4. Obviously the request for input ports 1 and 4 cannot be granted at the same time from output 3. This dead lock can be resolved using a counter. The request will be aborted if it is not granted during a long period and therefore this mode may cause performance

10	Header	Start of packet
01	D0	
	• •	
11	CRQ	
01	D(N-D)	D words before
	• •	
01	DN	End of packet
10	Header	Start of packet
01	D0	

Figure 16.43 CRQ ahead of the end of a packet.

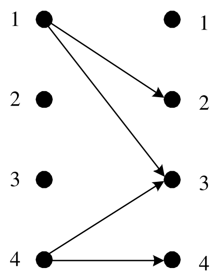


Figure 16.44 Example of dead lock for multicast traffic.

degradation. In the third mode, it transfers the multicast packet when parts of the requests are granted and the input port again sends a request for ungranted output ports. This mode requires the input port to buffer and reread the multicast packet until it has been transferred to all the destination ports.

The VSC870/VSC880 CrossStream can also support cell switching but the cell-mode scheduling algorithm is not integrated in the VSC880, which makes this architecture uncommon. The next generation GigaStream from Vitesse represents a typical cell switching architecture.

GigaStream: VSC872/882 Chip Set with Bufferless Crossbar

System Overview. Figure 16.45 shows how to use Vitesse's GigaStream chip set to construct a CIOQ-like switching solution. The switch system is composed of two chips: the queuing engine VSC872 and the crossbar switch VSC882, respectively. The VSC872 consists of two parts, the ingress part in the left of the figure and the egress part in the right. The ingress of VSC872 provides two standard 32-bit common switch interfaces (CSIX) in the front each running up to 166 Mbps × 32 = 5.312 Gbps connecting to TM or NP. The arriving frames are first sent to the frame parsing module where different kinds of frames, for example, unicast and multicast, are distinguished and sent to their corresponding queues for temporary storing.

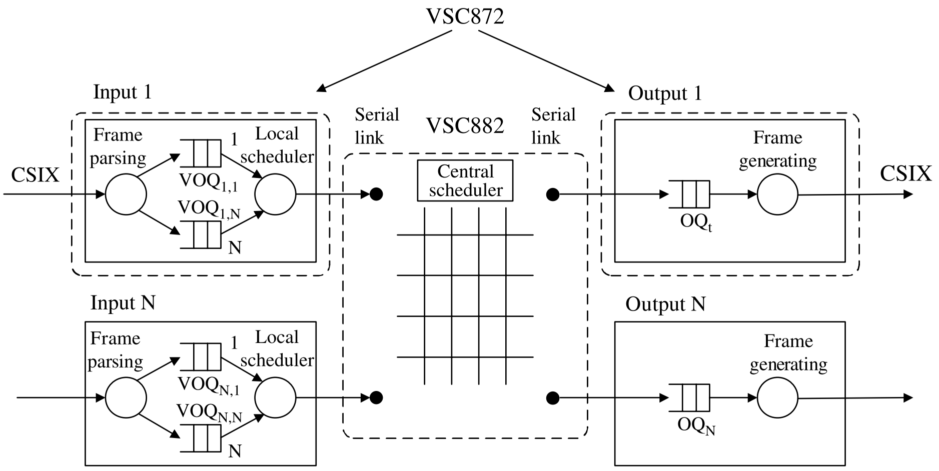


Figure 16.45 Architecture of GigaStream.

Because of the HOL blocking in the input queuing structure described in Chapter 5, the queuing engine VSC872 implements a VOQ instead. VSC872 provides eight serial links connecting the crossbar switch VSC882 at a speed up to 2.64384 Gbps data rate (2.5 Gbps for pure packets/cells transmission). With the execution of a sophisticated request–grant scheduling (will be illustrated later) between the local scheduler of VSC872 and the central scheduler of VSC882, the cells stored in the VOQ are scheduled and transferred to the crossbar switch VSC882 per cycle on the serial links. Since there are eight serial links in the direction to VSC882, up to eight cells can be transferred simultaneously in a same cycle to eight different VSC882s in parallel. However, each single cell is transmitted through the same crossbar switch in one cycle and there is no need to reorder at the egress of VSC872. Hence, the maximal transmission bandwidth between one VSC872 and one or more VSC882s is $8 \times 2.5\,\text{Gbps} = 20\,\text{Gbps}$, while the maximum bandwidth for the VSC872's receiving frames is $2 \times 5.312\,\text{Gbps} = 10.624\,\text{Gbps}$, providing a internal speedup of 2. Therefore, the egress of VSC872 uses an output buffer to accommodate the speedup. The cells leaving the output buffer are transformed into the correct frame pattern by the frame generating module and then sent out of the switching system.

Queuing Architecture and Scheduling Algorithm of GigaStream. Figure 16.46 shows the queuing system in the VSC872. The chip supports both unicast and multicast traffic. For unicast traffic, a strict high-priority (HP) and seven lower-priority (LP) queues of each VOQ are implemented. This queuing structure is replicated 16 times for its 16 output ports. For multicast traffic, the queuing architecture is similar to that of unicast traffic. When a multicast frame arrives, it is copied into all the VOQs belonging to the destination ports of the frame. The multicast VOQ is served similar to the unicast VOQ. In fact, the multicast

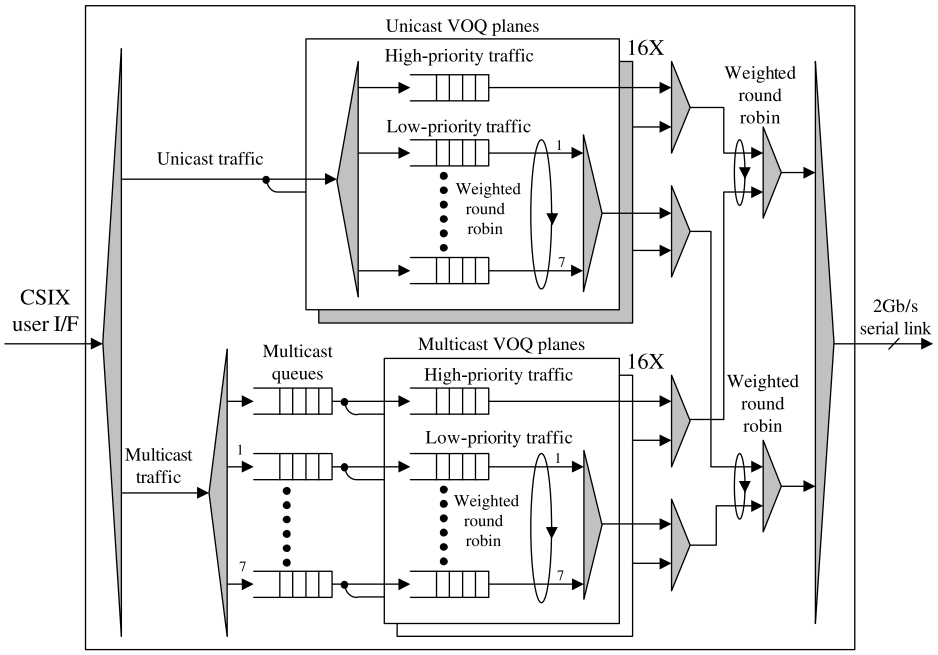


Figure 16.46 VSC872 queuing structure.

frame is stored in the ingress buffer as a single copy. The transfer from the multicast queue to the multicast VOQs involves only the copy of the corresponding pointers. When a multicast frame has been delivered to all of its destination ports, it is deleted from the buffer memory.

The GigaStream chip set implements three levels of scheduling. The first level generates CRQs for frames waiting in the VOQs. The second level generates a request ACK when a CRQ can be accommodated. The third level selects a queue whose frame will be sent when an ACK is received. Both the first level and the third level scheduling are executed in VSC872's local scheduler and the second level scheduling is performed in VSC882's central scheduler with an iterative matching process.

FIRST LEVEL SCHEDULING: CRQ GENERATION. For each scheduling cycle, a request can be made for one HP frame and one LP frame per VOQ. The VSC872 can generate a connection request for every non-empty HP VOQ. However, for a LP VOQ, it is allowed to send a request only when its connection cost is accumulated to be equal to or greater than the cost to gain the connection. At the end of each scheduling cycle, if current credit of any LP VOQ is below the connection cost, no request is sent and its credit is increased by a programmed weight value. If a LP VOQ credit is equal or above the connection cost, a CRQ is generated and the credit amount is debited by the connection cost at that moment.

SECOND LEVEL SCHEDULING: ACK GENERATION. The VSC882 has its own independent scheduling cycle and algorithm to generate request ACKs. This central scheduler looks at all the CRQs coming from all the local schedulers of every input port and goes through an iterative matching process similar to that of *i*SLIP to assign grants to requests in a fair manner. The grant process is first made only for HP CRQs. After four iterations for HP CRQs, LP CRQs are loaded and the grant process is executed for both LP CRQs and the remaining HP CRQs. In total, the VSC882 implements an algorithm that guarantees a minimum of 10 iterative matching cycles for the frame size of 40 bytes and more for a larger frame size. In theory, it requires 16 iterations to match 16 output ports to 16 input ports. This case happens only if all inputs request all the output ports and the round robin pointers are at the same location for all the output ports. By simulation and analysis, the scheduling algorithm can converge to a maximal match within four iteration steps for a 16×16 switch.

By allowing the HP CRQs first, it strictly reserves the available bandwidth for the HP traffic. This is not fair for LP traffic. After finishing the scheduling algorithm, each ACK gives the local scheduler permission to send a frame to one specific switch output.

THIRD LEVEL SCHEDULING: QUEUE GENERATION. If the local scheduler receives an ACK from the central scheduler, it must find the best frame to send to the granted switch output from its corresponding VOQs. If a HP frame is waiting, the local scheduler always selects the HP VOQ even if it is not waiting when the original CRQ is generated. If no HP frame is waiting, the "credit" is used to select a LP traffic. The credit added at the end of each scheduling cycle is the programmed weight value of the class and the connection cost is always the maximum of all class weights. "Credit" is in fact used to implement a weighted round-robin manner.

If both unicast and multicast have waiting frames to be sent, the simple weighting round-robin scheduling algorithm is used to allocate bandwidth between unicast and multicast traffic.

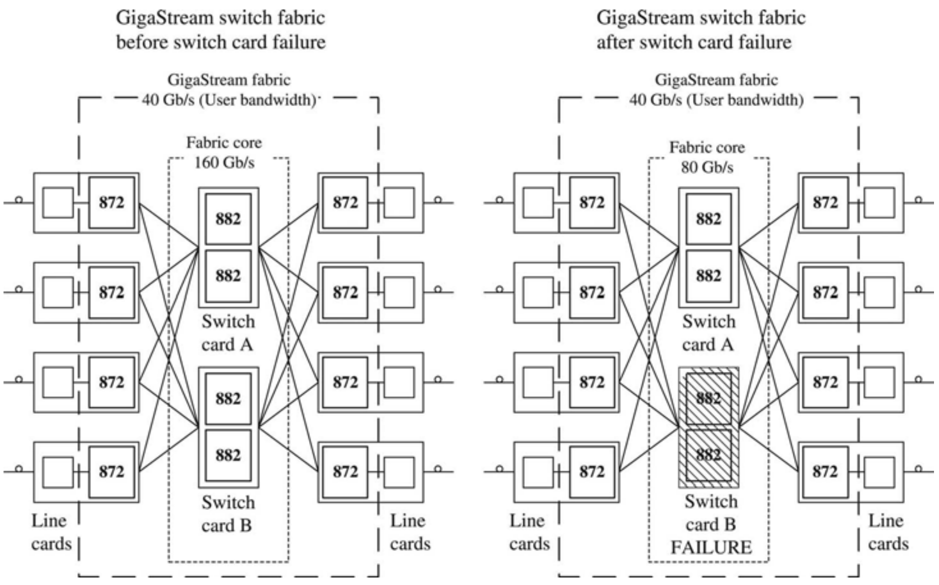

GigaStream switch fabric before switch card failure
GigaStream fabric 40 Gb/s (User bandwidth)
Fabric core 160 Gb/s
872
882
Switch card A
Switch card B
Line cards
GigaStream switch fabric after switch card failure
GigaStream fabric 40 Gb/s (User bandwidth)
Fabric core 80 Gb/s
Switch card B FAILURE

Figure 16.47 GigaStream active redundancy.

Active Redundancy in GigaStream. Figure 16.47 shows a medium scale switching system in which each VSC872 is connected to four VSC882s. Two VSC882s are in switch card A and the other two are in switch card B. The switch fabric on the left shows normal operation, where both the switch cards are working, offering a 4×40 Gbps $= 160$ Gbps switching capacity. When one of the switch cards fails, as shown in the figure on the right, the serial links between the failed switch card and the connecting VSC872s detect link errors automatically. The VSC872s have to load-balance the traffic formerly on the failed switch to the working switch within a very short time period. Although the performance may worsen, the switching system remains working at half the throughput capacity.

TeraStream: VSC871/881 Chip Set with Buffered Crossbar. Vitesse's TeraStream chip set presents another type of CIOQ switch: buffered crossbar. Figure 16.48 shows a typical architecture of TeraStream, which comprises two types of chips: VSC871 and VSC881. The architecture extends the dual chip architecture of Vitesse's switching fabric.

Compared to Vitesse's former products, GigaStream and CrossStream, the new architecture gains three key advantages to achieve Terabit switching. Firstly, as shown in Figure 16.48, the 24 VSC871s are connected by 13 VSC881s in three stages. This multistage configuration of VSC881 guarantees the design of mass input ports. Secondly, through link bundling and slicing, several serial links between VSC871 and VSC881 or between neighbor VSC881s can be congregated to a single link with the aggregated bandwidth. Therefore, much higher line speed on input port can be achieved. Finally, the VSC881 implements the advanced switching structure of a buffered crossbar.

Multistage Configuration. The TeraStream supports both single-stage and multistage configurations. However, a multistage topology is required to achieve terabit switching capacity. A variety of multistage topologies can be constructed using the VSC881. Figure 16.49 shows some possible configurations. Historically, the Clos and Banyan architectures are very well

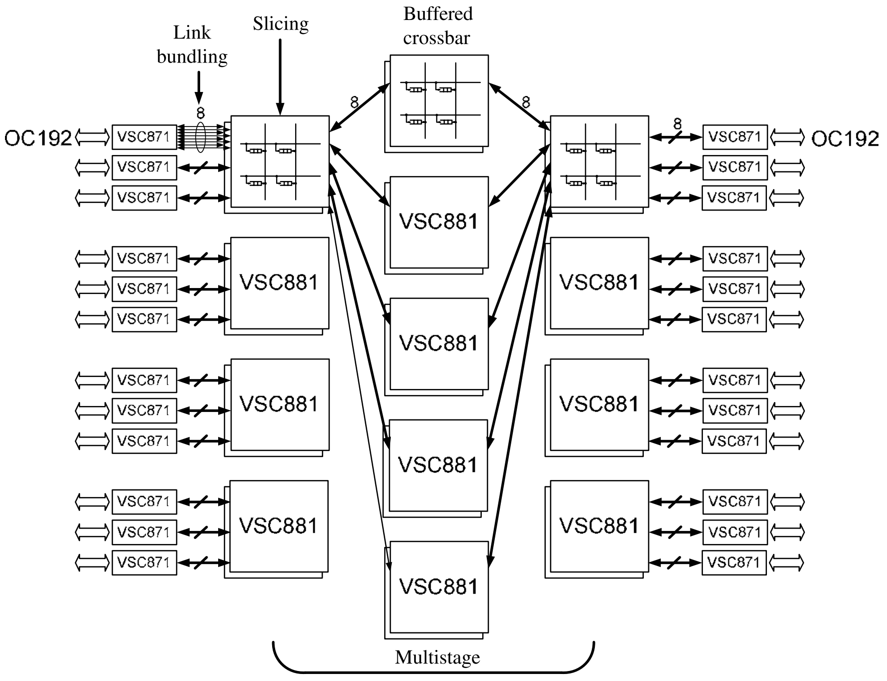


Figure 16.48 Typical architecture of TeraStream.

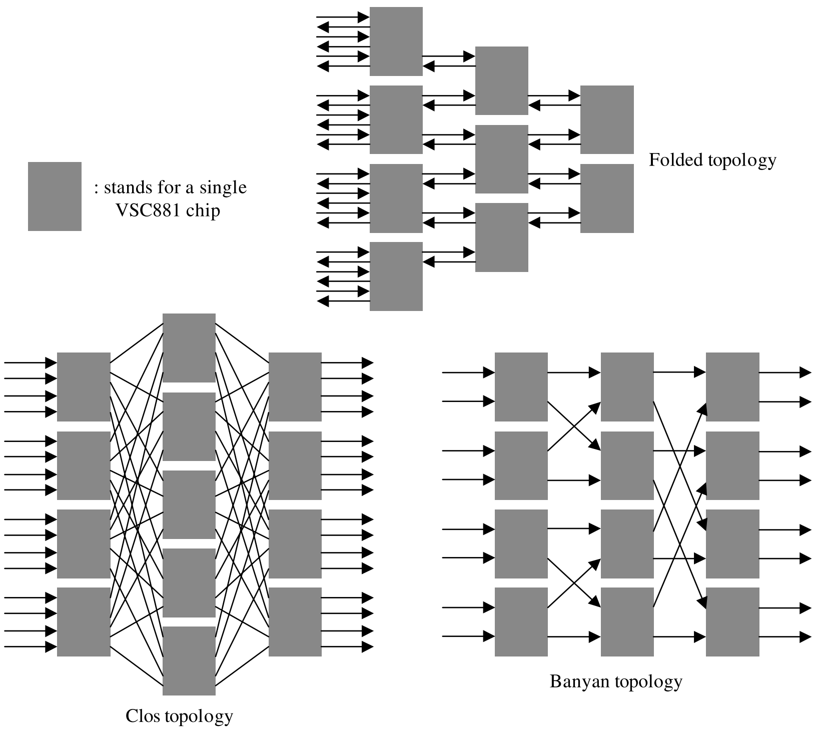


Figure 16.49 Possible multistage configurations of TeraStream.

known. They are “pass-through”, meaning that all connections have to go through all the stages. Another topology is the folded architecture, in which the requested connectivity determines the number of stages. Therefore, the folded architecture provides a highly scalable system with fewer devices.

Link Bundling and Slicing. Compared to the GigaStream architecture, there is no significant advantage in serial link technology. However, through slicing and link bundling, the internal forwarding speed per logical pipe is increased to 20 Gbps, eight times that of GigaStream. As shown in Figure 16.50, four VSC881 switch slices make up a single logical core switch element. One of the slices is assigned to be the master slice that controls the remaining slave slices. Master/slave synchronization is performed via a dedicated daisy chained broadcast bus where three high speed bi-directional links are used for synchronization of the incoming data flows and another three for synchronization of the outgoing data flows. All scheduling decisions are performed in the master slice and the slave slices merely act as read/write data storage buffers.

When transferring from VSC871 to VSC881, the internal cells of the fabric are split into several segments of equal size and are forwarded to the switch slices in the same group. In Figure 16.50, an 80-byte cell is divided into four 20-byte segments and they are reconstructed at the egress VSC871 after the switching stage.

Besides slicing, the TeraStream also supports link bundling. In Figure 16.50, two groups of the four slicing links are bundled together to achieve higher forwarding speed of a single logical link. There is no further segmentation involved in link bundling. Two independent cells of the same queue are transported in parallel across the logical pipe during any given time slot.

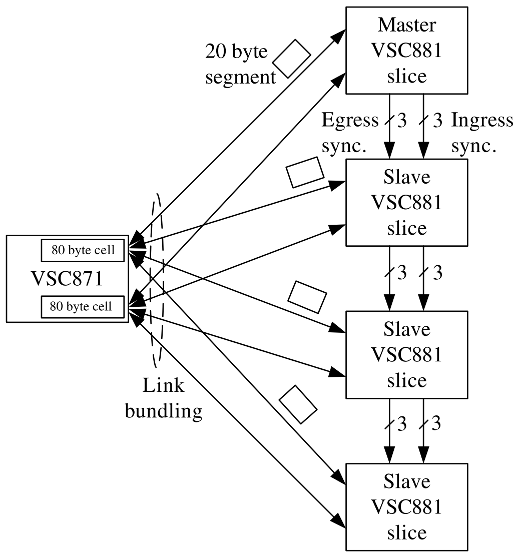


Figure 16.50 Link bundling and slicing in TeraStream.

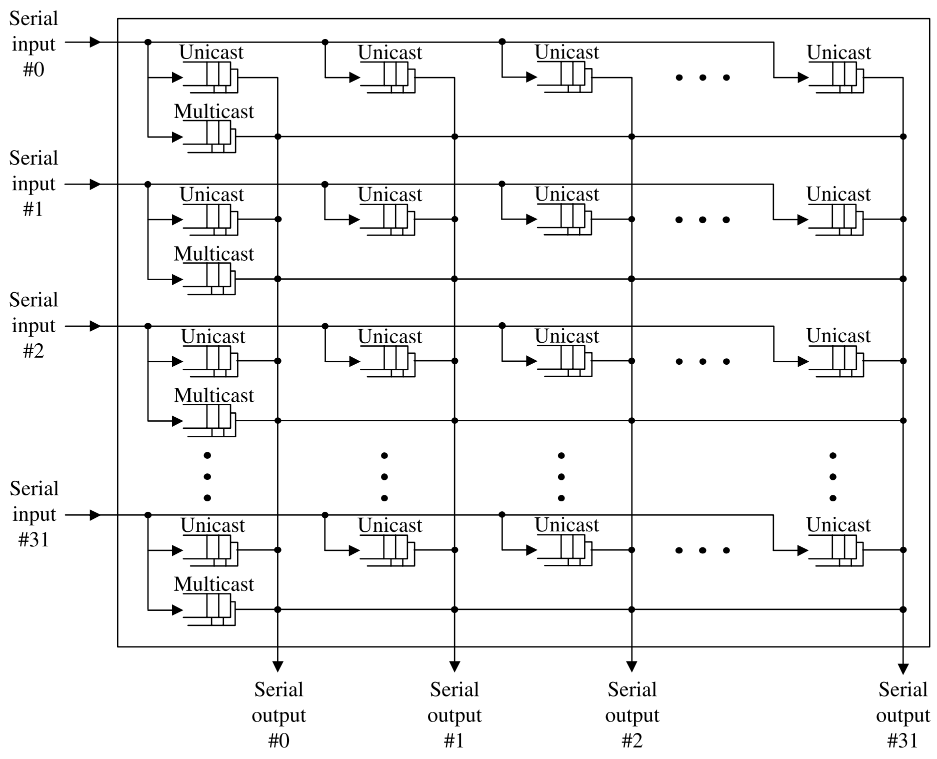


Figure 16.51 Crosspoint buffer structure of VSC881.

The maximal configuration of link bundling and slicing is dual link bundling plus quad slicing or quad link bundling plus dual slicing. Therefore, the maximum logical pipe bandwidth is 20 Gbps.

Buffered Crossbar. The internal structure of VSC881 is different from the traditional crossbar switch. It deploys a buffer in each crosspoint and is called a buffered crossbar structure. As in Figure 16.51, a unicast FIFO structure is maintained per cross point and a multicast FIFO structure is maintained per input. Each of these FIFO structures supports two priority levels: HP and LP. Each cross point unicast FIFO and input multicast FIFO support a FIFO depth up to eight cells.

The crosspoint buffered crossbar structure reduces (or avoids) the output contention. They allow the inputs to send cells to an output irrespective of simultaneous cell transfer to the same output.

16.4.3 Switch Fabric Chip Set from AMCC

AMCC constructs its Cyclone switching fabric [41] based on cost effectiveness. As shown in Figure 16.52, the switching fabric is not composed of a queuing engine (QE) and switch core (SW) anymore. Instead, five classes of chips are implemented together to realize packet queuing, scheduling, and switching. They are memory subsystem (MS), priority queue (PQ), arbiter (AR), crossbar (XB), and earliest deadline first queue (EDFQ). MS buffers cells, notifies PQ, and manages TM flow control credits. PQ bids AR for path to transmit cells of

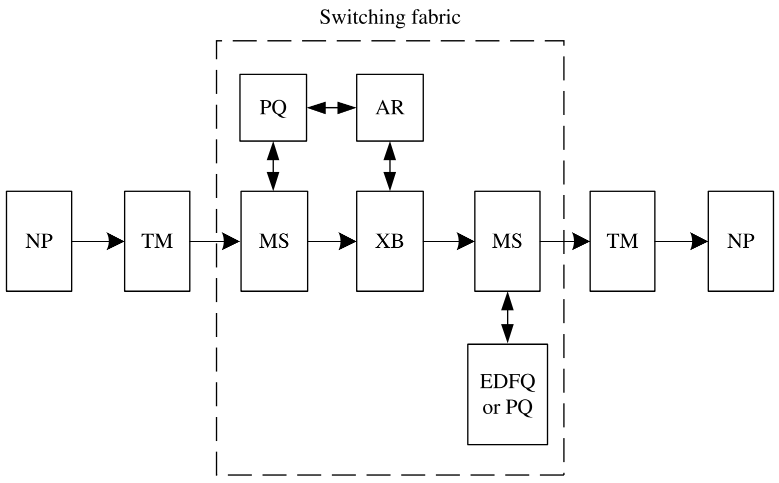


Figure 16.52 Cyclone switching fabric architecture.

same packet and manages fabric flow control credits. AR grants PQ a path and configures XB. XB passes all cells belonging to one packet in a single link. EDFQ releases cells based on WFQ. This design method brings significant flexibility, so that an engineer can build his/her own switching fabric at minimum cost. In addition, the architecture is scalable, as extra chips can be added easily to support higher throughput or better QoS performance.

The switching fabric supports up to 40 Gbps (4 × OC192) port rate, with a maximum of 32 ports. Figure 16.53 shows a typical configuration that consists of 32 20 Gbps line cards and four 320 Gbps switch cards.

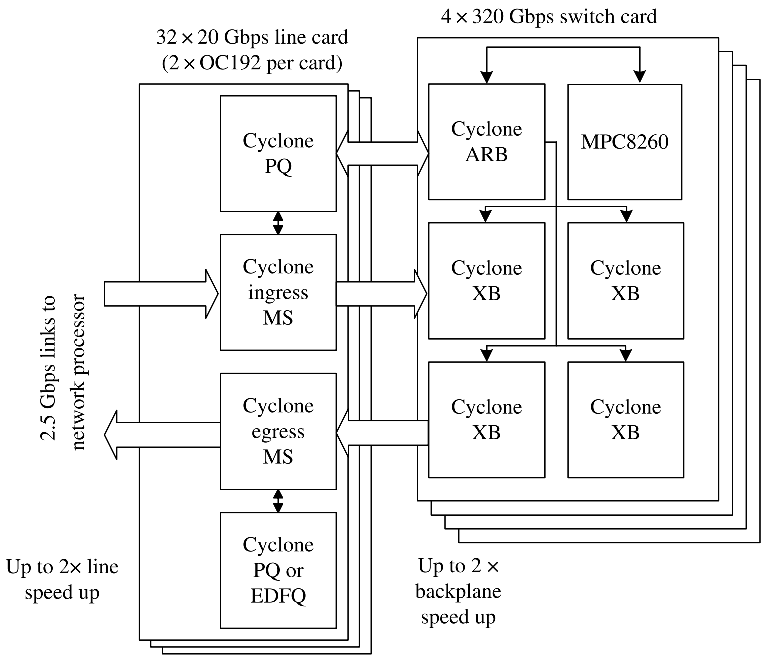


Figure 16.53 Cyclone switching fabric configuration.

Memory Subsystem. The cyclone memory subsystem block diagram is shown in Figure 16.54. Memory subsystem consists of a memory buffer (cell storage), cell/free cell list, memory management module, and serial interfaces. At the ingress port the cell sent to the memory subsystem is first de-serialized to a parallel data pattern and then stored in the memory buffer. A list of occupied cell storage and free cell storage is maintained by the memory management module and updated when any cell is stored or scheduled out. Up to 32,000 cells can be buffered in one MS. The information of new packet arrival is also passed through the queue interface to the PQ, where the packet ID is stored as output port and priority. Once a packet in PQ is granted by the AR, cells of this packet are read out from the memory buffer, serialized and transferred to the backplane continuously. In egress, cells switched by the crossbar are sent to the egress MS and stored again. Either PQ or EDFQ can be deployed at the egress to schedule cells out of the MS.

Priority Queue. The PQ can be located at either the ingress ports or the egress ports. On the ingress side, the PQ buffers packet ID in a VOQ structure. Each output consists of eight CoS. Up to 32 output ports are supported per channel with a maximum of four channels per PQ. The PQ implements the deficit round robin (DRR), weighted round robin (WRR), modified deficit round robin (MDRR), and modified weighted round robin (MWRR) algorithms to generate bids to the AR via its integrated serial links. The PQ makes four bids every 32 ns to its attached ARs. On the egress side, the PQ is used to schedule packets out of the egress MS. The PQ is also responsible for flow control.

Arbiter. The AR is located in the switching card configuring the XB of the same card. Up to five XBs are deployed per switching card. The AR receives bids from the PQ of every input port cards and does input–output matching using a round robin, maximal matching scheduling algorithm. The matching results are sent back through serial links to the corresponding PQ.

Besides best-effort switching, the AR also contains four TDM reservation tables to support TDM switching. Figure 16.55 shows a simple switch card architecture for optical or electrical backplanes with one AR per card.

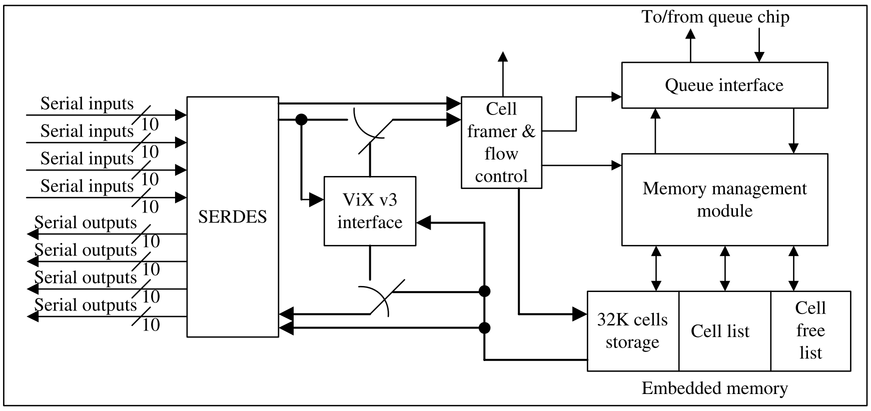


Figure 16.54 Cyclone memory subsystem block diagram.

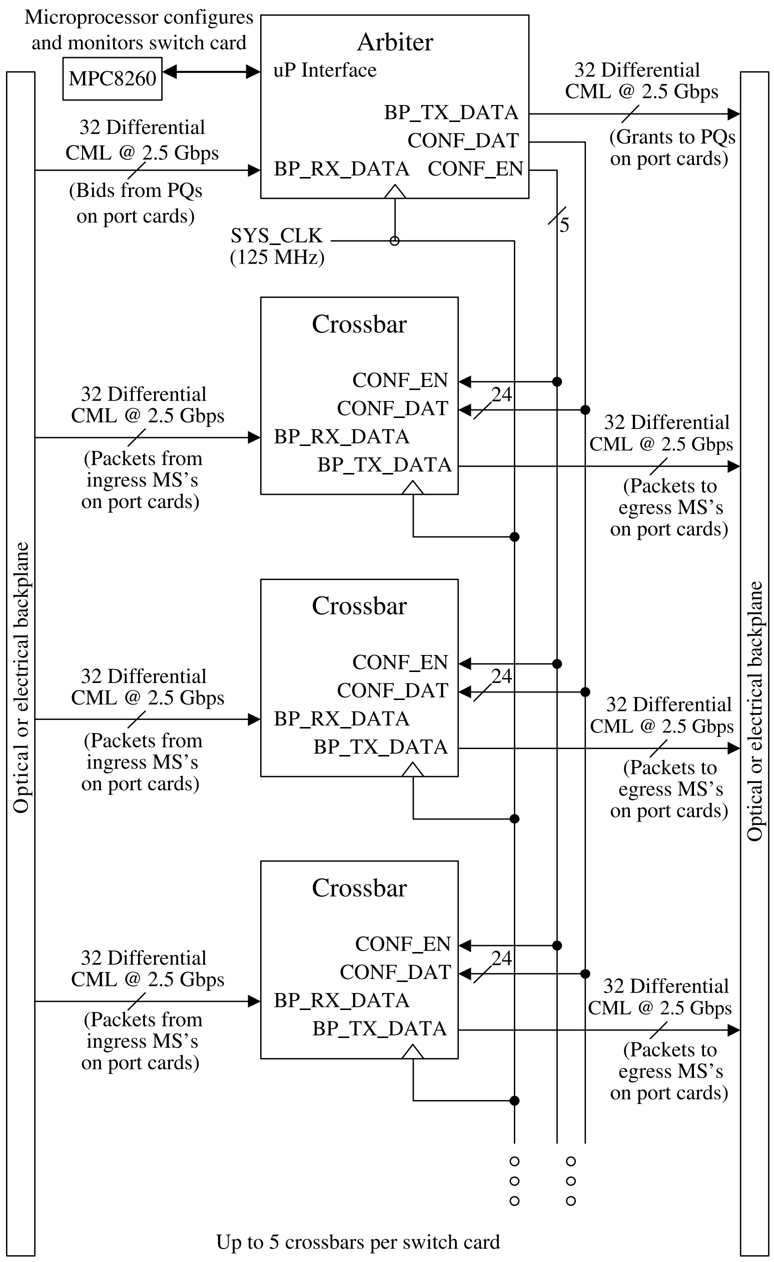

Microprocessor configures and monitors switch card
MPC8260
Arbiter
uP Interface
BP_TX_DATA
CONF_DAT
BP_RX_DATA
CONF_EN
32 Differential CML @ 2.5 Gbps
(Bids from PQs on port cards)
32 Differential CML @ 2.5 Gbps
(Grants to PQs on port cards)
SYS_CLK (125 MHz)
5
Crossbar
CONF_EN
CONF_DAT
24
BP_RX_DATA
BP_TX_DATA
32 Differential CML @ 2.5 Gbps
(Packets from ingress MS's on port cards)
32 Differential CML @ 2.5 Gbps
(Packets to egress MS's on port cards)
Optical or electrical backplane
Optical or electrical backplane
Up to 5 crossbars per switch card

Figure 16.55 Cyclone switch card.

Crossbar. The XB of Cyclone switching fabric is a typical synchronous 64 Gbps (data switching capacity) 32 × 32 crossbar with 2.5 Gbps line rate in each port. It is controlled and configured by the AR. When an output is not matched to any input ports, idle cells are generated for them.

Earliest Deadline First Queue. The EDFQ is implemented at the egress port. It performs a WFQ algorithm to schedule packets out of egress MS. In each EDFQ, 16 schedulers are available so that up to four OC-192 channels or 16 OC-48 sub-channels can be supported by one EDFQ. The scheduling is based on flow ID, CoS, and packet length. The EDFQ also takes charge of flow control at the egress port.

16.4.4 Switch Fabric Chip Set from IBM (now of AMCC)

The following section explains IBM's switch chipset, which is quite different from switching architectures described in the previous sections. The overall architecture of the IBM third generation switching fabric [42] is shown in Figure 16.56. The IBM PowerPRS Q-64G is a kind of packet routing switch chip providing up to 512 Gbps aggregate throughput while the IBM PowerPRS C192 is a companion device to the Q-64G functioning as the switch core access layer between the protocol engine's OC-48 or OC-192 CSIX interfaces and the switch core. The switching fabric follows the popular topology "ingress queuing engine (IQE)–switch core (SW)–egress queuing engine (EQE)," with PowerPRS C192 as IQE and EQE and PowerPRS Q-64G as SW.

However, the switching fabric is novel in the design of 32 × 32 switch core Q-64G. It implements the shared memory mechanism and constructs the 32 × 32 switch from four 16 × 16 sub-switches mounted in port expansion by two. Figure 16.57 depicts the internal structure of the shared memory switch Q-64G. It is mainly composed of 32 input

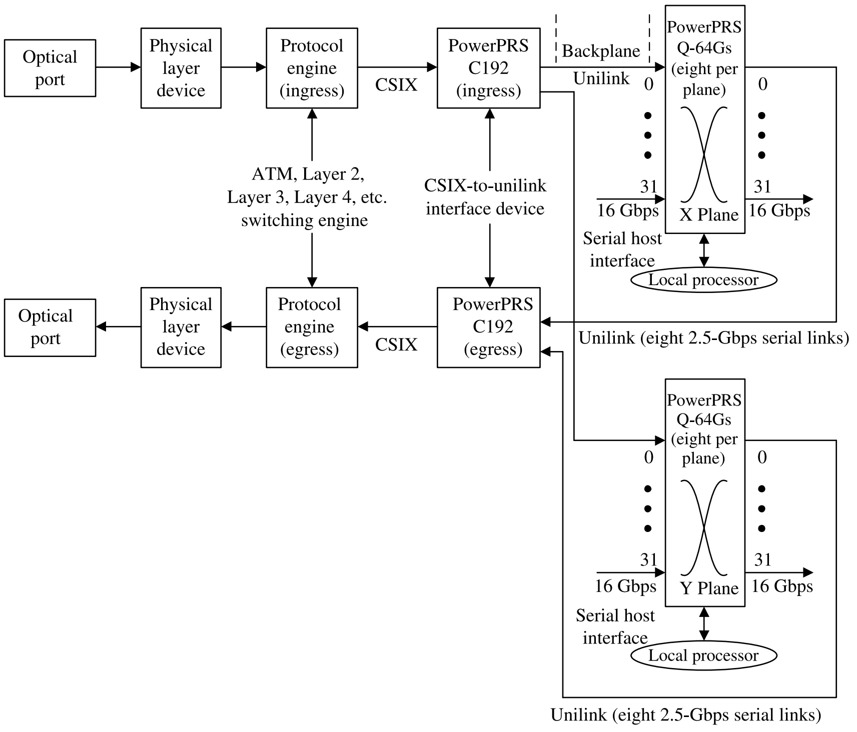


Figure 16.56 System view of the switching fabric with Q64-G (configured with redundant switch planes).

controllers, 32 output controllers, four sub-switch elements, and other control sections (sequencer, output queue scheduler, credit table). Each port has one input controller and one output controller. Each sub-switch is connected to half of the input controllers and half of the output controllers so that the ingress cell flows are distributed uniformly across the four sub-switches. For example, sub-switch A forwards those packets that come from input port 0–15 but head for output port 0–15.

Each component of the Q-64G switch core is described in detail using the packet forwarding path in the following.

Sequencer. As shown in Figure 16.57, a sequencer connects to all of the input controllers and the output controllers. It manages the access of these controllers to the shared memory of every sub-switch.

The sequencer makes scheduling in a time-division multiplexing (TDM) manner. That is, each time slot it grants the shared memory access to two input controllers and two output controllers, with one in 0–15 ports and the other in 16–31 ports. In the next time slot, it grants two other input and output ports. Finally, the sequencer visits every port in one cycle.

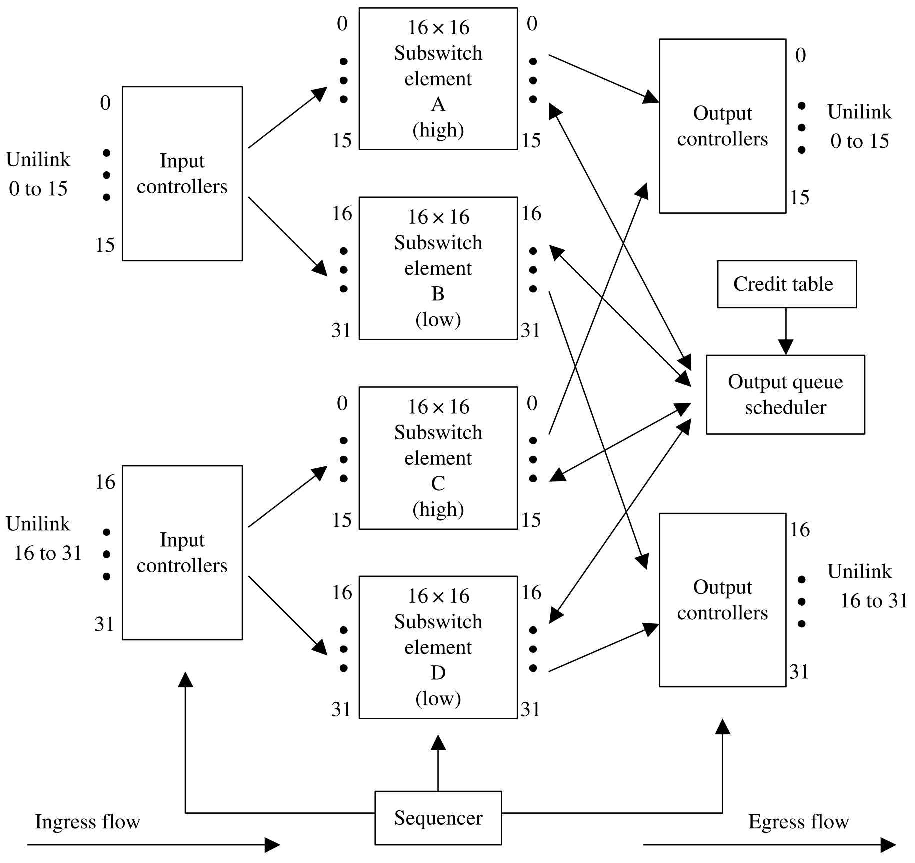


Figure 16.57 Q64-G block diagram.

Input Controller. When an input controller is granted by the sequencer, it forwards two packets at a time to the shared memory with one on the high channel (0–15 output ports) and the other on the low channel (16–31 output ports). As shown in Figure 16.57, once granted by the sequencer, the input controller of port 1 sends two packets to the shared memory of sub-switch A and sub-switch B, respectively. However, the data actually sent is the cells with equal length segmented from the packet.

The address of the destined shared memory is provided by the address manager of the corresponding sub-switch. The input controller also forwards the priority of the packet, destined memory address and the packet destination to the output queue access manager inside the sub-switch, and this address is stored in an output queue constructed one per output per priority. It is illustrated in Figure 16.58.

16 × 16 Sub-Switch. The internal block of each sub-switch is shown in Figure 16.58. The key feature of this switch architecture is that it does not switch cells by connecting crosspoints anymore. Instead, it integrates 2048 10-byte rows for unicast traffic and 1024 10-byte rows for multicast traffic of shared memory per sub-switch and forwards cells by writing and reading the shared memory. The writing of the memory (cell flow in the ingress direction) is controlled by the external input controllers per input port. The reading of the memory (cell flow in the egress direction) is controlled by the output controllers per output port. Simultaneously with the cell ingress into shared memory, the address of the cell in memory is stored in the corresponding output queue, which constructs one per output per priority. Also, an external output queue scheduler is connected to every output queue of each sub-switch, gathering the occupation status of the output queue. It decides from which

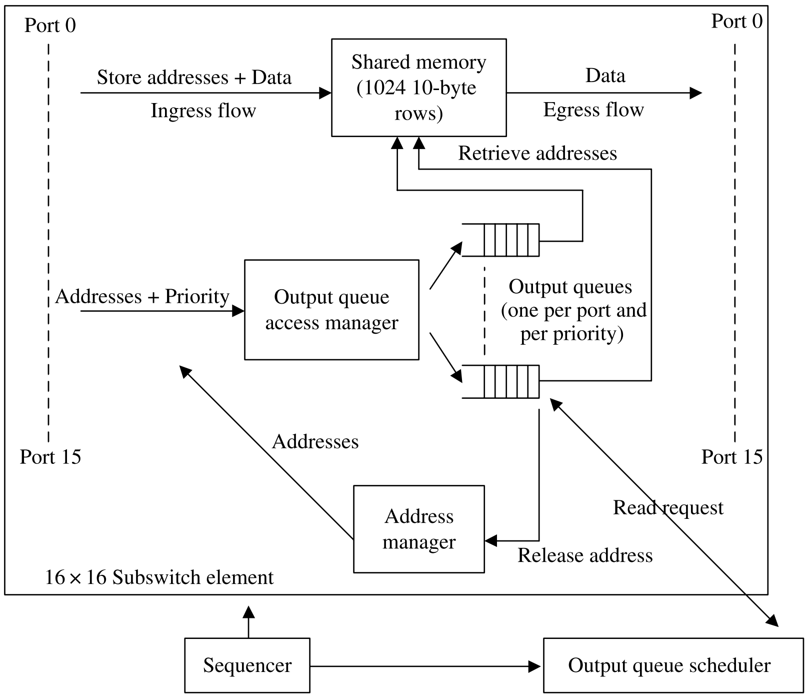


Figure 16.58 16 × 16 sub-switch element block diagram.

output queue the shared memory should obtain the egress cell address; that is, it controls the output packet selection when the output controller is granted by the sequencer. Moreover, an address manager is used to maintain the status of shared memory occupation and provide empty shared memory entry for external input controller.

Output Controller. Similar to the input controller, when an output controller is granted by the sequencer, it retrieves two packets at a time from the shared memory, one from the high channel (0–15 input ports) and the other from the low channel (16–31 input ports). As shown in the Figure 16.57, once granted by the sequencer, the output controller of port 1 receives two packets from the shared memory of sub-switch A and sub-switch C, respectively. The output packet selection is done in the output queue scheduler, and the output controller just reads data from the shared memory in the given address. After inserting some flow control information into the packet header, the output controller forwards the packet to the physical interface for serialization, and then transmits it to the egress QE. Also, the data actually received by the next egress QE is the cells with equal length segmented from the packet.

Output Queue Scheduler. When the output controller is granted by the sequencer, several packets can be chosen from the output queue in the corresponding port for transmitting, unlike the input controller where only the ingress packet at that cycle can be scheduled in.

An output queue scheduler has to be deployed to select packets to be scheduled out and notify the output queue to provide correct egress packet address of the shared memory. The output queue scheduler chooses egress packets by priority. However, through the implementation of a credit table, it can guarantee fixed bandwidth for every queue as well.

Each input (output) port of the Q-64G switch core is connected to the ingress (egress) QE using a serial link, which operates at 2.5 Gbps, with 8b/10b encoding for link synchronization and supervision. With a single 32×32 Q-64G chip, 2 Gbps pure data line rate and a total of 64 Gbps switching capacity can be reached. To expand link speed, multiple chips can be configured in parallel to form a larger switch core with higher port throughput. A maximum of eight chips can be reached to support 32 ports with 16 Gbps data bandwidth per port, resulting in a total of 512 Gbps throughput. Figure 16.59 shows the four chips' configuration, where one chip is set as master and the others as slaves. Only the master device performs packet routing and queuing and all the other slave devices are synchronized to the master device through the packet synchronization signals that synchronize the sequencer, shared memory addresses and output queue scheduler.

In conclusion, the Q-64G switch core is quite different from previous switching architectures. It deploys central shared memory to buffer packets when output port conflicts happen. The architecture benefits a lot in completely avoiding input–output port matching, which requires much computing complexity and much transmission bandwidth to send request and grant. At the same time, it permits an efficient handling of multicast and broadcast traffic through the replication of a single copy of a cell through the corresponding output ports as soon as the individual ports become available. However, the central shared memory architecture has rarely been adopted before because of its obvious disadvantage that the shared memory should operate at a speed of $2N$ times the line rate, where N is the port number. It is impossible to provide such a high bandwidth using external central shared memory. Fortunately, the quick development of ASIC manufacturing enables the building of a buffer inside chips. To reduce the necessary shared memory bandwidth, the Q-64G switch core also introduces parallel processing inside the chip.

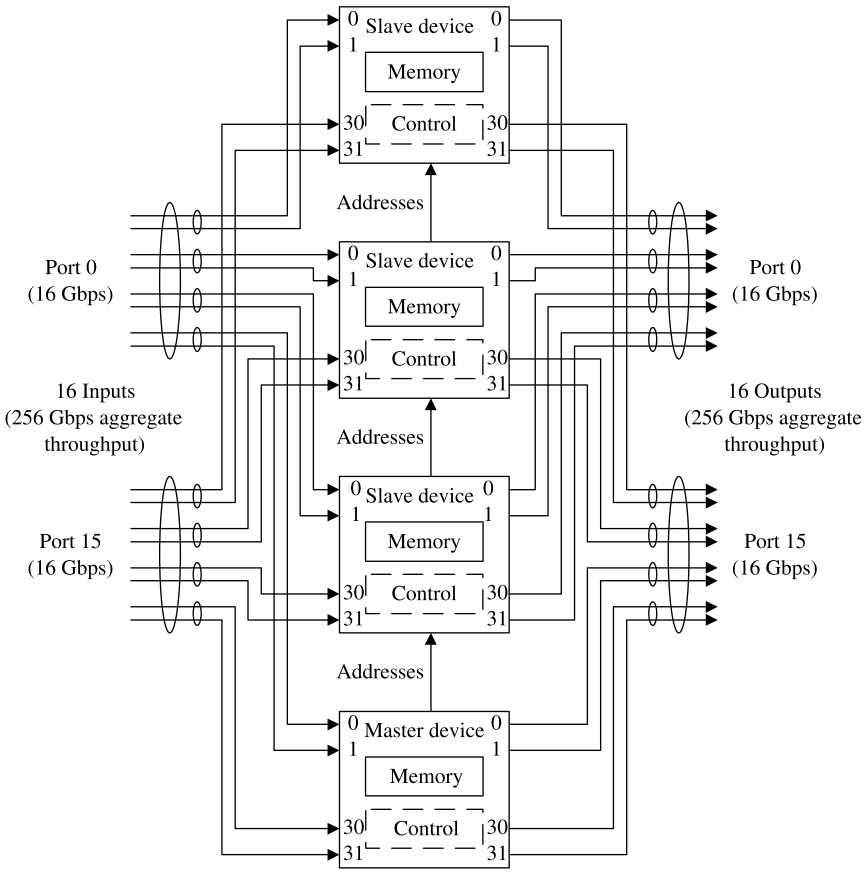


Figure 16.59 Parallel Q64-G configuration (speed expansion to 256 Gbps).

16.4.5 Switch Fabric Chip Set from Agere

The switching fabric solutions from Agere contain two main architectures. The first is built using Pi40X/C [43, 44], with similar topology to the popular three-stage structure: IQE–SW–EQE structure. The second is a single chip solution implementing Agere's new shared-memory stand-alone switch chip Pi40SAX [45]. The first topology can provide large switching capacity beyond terabits/sec, while the second provides a solution that is easy to build and control.

Pi40X/Pi40C Chip Set for Terabit Switching. Figure 16.60 shows a typical switching fabric constructed using Pi40X/C. Compared to other common switching fabrics, the Pi40 series differ in two aspects. First, as shown in Figure 16.61, the IQE and EQE are located in separate Pi40X chips and an extra communication channel is built between them to communicate scheduling information. Second, more serial links are implemented in both the QE Pi40X and the XB Pi40C providing much higher bandwidth. These serial links can also be bundled freely to aggregate different line-speed links. This is shown in Figure 16.62. The Pi40X has 32 2.5 Gbps simplex serial links in line side, which can be bundled to one

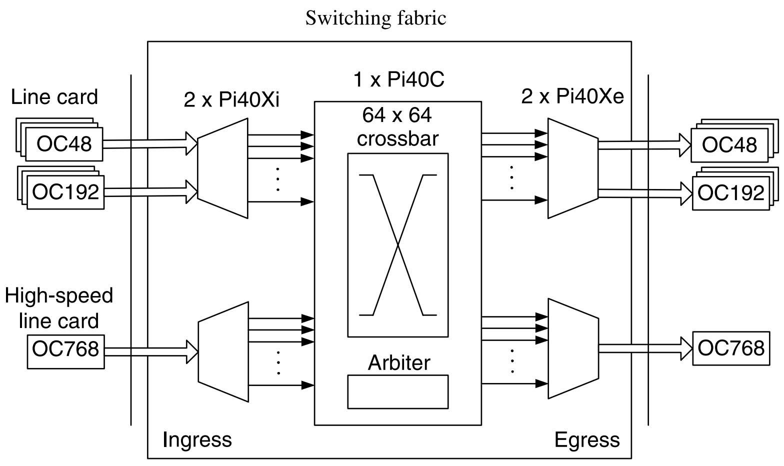


Figure 16.60 Agere's switching fabric with Pi40X and Pi40C chips.

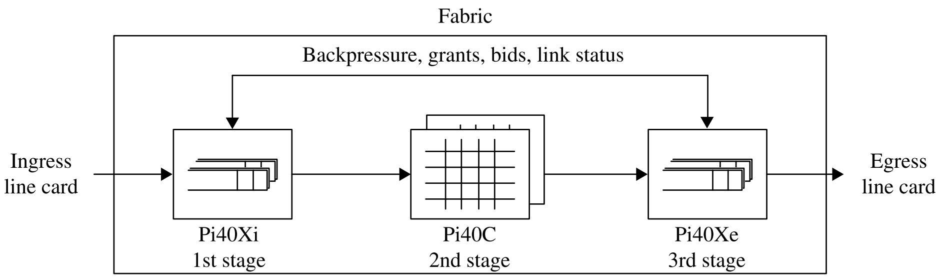


Figure 16.61 Pi40X ingress–egress pair communication.

OC768 channel, four OC192 channels, 16 OC48 channels, 32 OC12 channels[4], or any combination of them. The Pi40C has 64 2.5 Gbps duplex serial links providing 160 Gbps aggregate switching capacity in a single chip. Therefore, a few parallel Pi40C can support terabit switching.

Flexible Configuration of Pi40X/C. Since the serial links between Pi40X and Pi40C can be bundled freely, the connection patterns between them can be arbitrary. The fundamental topology is a three-stage Clos network as shown in Figure 16.63. The connections between stage 1 and 2, as well as between stage 2 and 3 are both in a fully connected pattern. Figure 16.63 shows that the switching fabric supports 1 : 1 redundancy in all the stages.

A minimal configuration of Pi40X/C is shown in Figure 16.64. Thirty-two egress serial links of Pi40X are bundled together to form a 80 Gbps virtual channel. When less serial links are bundled together more virtual channels between Pi40X and Pi40C are available. More Pi40Cs can be used to achieve higher switching capacities. A maximal configuration is shown in Figure 16.65, which provides a 2.5 Tbps throughput.

[4]Since there are only 32 serial links.

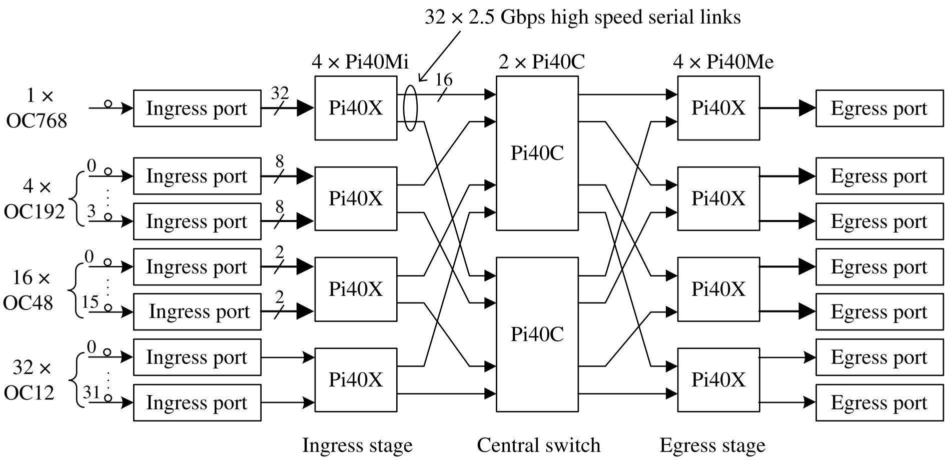


Figure 16.62 Link bundling of Pi40X and Pi40C.

Queuing and Scheduling in Pi40X/C. Figure 16.66 shows the queuing and scheduling mechanism in Pi40X/C switching fabric. Note that multicast queues are not depicted there.

In the ingress QE Pi40X, cells are buffered in the internal shared memory. The memory is organized into 1024 unicast routing queues and 64 multicast routing queues. The unicast routing queues can be configured using either a simple VOQ structure or an input/output pair queuing (IOPQ) structure. Each unicast or multicast routing queue has two sub-queues, one for best-effort (BE) traffic and the other for guaranteed bandwidth (GBW) traffic. The best effort traffic will be served using a work conserving fair queuing scheduling while

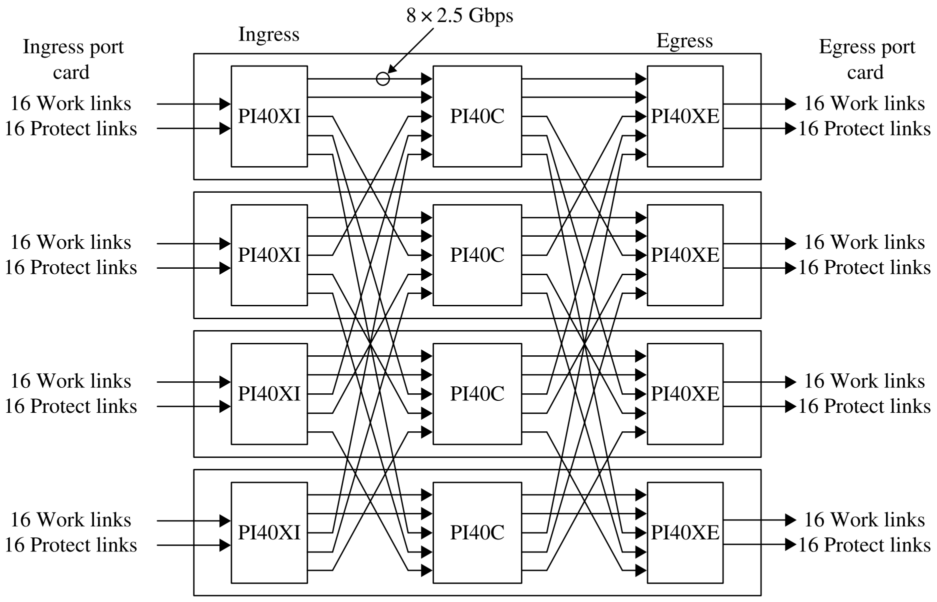


Figure 16.63 Typical three-stage Clos network of Pi40X and Pi40C.

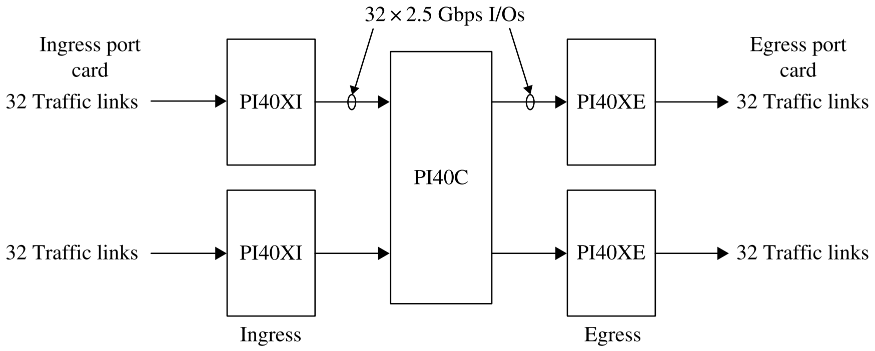


Figure 16.64 80 Gbps switching fabric composed of Pi40X and Pi40C.

the guaranteed traffic, most of which is real-time traffic, will be transferred through prior virtual TDM pipes.

In the egress QE Pi40X, the 1024 unicast queues are divided into different groups up to the number of line side egress links. For example, an egress Pi40X that is connected to eight OC48 links will provide 1024/8 = 128 queues for each OC48 link. The cell destined for one of these links will be buffered in one of the corresponding 128 queues. Each group of queues is scheduled by a separate scheduler.

The scheduling of Pi40X/Pi40C chip set still follows the request–grant manner. In any given scheduling period, the GBW traffic is always prioritized over BE traffic. The GBW traffic is scheduled using a shaped virtual clock so that each GBW queue will be served once per service period. The service period of each GBW queue is a configurable value set by the user to provide proper priority. When there is no GBW traffic to schedule, a BE queue is chosen using a WRR algorithm. In every scheduling period, one queue selection is made and this request is sent to one of the connected XB Pi40C inside a previously granted cell's header. The Pi40C has an AR that decides whether to accept the request or not. Once the request is accepted, the grant signal is attached in the header of a cell destined for the requesting ports. The egress Pi40X receives the grant and further passes it to the

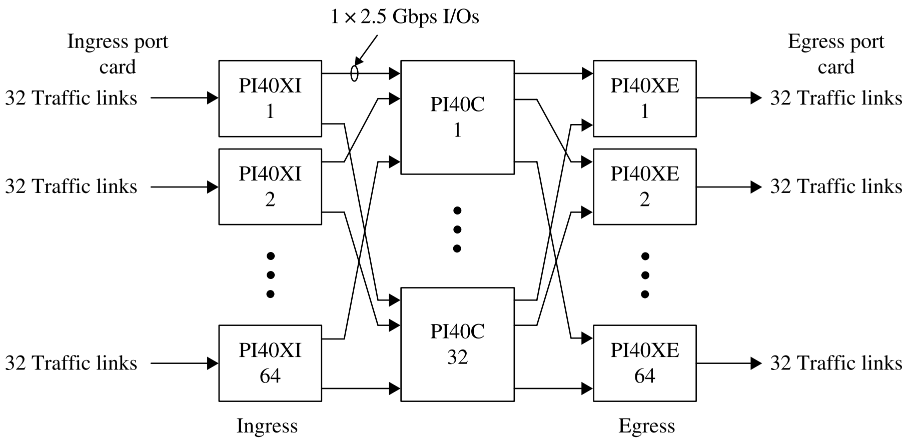


Figure 16.65 2.5 Tbps switching fabric composed of Pi40X and Pi40C.

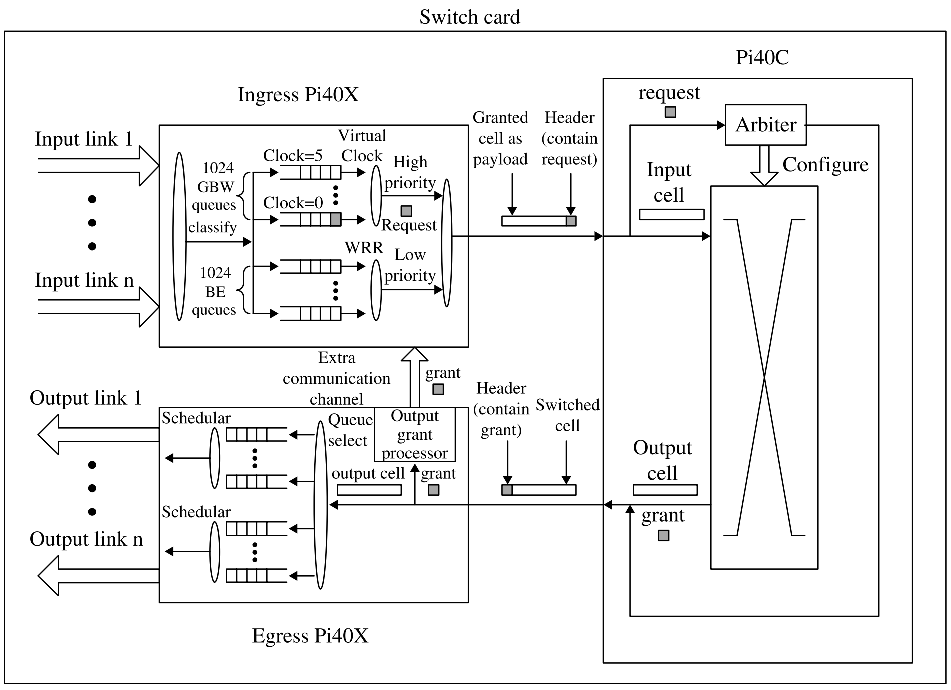


Figure 16.66 Queuing and scheduling structure of ingress and egress Pi40X.

ingress Pi40X in the same port card through the extra communication channel between them. Finally, the granted cell is transferred out to the XB in the next cycle.

Pi40SAX Stand-Alone Switch Chip. Figure 16.67 shows another switching fabric solution from Agere, the stand-alone Pi40SAX. This switching fabric chip does not use an input-queuing structure anymore. Instead, input cells from each port are buffered in a single shared memory. Thus, there is no need to use separate ingress/egress queuing chips for each port and this greatly reduces the system's complexity. The chip performs well in building switching fabrics of edge routers. However, because of the limitation in internal shared memory's access bandwidth, the chip cannot reach a much higher switching capacity.

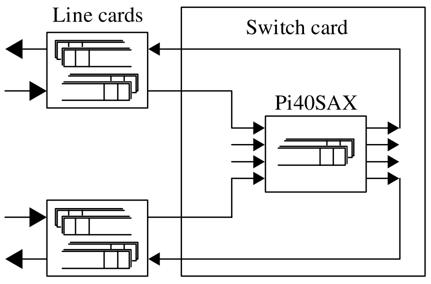


Figure 16.67 Switching system using Pi40SAX.

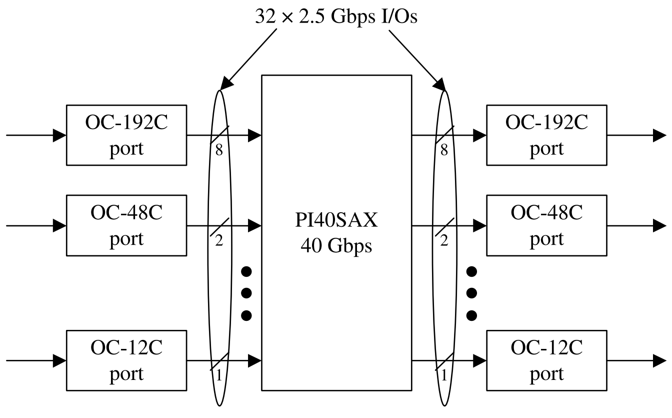


Figure 16.68 Pi40SAX link bundling.

From Figure 16.68, we can see that the Pi40SAX consists of 32 2.5 Gbps input and output serial links. Considering the in-band routing and framing overhead, the user bandwidth per serial link is less than 2.5 Gbps, approximately half of that, supporting an OC12 or GE port per link. To adapt to higher port rate, these links can be bundled flexibly. Eight serial links bundled together can accommodate an OC192 port and two links accommodate an OC48 port. The 32 serial links can be partitioned to any different groups, interfacing to a mixture of OC192, OC48, and other ports. Moreover, the Pi40SAX supports both the line card and the switch card redundancy. As shown in Figure 16.69, the failure of any single card does not affect the performance of the whole system.

Figure 16.70 gives an internal block diagram of Pi40SAX. Serialized data is accepted and converted into cell data (byte alignment) by the input link group (ILG), which bundles several serial links. The input port controller is responsible for directing traffic from the ILG, transferring cell payload with some control and routing information to the buffer memory controller (BMC) and the cell header to the queue manager (QM).

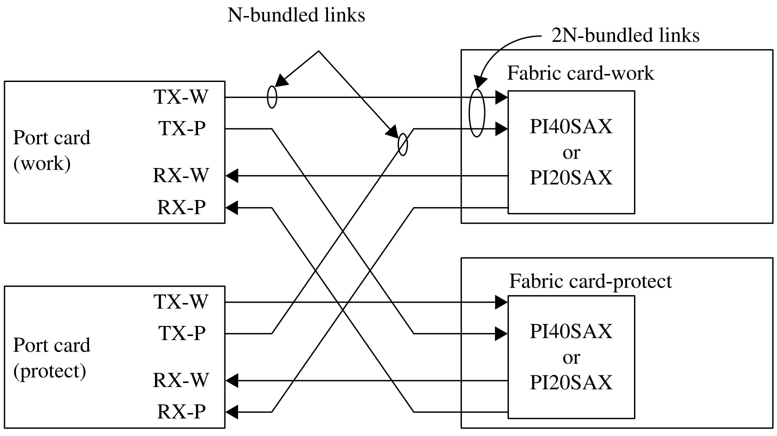


Figure 16.69 Pi40SAX 1 + 1 port and fabric card redundancy.

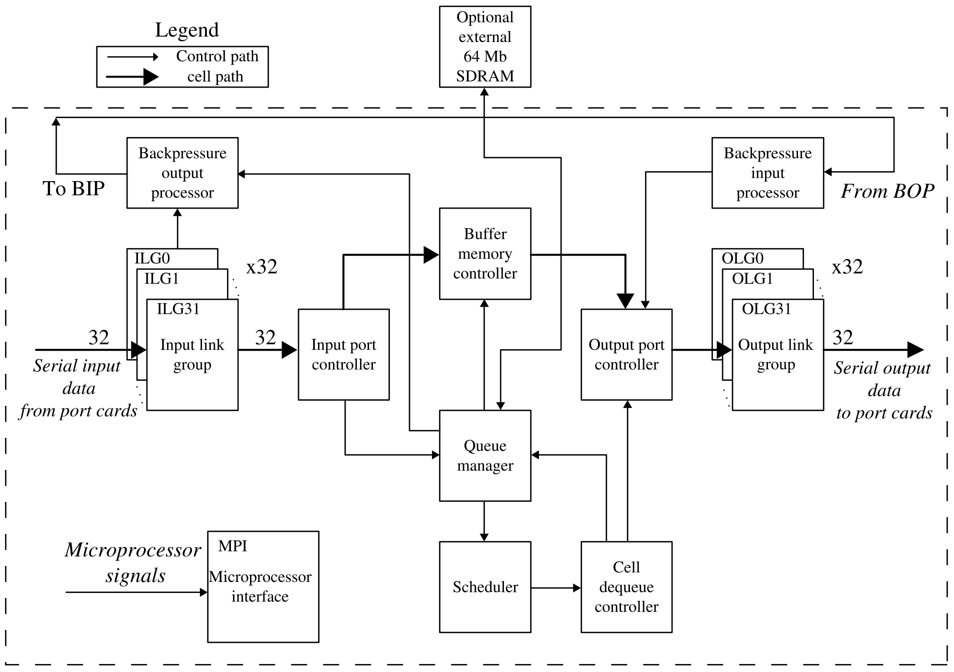


Figure 16.70 Pi40SAX internal block diagram.

The BMC buffers cell data in the internal shared memory, and the QM maintains cell address in 1024 unicast queues and 32 multicast branches. The 1024 unicast queues are configured either in a VOQ structure or an IOPQ implementation. Also, each unicast or multicast queue contains two sub-queues. One for GBW traffic, scheduled as the queue of shaped virtual clock, and the other for BE traffic, using the WRR scheduling. This queuing structure as well as its scheduling algorithm are the same as those of Pi40X/C.

The scheduling takes place in the scheduler and through the cell dequeue controller passed to the output port controller, where cell data is received and further sent to the output link group (OLG). The OLG serializes cell data, selects a correct output link, and finally forwards it to the port card through this link.

Extra backpressure mechanism is implemented in the Pi40SAX, differentiated into two types. One is the egress backpressure. Once a port card cannot afford to process incoming traffic from the switching fabric, it generates an egress backpressure signal and attaches it to an ingress cell header. When the cell is received by the ILG in Pi40SAX, this backpressure signal is extracted and through backpressure output/input processor sent to the output port controller, where cells destined for this busy output port are idled for a predefined number of cell cycles. The other type of backpressure is generated inside the Pi40SAX. When any of its internal queues exceeds the programmed threshold, backpressure signal is attached to a cell header and sent to the port card. Consequently, traffic heading for this queue is blocked until no backpressure signal is received.

Additionally, a microprocessor interface is available for configuring the chip and handling interrupts. Optional external SDRAM can be added to store lookup tables in multicast routing.

REFERENCES

[1] *The Challenge for Next Generation Network Processors*, Agere Inc., Apr. 2001, white paper.

[2] N. Shah, "Understanding network processors," Master's thesis, University of California, Berkeley, Sept. 2001, Master Degree Dissertation.

[3] "Network processing forum." [Online]. Available at: http://www.npforum.org/.

[4] J. M. Rabeay, M. Potkonjak, F. Koushanfar, S.-F. Li, and T. Tuan, "Challenges and opportunities in broadband and wireless communication designs," in *Proc. IEEE/ACM International Conference on Computer Aided Design*, San Jose, California, pp. 76–82 (Nov. 2000).

[5] *Intel Internet Exchange Architecture Network Processors Flexible, Wire-Speed Processing from the Customer Premises to the Network Core*, Intel Inc., 2002, white paper.

[6] *Network Processors: Why? What? When?*, Silicon Access Inc., July 2002, presentation.

[7] *Network Processor Designs for Next-Generation Networking Equipment*, EZchip Inc., Dec. 1999, white paper.

[8] X. Nie, L. Gazsi, F. Engel, and G. Fettweis, "A new network processor architecture for high-speed communications," in *Proc. IEEE Workshop on Signal Processing Systems*, Taipei, China, pp. 548–557 (Oct. 1999).

[9] T. Wolf and M. Frankly, "CommBench – A telecommunication benchmark for network processors," in *Proc. IEEE International Symposium on Performance Analysis of Systems and Software*, Austin, Texas, pp. 154–162 (Apr. 2000).

[10] H. Liu, "A trace driven study of packet level parallelism," in *Proc. IEEE International Conference on Communications*, New York, New York, vol. 4, pp. 2191–2195 (Apr. 2002).

[11] P. Crowley, M. E. Fiuczynski, J.-L. Baer, and B. N. Bershad, "Characterizing processor architectures for programmable network interface," in *Proc. International Conference on Supercomputing*, Santa Fe, New Mexico, pp. 54–65 (May 2000).

[12] W. Bux, W. E. Denzel, T. Engbersen, A. Herkersdort, and R. P. Luijten, "Technologies and building blocks for fast packet forwarding," *IEEE Communications Magazine*, vol. 39, issue 1, pp. 70–77 (Jan. 2001).

[13] *Challenges in Designing 40-Gigabit Network Processors*, EZchip Inc., Dec. 2001, white paper.

[14] *NP-1: Reducing Router Chip-Count, Power and Cost by 80%*, EZchip Inc., June 2002, white paper.

[15] D. A. Patterson and J. L. Hennyssy, *Computer Architecture: A Quantitative Approach*, 3rd ed. Morgan Kaufmann, San Francisco, California, 2003.

[16] *PowerNP NP4GS3 Network Processor Data Sheet*, IBM Inc., Feb. 2002.

[17] B. Klein and J. Garza, "Agere systems – communications optimized payload plus network processor architecture," in *Network Processor Design: Issues and Practices*. Morgan Kaufmann, San Francisco, California, 2002, vol. 1, pp. 219–233.

[18] *Intel IXP2800 Network Processor – For OC-192/10 Gbps network edge and core applications*, Intel Inc., product brief.

[19] J. Marshall, "Cisco systems – Toaster2," in *Network Processor Design: Issues and Practices*. Morgan Kaufmann, San Francisco, California, 2002, vol. 1, pp. 235–248.

[20] *Implementing a Flexible Hardware-based Router for the New IP Infrastructure*, Juniper Inc., white paper.

[21] *LA-1 interface*. [Online]. Available at: http://www.npforum.org/ApprovedSpecs.htm

[22] H. Bhugra, *LA-1: Examining the Look-Aside Processor Interface*. [Online]. Available at: http://www.commsdesign.com/

[23] M. J. Miller, "IDT network search engine with QDR LA-1 interface," in *Network Processor Design, Issues and Practices*. Morgan Kaufmann, San Francisco, California, 2003, vol. 2, pp. 365–384.

[24] *4.5M and 9M Network Search Engine (NSE) with QDR Interface, Data Sheet*, IDT, 2002.

[25] *Network Search Engine (NSE) TCAM with QDR Interface, User's Manual*, IDT, 2002.

[26] *PAX.port 2500*, *Data Sheet*, IDT, 2003.

[27] *PAX.port 2500*, *Technical Summary*, IDT, 2003.

[28] R. T. Vineet Dujari and A. Shelat, "PMC-Sierra, Inc. – ClassiPI," in *Network Processor Design, Issues and Practices*. Morgan Kaufmann, San Francisco, California, 2002, vol. 1, pp. 291–305.

[29] *PM2329 ClassiPI Network Classification Processor Datasheet*, PMC-Sierra, Inc., 2001.

[30] *Technical Guide to the TM10-Preliminary Data Book*, Agere System Inc., Dec. 2002.

[31] *Technical Guide to the APP550 and APP530 Network Processors*, *Version* 7, Agere System Inc., May 2003.

[32] *Technical Guide to the APP540/520 Network Processors*, *Version* 1, Agere System Inc., May 2003.

[33] *ZTM Advanced Traffic Manager Product Brief*, ZettaCom Inc., May 2004.

[34] *ZTM552 Traffic Manager Preliminary Data Sheet, version 3.0*, ZettaCom Inc., Dec. 2003.

[35] *ZTM552 Traffic Management Training*, ZettaCom Inc., May 2004.

[36] Vitesse. [Online]. Available at: http://www.vitesse.com

[37] *VSC870 datasheet Rev 4.0: high performance serial backplane transceiver*, Vitesse. [Online]. Available at: http://www.vitesse.com

[38] *VSC880 datasheet Rev 4.0: high performance* 16×16 *serial crosspoint switch*, Vitesse. [Online]. Available at: http://www.vitesse.com

[39] *GigaStream intelligent switch fabric VSC872/VSC882 design manual Rev 2.2*, Vitesse. [Online]. Available at: http://www.vitesse.com

[40] *TeraStream intelligent switch fabric VSC871/VSC881 design manual Rev 1.0*, Vitesse. [Online]. Available at: http://www.vitesse.com

[41] *Cyclone Switch Fabric S8505-S8905 Product Concept*, *Rev0.07*, AMCC.

[42] *PowerPRS Q-64G Packet Routing Switch datasheet, initial release*, IBM.

[43] *Technical Guide to the Pi40X*, *Rev 18*, Agere System Inc., Mar. 2003.

[44] *Technical Guide to the Pi40C*, *Rev 13*, Agere System Inc., Mar. 2003.

[45] *Technical Guide to the Pi40SAX and Pi20SAX*, *Rev 12*, Agere System Inc., Sept. 2003.

INDEX

High Performance Switches and Routers, by H. Jonathan Chao and Bin Liu
Copyright © 2007 John Wiley & Sons, Inc.