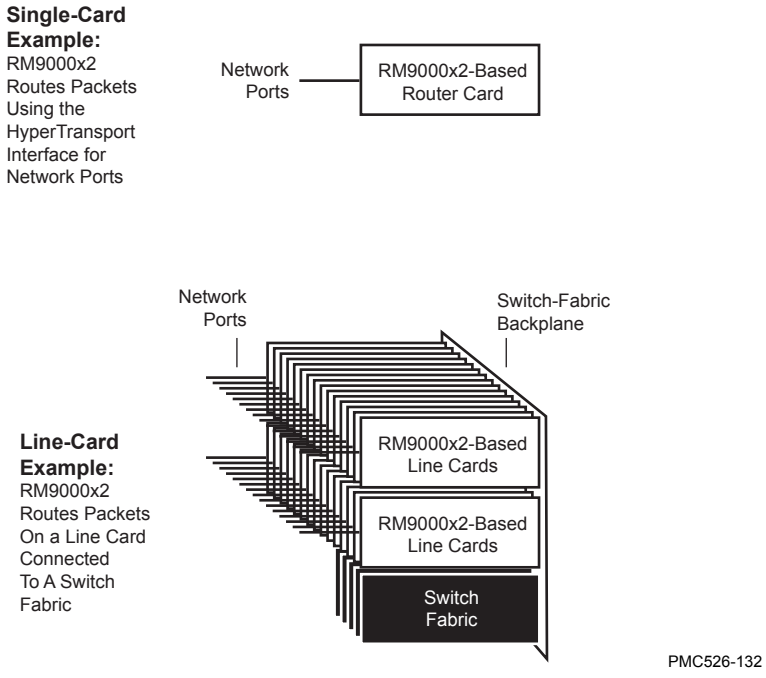
1.2 Application Examples

Although the RM9000x2 chip can be used in many types of applications, including digital imaging and advanced storage systems, its design is especially well-suited for networking applications. This section presents a few examples of how the chip might be used in such applications.

Figure 1 shows two basic types of networking applications. Figure 1 (top) shows a *Single-Card Example*, in which one RM9000x2-based card supports multiple network ports. The chip performs all control-plane and data-plane functions, including routing-table lookups, port-to-port forwarding, and statistics gathering. The chip's HyperTransport interface can support multiple network ports, which are implemented as multiple HyperTransport devices in the HyperTransport topology.

Figure 1 (bottom) shows a *Line-Card Example*, in which multiple RM9000x2-based line cards—each one supporting a set of network ports—are connected to a switch fabric via a backplane. In this example, the processing of a typical packet is split. Ingress processing is done by the RM9000x2 chip on the card through which the packet enters the switch fabric, and egress processing by the card through which the packet leaves. By contrast, in the single-card example all packet processing—ingress and egress— is done by a single RM9000x2 chip.

Figure 1 Networking Examples



1.2.1 Memory Accesses in Networking Examples

Figure 2 shows the basic sequence in which external SDRAM or FCRAM memory is accessed as packets move in and out of the RM9000x2 chip for these two examples. The *Line-Card Example* in Figure 2 shows only the switch-output movement; the switch-input movement follows the same paths, but in the opposite direction.

Figure 2 Memory Accesses for Networking Examples

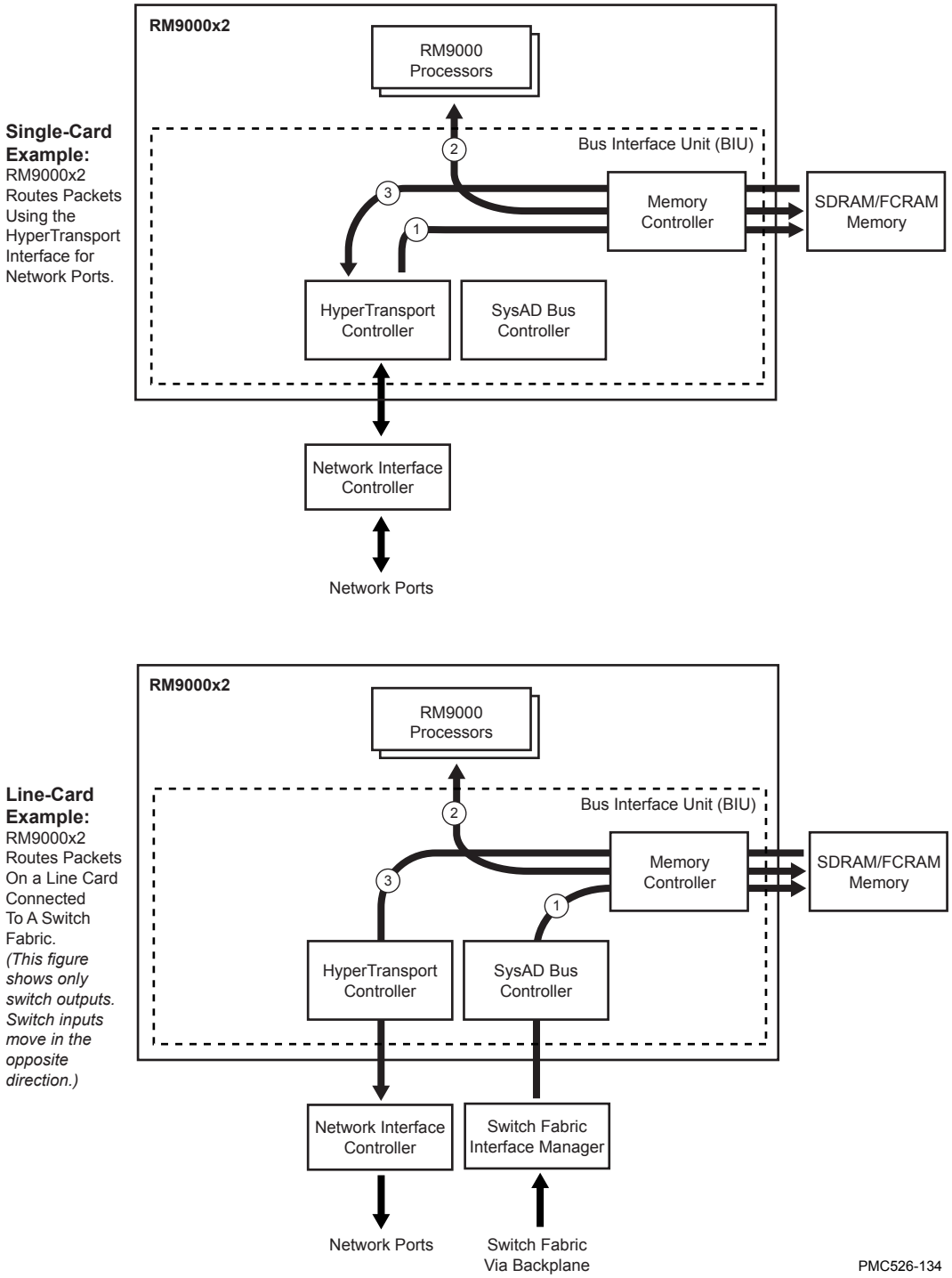


Figure 2 (top) shows the sequence for the *Single-Card Example*:

1. Packets coming in from the network through the HyperTransport port are written to memory.
2. Packets (or at least the header portions) are read from memory by a processor, operated on, and written back to memory.
3. Packets going out onto the network through the HyperTransport port are read from memory.

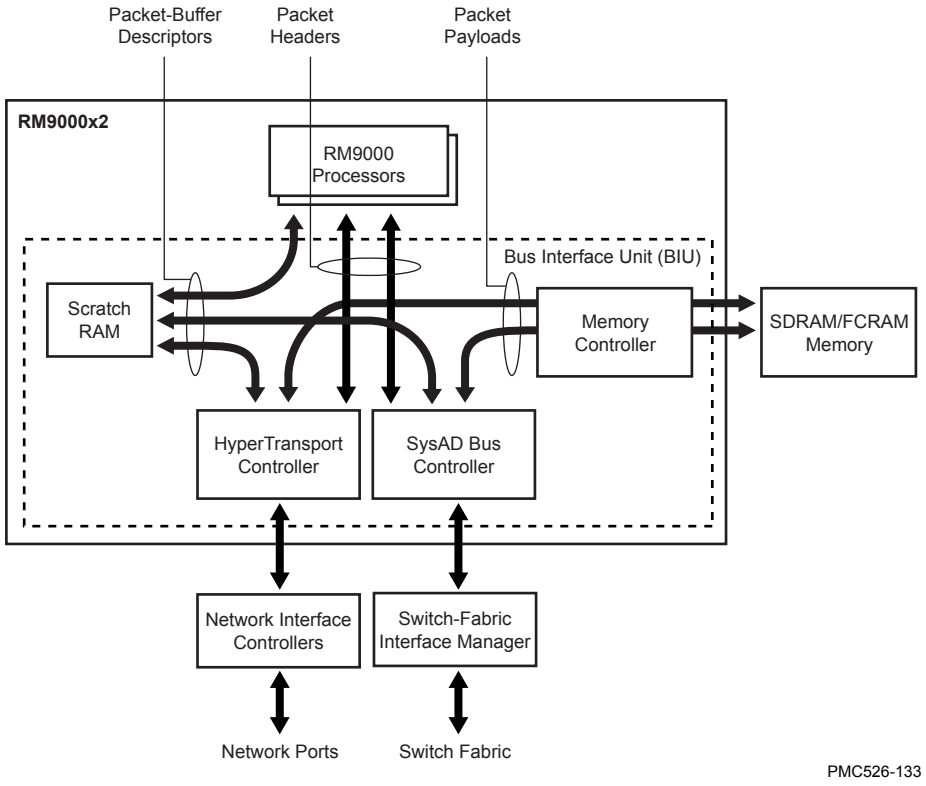
Figure 2 (bottom) shows the switch-output case of the *Line-Card Example* (the switch-input case moves data in the reverse direction). In this example, the SysAD interface connects to the switch fabric via a backplane, and the HyperTransport interface connects to the physical media on one or more network ports. The sequence is:

1. Packets coming in on the SysAD interface from a switch-fabric output port are written to memory.
2. Packets (or at least the header portions) are read from memory by a processor, operated on, and written back to memory.
3. Packets going out onto the network through the HyperTransport port are read from memory.

In the The *Line-Card Example*, the 64-bit, 200MHz SysAD interface used for the switch connection can support throughput for one or more OC-12 (622Mbit/sec) flows. However, flow latency is longer in the *Line-Card Example* than in the *Single-Card Example*, because packets must traverse the memory interface of two line cards to pass through the switch—store-and-forward on the input to the switch fabric, and store-and-forward on the output of the switch fabric. This results in four memory accesses, which is twice the number of memory accesses required in the *Single-Card Example*, not including memory accesses that may be performed by queuing functions for the switch.

Figure 3 shows details of how data might move in the *Line-Card Example* of Figure 2 (bottom).

Figure 3 Details of Line-Card Data Movement



In Figure 3, both directions of data movement (switch inputs and switch outputs) are combined into single bidirectional paths. The example shows some of the methods by which specific parts of the data can be handled:

- *Packet-Buffer Descriptors*—Descriptors that specify the properties of packet-traffic blocks can be moved between the HyperTransport network port, the SysAD switch-fabric port, and an RM9000 processor by means of the on-chip scratch RAM. This avoids the need to go off-chip to SDRAM/FCRAM memory to momentarily store these short-lived data structures. Transfers between scratch RAM, the HyperTransport interface, and the SysAD interface can be done using DMA.
- *Packet Headers*—Headers of programmable size can be transferred by DMA between the HyperTransport or SysAD interfaces and a processor's L1 Dcache or L2 cache:
  - The *Fast Packet Cache* feature can transfer header data directly into a processor's L1 Dcache. All accesses (including writebacks) touch only the processor's Dcache and bypass its L2 cache.
  - The *Direct Deposit Cache* feature can transfer header data directly into a processor's L2 cache, so as to avoid the latency of a memory read to begin header processing. The *Live*

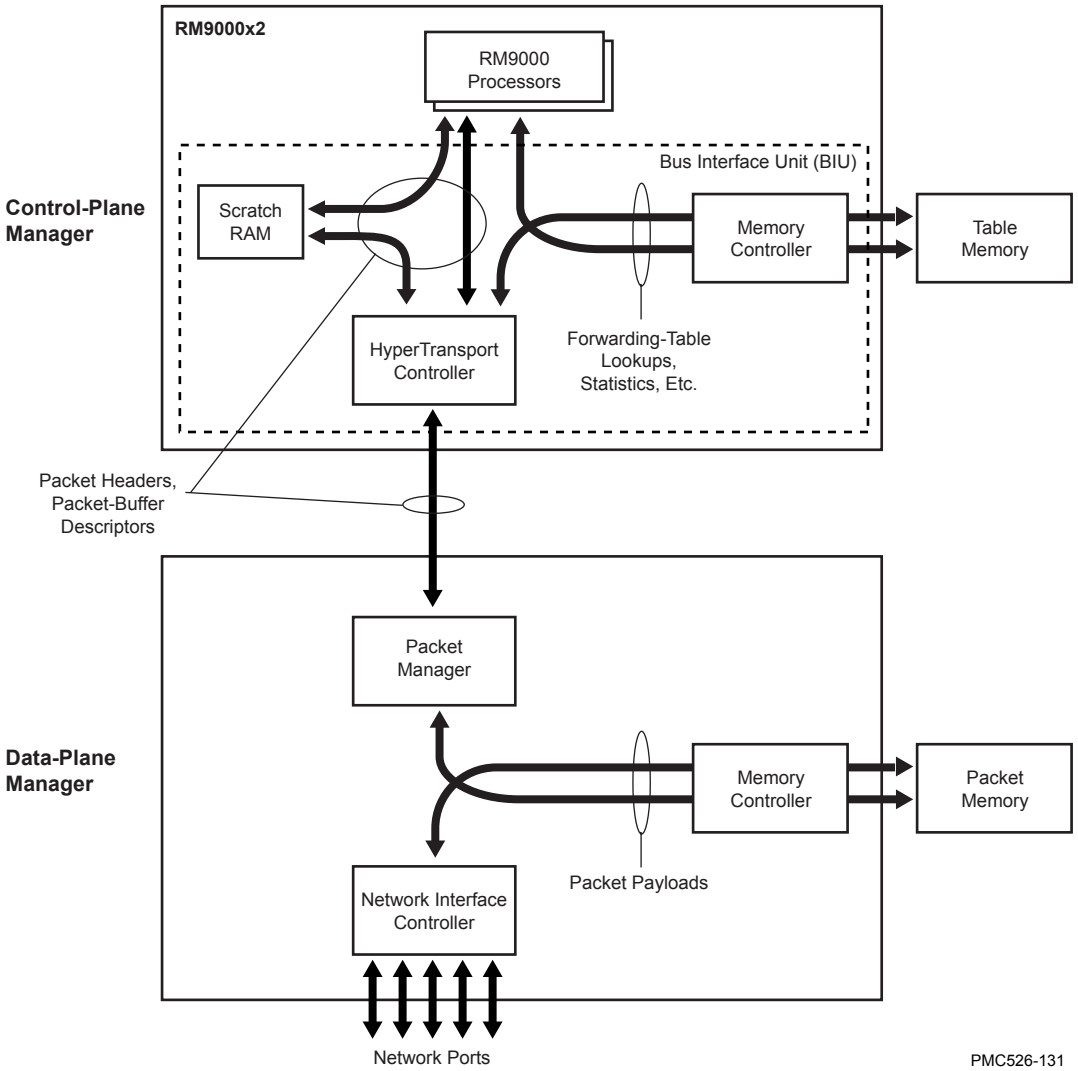
Deposit subfeature is supported on both the HyperTransport and SysAD interfaces. The *Auto Deposit* subfeature is supported only on the HyperTransport interface.

- *Packet Payloads*—The data portions of packets can be transferred by DMA between the HyperTransport or SysAD interface and the SDRAM/FCRAM memory.

1.2.2 Multi-Chip Networking Example

Figure 4 shows a router design in which the control-plane and data-plane functions are split between two chips.

Figure 4 Two-Chip Networking Example



The RM9000x2 chip, Figure 4 (top), handles control-plane functions, such as routing-table lookups, routing-table updates, and statistics gathering. It passes packet headers and packet-buffer descriptors to the data-plane chip, Figure 1 (bottom), which stores packet payloads and forwards them after receiving updated packet headers from the RM9000x2 chip. This type of design is well-suited to streaming data, in which the payloads have strong locality of memory reference (strongly sequential reference patterns). The control-plane functions, by contrast, have weak locality of memory reference and thus involve longer memory-access latencies that may require a dedicated control-plane manager.