

# Packet Subsystem on a Chip

Teja's packet-engine technology integrates all of the key aspects of a flexible packet processor.

by Bryon Moyer  
VP, Product Marketing  
Teja Technologies, Inc.  
[bmoyer@teja.com](mailto:bmoyer@teja.com)

As the world gets connected, more and more systems rely on access to the network as a standard part of product configuration. Traditional circuit-based communications systems like the telephone infrastructure are gradually moving towards packet-based technology. Even technologies like Asynchronous Transfer Mode (ATM) are starting to yield to the Internet Protocol (IP) in places. All of this has dramatically increased the need for packet-processing technology.

The content being passed over this infrastructure has increased the demands on available bandwidth. Core routers target 10 Gbps; edge and access equipment work in the 1-5 Gbps range. Even some end-user equipment is starting to break the 100 Mbps range. The question is how to design systems to accommodate these speeds.

These systems implement a wide variety of network protocols. Because the protocols start out as software, it's easiest for network designers if as much of the functionality as possible can remain in software. So the further software programmability can be pushed up the speed range, the better. Although FPGAs can handle network speeds as high as 10 Gbps, RTL has typically been required for 1 Gbps and higher.

# Most traffic that goes through the system looks alike, and processors can be optimized for that kind of traffic.

Teja Technologies specializes in packet-processing technologies implemented in high-level software on multi-core environments. Teja has adapted its technology to Xilinx® Virtex™-4 FPGAs, allowing high-level software programmability of a packet-processing engine built out of multiple MicroBlaze™ soft-processor cores. This combination of high-level packet technology and Xilinx silicon and core technology – using Virtex-4 devices with on-board MACs, PHYs, PowerPC™ hard-core processors, and ample memory – provides a complete packet-processing subsystem that can process more than 1 Gbps in network traffic.

## The Typical Packet Subsystem

The network “stack” shown in Figure 1 is typically divided between the “control plane” and the “data plane.” All of the packets are handled in the data plane; the control plane makes decisions on how the packets should be processed. The lowest layer sees every packet; higher layers will see fewer packets.

The control plane comprises a huge amount of sophisticated software. The data-plane software is simpler, but must operate at very high speed at the lowest layers because it has such a high volume of packets. Packet-processing acceleration usually focuses on layers one to three of the network stack, and sometimes layer four.

Most traffic that goes through the system looks alike, and processors can be optimized for that kind of traffic. For this reason, data-plane systems are often divided into the “fast path,” which handles average traffic, and the “slow path,” which handles exceptions. Although the slow path can be managed by a standard RISC processor like a PowerPC, the fast path usually uses a dedicated structure like a network processor or an ASIC. The focus of the fast path is typically IP, ATM, VLAN, and similar protocols in layers two and three. Layer four protocols like TCP and UDP are also often accelerated.

Of course, to process packets, there must be a way to deliver the packets to and from the fast-path processor. Coming off an Ethernet port, the packets must first traverse the physical layer logic (layer one of the stack, often a dedicated chip) and then the MAC (part of layer two, also often its own dedicated chip).

One of the most critical elements in getting performance is the memory.

Memory is required for packet storage, table storage, and for program and data storage for both the fast and slow paths. Memory latency has a dramatic impact on speed, so careful construction of the memory architecture is paramount.

Finally, there must be a way for the control plane to access the subsystem. This is important for initialization, making table changes, diagnostics, and other control functions. Such access is typically accomplished through a combination of serial connections and dedicated Ethernet connections, each requiring logic to implement.

A diagram of this subsystem is shown in Figure 2; all of the pieces of this subsystem are critical to achieving the highest performance.

## The Teja Packet Pipeline

One effective way to accelerate processing is to use a multi-core pipeline. This allows you to divide the functionality into stages and add parallel elements as needed to hit performance. If you were to try to assemble such a structure manually, you would immediately

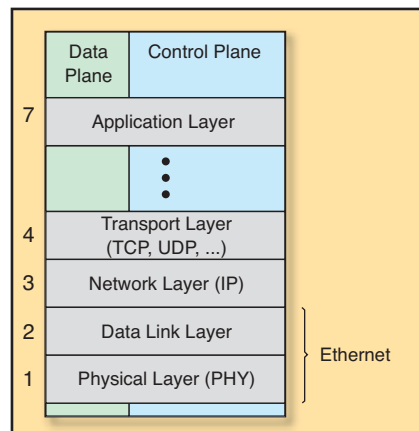


Figure 1 – The network protocol stack

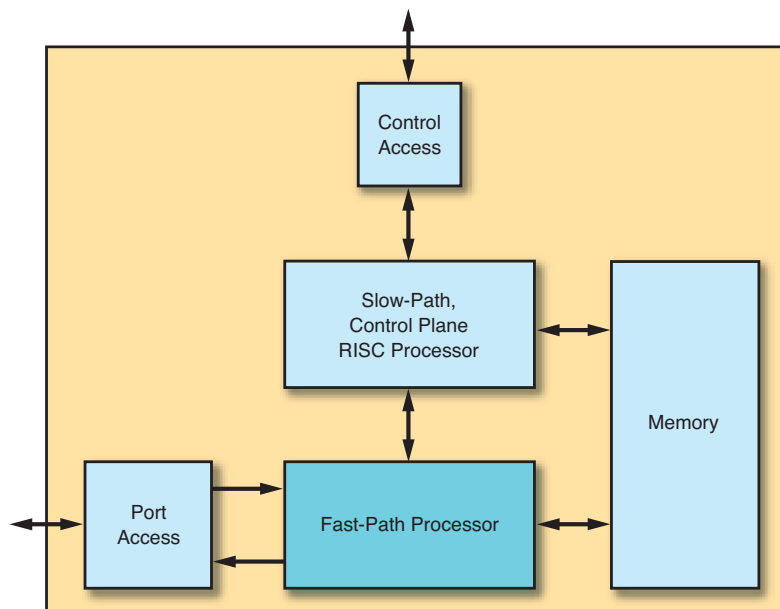


Figure 2 – Typical packet-processing subsystem

ly encounter the kinds of challenges faced by experienced multi-core designers: how to structure communication between stages, scheduling, and shared resource access.

Teja has developed a pipeline structure by creating its own blocks that implement the

Access to the pipeline is provided by a block that takes each packet and delivers the critical parts to the pipeline. Because this block is in the critical path for every packet, it must be very fast, and has been designed by Teja for very high performance.

you can add more parallel processing, or create another pipeline stage. The reverse is also true: if a given pipeline provides more performance than the target system requires, you can remove engines, making the subsystem more economical.

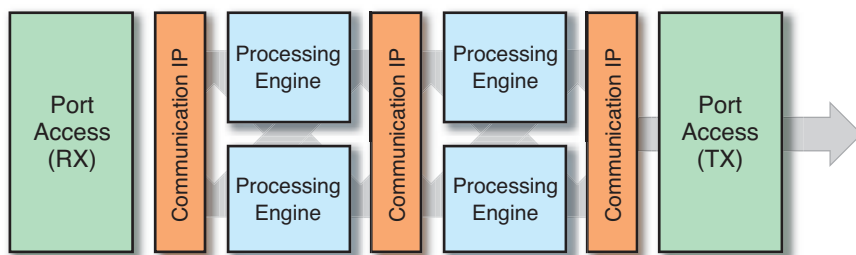


Figure 3 – Teja packet-processing pipeline

necessary functions for efficient processing and inter-communication. By taking advantage of this existing infrastructure, you can assemble pipelines easily in a scalable fashion.

The pipeline comprises processing engines connected by communication blocks and accessed through packet access blocks. Figure 3 illustrates this arrangement.

The engine consists primarily of a MicroBlaze processor and some private block RAM on the FPGA. In addition, if a stage has a particularly compute-intensive function like a checksum, or a longer-lead function like an external memory read or write, an offload can be included to accelerate that function. Because the offload can be created as asynchronous if desired, the MicroBlaze processor is free to work on something else while the offload is operating.

The communication blocks manage the transition from stage to stage. As packet information moves forward, the communication block can perform load balancing or route a packet to a particular engine. Although the direction of progress is usually “forward” (left to right, as shown in Figure 3), there are times when a packet must move backwards. An example of this is with IPv4/v6 forwarding, when an IPv6 packet is tunneled on IPv4. Once the IPv4 packet is decapsulated, an internal IPv6 packet is found, and it must go back for IPv6 decapsulation.

The result of this structure is that each MicroBlaze processor and offload can be working on a different packet at any given time. High performance is achieved because many in-flight packets are being handled at once.

The key to this structure is its scalability. Anytime additional performance is needed,

### The Rest of the Subsystem

What is so powerful about the combination of Teja’s data-plane engine and the Virtex-4 FX devices is that most of the rest of the subsystem can be moved on-chip. Much of the external memory can now be moved into internal block RAM. Some external memory will still be required, but high-speed DRAM can be directly accessed by the Virtex-4 family, so no intervening glue is required. The chips have built-in Ethernet MACs which, combined with the available PHY IP and RocketIO™ technology, allow direct access from Ethernet ports onto the chip.

The integrated PowerPC cores (as many as two) allow you to implement the slow path and even the entire control plane on the same chip over an embedded operating system such as Linux. You can also provide

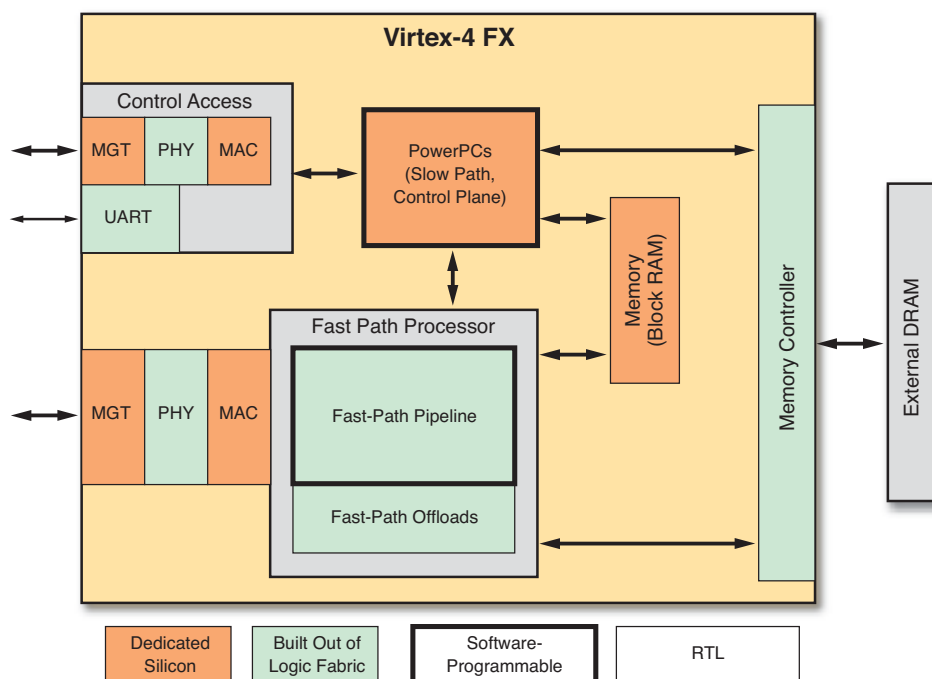


Figure 4 – Single-chip packet-processing subsystem

control access through serial and Ethernet ports using existing IP.

As a result, the entire subsystem shown in Figure 2 (with the exception of some external memory) can be implemented on a single chip, as illustrated in Figure 4.

### Flexibility: Customizing, Resizing, Upgrading

Teja's packet-processing infrastructure provides access to our company's real strength: providing data-plane applications that you can customize. We deliver applications such as packet forwarding, TCP, secure gateways, and others with source code. The reason for delivering source code is that if

One of the most important aspects of software programmability is field upgrades. With a software upgrade, you can change your code – as long as you stay within the amount of code store available. As the Teja FPGA packet engine is software-programmable, you can perform software upgrades. But because it uses an FPGA, you can also upgrade the underlying hardware in the field. For example, if a software upgrade requires more code store than is available, you can make a hardware change to make more code store available, and then the software upgrade will be successful. Only an FPGA provides this flexibility.

now IPv4 still dominates. At its most basic, IPv4 comprises the following functions:

- Filtering
- Decapsulation
- Classification
- Validation
- Broadcast check
- Lookup/Next Hop calculation
- Encapsulation

Teja has implemented these in a two-stage pipeline, as shown in Figure 5. Offloads are used for the following functions:

- Checksum calculation
- Hash lookup
- Longest-prefix match
- Memory access

This arrangement provides full gigabit line-rate processing of a continuous stream of 64-byte packets, which is the most stringent Ethernet load.

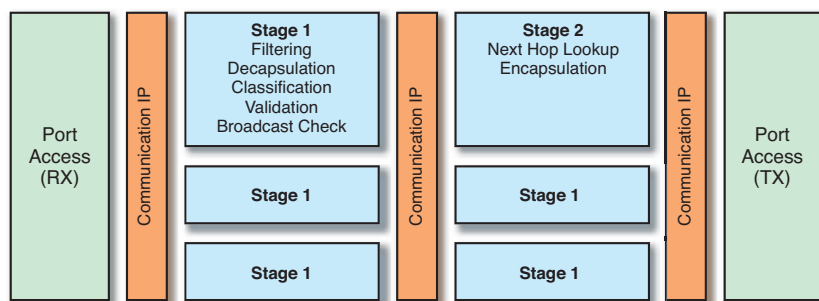


Figure 5 – IPv4 forwarding engine

you need to customize the operation of the application, you can alter the delivered application using straight ANSI C. Even though you are using an FPGA, it is still software-programmable, and you can design using standard software methods.

An application as delivered by Teja is guaranteed to operate at a given line rate. When you modify that application, however, the performance may change. Teja's scalable infrastructure allows you to tailor the processor architecture to accommodate the performance requirements in light of changed functionality.

In a non-FPGA implementation, if you cannot meet performance, then you typically have to go to a much larger device, which will most likely be under-utilized (but cost full price). The beauty of FPGA implementation is that the pipeline can be tweaked to be just the right configuration, and only the amount of hardware required is used. The rest is available for other functions.

Because a structure like this is typically designed by high-level system designers and architects, it is important that ANSI C is the primary language. At the lowest level, the hardware infrastructure, the mappings between software and hardware, and the software programs themselves are expressed in C. Teja has created an extensive set of APIs that allow both compile-time and real-time access from the software to the various hardware resources. Additional tools will simplify the task of implementing programs on the pipeline.

### IPv4 Forwarding Provides Proof

Teja provides IPv4 and IPv6 forwarding as a complete data-plane application. IPv4 is a relatively simple application that can illustrate the power of this packet engine. It is the workhorse application under most of the Internet today. IPv6 is gradually gaining some ground, with its promise of plenty of IP addresses for the future, but for

### Conclusion

Teja Technologies has adapted its packet-processing technology to the Virtex-4 FX family, creating an infrastructure of IP blocks and APIs that take advantage of Virtex-4 FX features. The high-level customizable applications that Teja offers can be implemented using software methodologies on a MicroBlaze multi-core fabric while achieving speeds higher than a gigabit per second. Software programmability adds to the flexibility and ease of design already inherent in the Virtex family.

The flexibility of the high-level source code algorithms is bolstered by the fact that the underlying hardware utilization can be specifically tuned to the performance requirements of the system. And once deployed, both software and hardware upgrades are possible, dramatically extending the potential life of the system in the field.

Teja Technologies, the Virtex-4 FX family, and the MicroBlaze core provide a single-chip customizable, resizable, and upgradable packet-processing solution. ●●●