Electronic Design

SCOTT SCHWEITZER, Director of SmartNIC Product Planning,
VISWATEJA NEMANI, Product Manager of AI & Software Solutions,
JON SREEKANTH, Software Architect, and
ROB LATIMER, Software Architect, Achronix Semiconductor

Redefining SmartNICs: The Composable Advantage

Explore the future of high-speed networking, uncovering how SmartNICs and FPGA technology transform packet processing and reshape 400 GbE.

A SmartNIC is a programmable device that connects a server's CPU complex to a network. A composable SmartNIC adds enhanced flexibility, which means that everything between the raw Ethernet interface and the PCI Express bus to the host CPU cores can be completely and dynamically reconfigured while packets are in flight.

Most SmartNICs or DPUs on the market are built around a single application-specific integrated circuit (ASIC), with the best examples being the NVIDIA Bluefield and Intel's Mt. Evans. While these SmartNICs include programmable CPU cores to assist in packet processing, their architectures are fixed. As a result, their packet-processing pipelines are very well-defined, highly optimized, and cast in silicon. If composability is desired, then these approaches are too rigidly designed.

Building an ASIC is easier, far more predictable, power-efficient, reliable, and faster to bring to market to achieve the vendor's SmartNIC revenue targets than a large programmable FPGA, which is required for a composable platform. Still, it's far less flexible than a composable platform.

With composability appliances and solutions, vendors can build whatever their clients need and tune those solutions over time along a wide range of potentially shifting performance priorities. This is extremely hard to do with a programmable ASIC and, in some cases, even impossible. By composable, we mean using dynamically reconfigurable logic coupled with an on-chip mesh network, rather than CPU instructions running on cores connected via fixed data buses.

Inside the FPGA Core: Powering Composable SmartNICs

A composable platform can spin up dozens of parallel logic blocks that can manipulate packet streams, transforming each stream with every clock cycle. These blocks can change as the role of the SmartNIC evolves with the required server workloads over minutes, hours, or even years. These flexible logic structures—field-programmable gate arrays (FPGAs)—are at the heart of composable SmartNICs. This same technology has been an integral computational component of the navigational and control systems for our space probes and Mars rovers for decades. Now, they're found in autonomous vehicles on this planet.

These FPGAs aren't simply a block of programmable logic on a chip.They're surrounded by other hardened IP blocks that condition the inputs and outputs. One example is a 112-Gb/s serializer deserializer (SerDes) feeding into multi-rate 400-GbE controllers and fifth-generation PCI Express controllers, GDDR6 and DDR4 memory controllers, as well as general-purpose input/output *(see figure)*.

These programmable logic blocks include two extraordinary types of local memories residing next to the logic on the chip: a high-speed register file called LRAM and block memories called BRAMs, as well as digital signal processors and machine-learning processors assigned to each block. Then, these logic blocks are tied to all of the perimeter-hardened IP blocks by a two-dimensional on-chip mesh network. This provides a complete fabric to load a dynamically composable multi-stage networking pipeline crafted from one or more streams of bits to configure the programmable logic.

Navigating the Soft Pipeline: Packet Processing Insights

This soft pipeline has multiple stages *(see figure)*. The first receives and conditions packets from the Ethernet Controller. It then caches them in a first-in first-out (FIFO) queue in the GDDR6 until the rest of the pipeline is ready to process them.

This stage in our implementation is called the Packet Interface. It has a second component that takes a buffer and transmits it as a packet into the Ethernet Controller. For 400 GbE, the Packet Interface has four independent Network Access Points (NAPs) in the mesh fabric on the chip, enabling it to sustain 400 Gb/s in both directions.

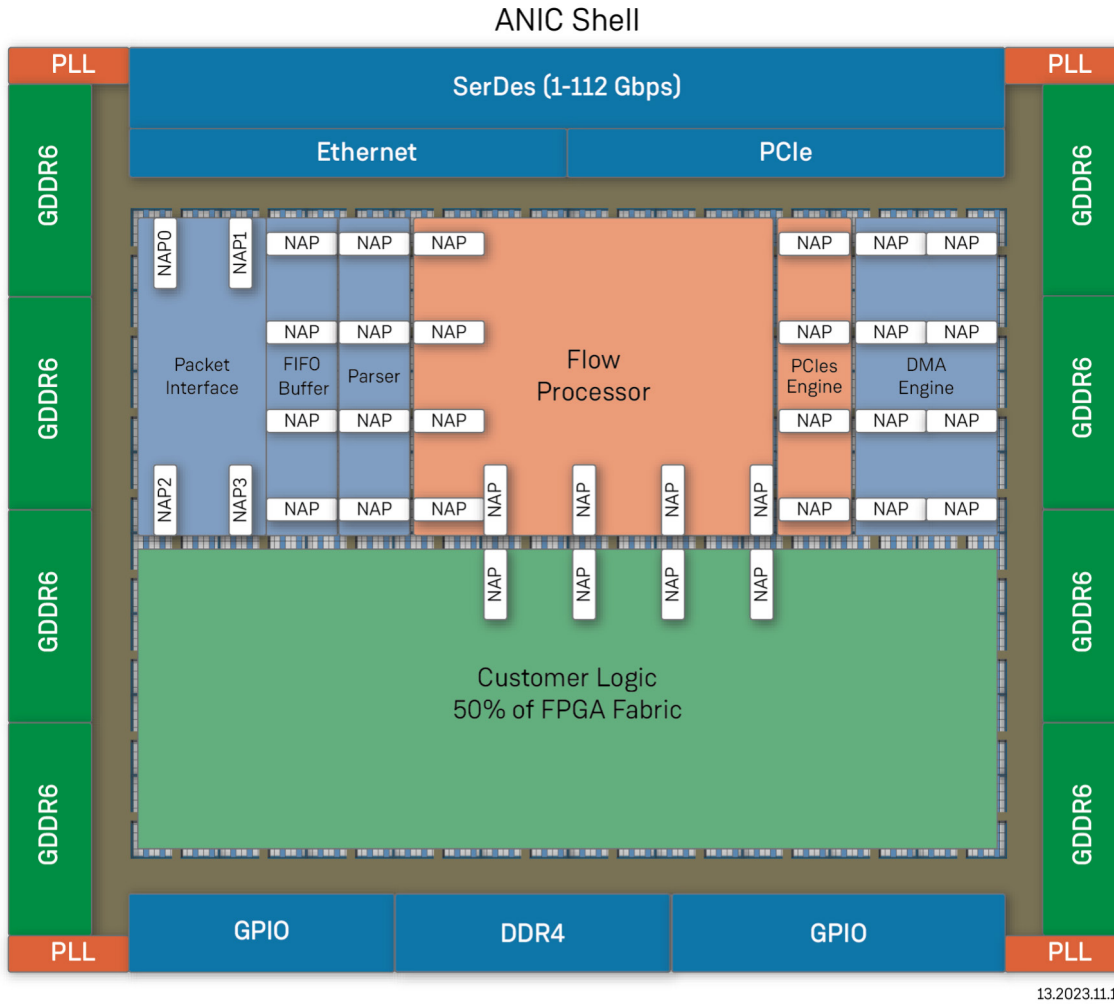
The following pipeline stage is called the Packet Parser on the receive side. It feeds off the FIFO queue, pulling packets from it, separating the header, and then doing some basic transforms on the header. These are subsequently saved as metadata alongside the header for future processing and potential indexing into a gargantuan flow table.

This stage also handles unwrapping packets and virtualization protocols. The packet could be handed off to custom code written by an integrator, OEM, or the customer.

Advanced Flow Management: GFT and Rules Engine

Written in RTL, this custom code can do anything with network packets. The only limit is the number of physical resources available, logic and memory, as well as the imagination of the engineer building the logic blocks:

- A Generic Flow Table (GFT) that takes the hash of the packet header, computed by the above Packet Parser, to determine in a "fast path" manner if this packet is part of a flow that's already been seen. If so, it records the statistics and executes the action defined for this packet. The software development kit (SDK) with the composable 400G SmartNIC will include a tool to preload the GFT with table entries on startup and



13.2023.11.10

FPGAs aren't simply a block of programmable logic on a chip. They're surrounded by other hardened IP blocks that condition the inputs and outputs. (Achronix)

another tool to pull the statistics from the GFT while in use. This stage is currently under development and will be available soon.

- A Rules Engine for packets that don't match an entry in the above flow table would then be passed off to a rules engine to determine what to do with the first packet of a new flow. Once a rule matches, a GFT entry will be created for the flow represented by this first packet, and the appropriate action will be taken. In this architecture, one rule can be tested against a packet with each clock cycle, and the rules engine can handle multiple packets at any given time.
- Defense against a Distributed Denial of Service (DDoS) attack, where the header and perhaps a small portion of the payload are inspected to see if it's one of the over two dozen known DDoS attack vectors.
- String search through a packet payload, unlike a deep packet inspection, is a simple search using a predefined finite set of keywords to determine whether they exist within the packet. This can be used by governments looking for terrorist threats or companies to determine if vital code-word projects or assets are being exfiltrated or discussed with outside entities.
- Deep packet inspection via rules, where you might want them to be applied *after* the packet makes it through the GFT above for packets headed to a specific destination, like the host CPU complex or rerouted packets headed for another container, VM, or server.

There are hundreds of possible use cases, from access control to security and storage. It all depends on what the customer requires.

The final RTL logic block in a composable SmartNIC is the DMA Engine. In our case, we support both Ring and Scatter/Gather mode. Ring mode provides excellent PCI Express (PCIe) efficiency, but it needs host memory copy. Scatter/Gather uses the PCIe bus inefficiently because of small, fragmented reads/writes but avoids host memory copy.

The key here is that both modes are available to customers, who can pick which best meets the application needs: small packet performance usually favors Ring mode. In contrast, larger average packet sizes favor Scatter/Gather.

To support a 400-GbE composable SmartNIC, you will also need the corresponding SDK for the server. This kit includes the PCIe device driver for the FPGA card that links the DMA Engine to user space host memory buffers. It also includes tools for manipulating the QSFP modules, loading the GFT and Rules Engine, pulling statistics from the GFT, and other tools and sample programs for utilizing the composable pipeline.