NETRONOME

FAQ

FREQUENTLY ASKED QUESTIONS

Netronome NFP Silicon and Composable IP Blocks

AN IDEAL SILICON ARCHITECTURE FOR HETEROGENEOUS PROCESSING MUST BE ABLE TO COMBINE DOMAIN-SPECIFIC AND GENERAL-PURPOSE PROCESSING, HARDWARE ACCELERATORS, MEMORY AND I/O ELEMENTS IN A COMPOSABLE WAY WITHOUT COMPROMISING PERFORMANCE, COST AND POWER.

Composability is an established value in the world of software. Why is this relevant in the world of silicon?

Heterogeneous processing is the new normal in modern server designs. The industry is rapidly moving away from multicore-based server designs based on general purpose CPUs. Coprocessors based on Domain-Specific Architectures (DSA) complement general purpose CPUs used as host processors. The number and variety of such domains are evolving and expanding. For example, coprocessors for networking, graphics, machine learning and virtual reality are being deployed. Such DSA-based designs require domain expertise, which means that the coprocessors are designed by different teams, likely geographically dispersed or maybe even in different companies. At the same time, all such designs have some common elements; the coprocessors in these designs need:

- Efficient access to both internal (on-chip) and external memory
- Ability to use hardware-based accelerators for common functions such as hashing, cryptography, add/multiply/multiply accumulate, etc.
- High bandwidth and low latency data transfers to and from network ports and host interfaces
- Ability to execute compute tasks closest to where the data resides to limit unnecessary data transfers

An ideal silicon architecture for heterogeneous processing must be able to combine domain-specific and general-purpose processing, hardware accelerators, memory and I/O elements in a composable way such that they can be combined in the fastest possible time without compromising performance, cost and power.

For further details on this topic, please read, “The Benefits of a Composable Silicon Architecture.”

What is DSA and what are the benefits?

The idea behind DSA is to achieve higher efficiency by tailoring the architecture to characteristics of the domain. Building silicon that conforms to DSA requires more domain-specific knowledge than general-purpose processors demand.

Due to the demise of Moore's Law, Dennard scaling and the restrictions of Amdahl's Law, general-purpose processors cannot scale efficiently. DSA enables efficient scaling as a result of the following distinct advantages:

- More effective use of parallelism for a specific domain:
  a. SIMD vs. MIMD
  b. VLIW vs. Speculative, out-of-order
- More effective use of memory bandwidth:
  a. User controlled versus caches
- Eliminates unneeded accuracy:
  a. IEEE replaced by lower precision FP
  b. 32-bit, 64-bit integers to 8-16 bits
- Domain-specific programming model matches application to the processor architecture

DOMAIN-SPECIFIC ARCHITECTURES AS SEEN TODAY WITH COPROCESSORS FOR NETWORKING, SECURITY, GRAPHICS, MACHINE LEARNING AND VIRTUAL REALITY ARE THE ONLY VIABLE WAYS FORWARD.

Is it easier to deliver parallelism and performance needed in data planes using many x86 or Arm cores with some hardware accelerators?

At first glance multi-core solutions – whether used as the host processor or a coprocessor – are attractive because the programming model is well understood by many. Developers are able to program with familiar tools, and identify parallelism via threads on multiple cores. Adding cores implies more transistors, and many make the assumption that performance scales with active cores. But Amdahl's Law limits performance gains from parallel processing using multi-core architectures built using general-purpose processors.

In the highest volume applications ranging from the data center infrastructure to mobile and IoT devices, energy consumption considerations have become paramount. Dennard scaling (constant power per square millimeter of silicon) has ended, which means that general-purpose processors have reached their power limit. Thermal dissipation is maxed out - chips turn off to avoid overheating!

At 22nm process, the multi-core Intel E7-8890 with 24 cores at 2.2 GHz has TDP of 165W (power limited). At 11nm process yields, a multi-core CPU with 96 cores at 4.9 GHz is expected to consume about 295W. If the power limit is 165W, only about 54 of the 96 cores will be active. In other words, end of Dennard scaling means multi-core scaling ends; full scaling will mean "dark silicon" with cores off.

There is no obvious path to efficient scaling with general-purpose processors. Domain-specific architectures as seen today with coprocessors for networking, security, graphics, machine learning and virtual reality are the only viable ways forward.

What are the different and most popular DSAs? Which one does Netronome support?

Popular DSAs as seen today with coprocessors are for the following applications: networking, security, storage, graphics, machine learning and virtual reality.

The Netronome NFP is a DSA optimized for networking and security applications. The composable nature of the NFP silicon enables the addition of other DSA architectures. Logic Blocks from Netronome can also be combined with another DSA (such as for storage or

machine learning) on a single chip, where the new DSA Logic Blocks utilize the underlying distributed switching and processing memory facilities. In this case, functionality realized in the DSA Logic Blocks can be used by developers in the same "sea-of-workers" run to completion model.

Is the NFP based on the Intel IXP architecture?

The NFP is derived from the original Intel IXP architecture. Over the last decade, Netronome has evolved the architecture to include the distributed switch fabric (DSF), processing memories to enable two significant new capabilities not available with the original IXP architecture:

1. The architecture has been made composable by introducing Logic Blocks and the ability to effectively attach the independently-developed Logic Blocks into a silicon die. The distributed switch fabric and processing memory features introduced by Netronome form the cornerstone of the composable architecture. These innovations not only enable rapid development and the introduction of new DSA capabilities but also optimize silicon cost and power by ensuring extremely efficient data movement across all processing elements and memory. The architecture also enables a deterministic full-chip backend process with significantly more predictable labor cost and duration for the design effort.
2. Using a "sea of workers" programming model with each worker queue in the architecture executing in a run-to-completion model, developers can create code in C directly, or C through the Linux kernel in eBPF, P4 or in assembly. To make programming easy, all other work queues related to packet and memory management are handled by infrastructure firmware provided by Netronome.

As a result of the above enhancements, the only elements of the original IXP architecture that still remain in use are the processing cores, also called the micro engines or MEs. These MEs can be replaced with other ISAs (instruction set architecture), such as RISC, while keeping all the benefits highlighted above intact.

What are the different options for using Netronome silicon IP?

Netronome silicon IP can be used in the following ways:

1. As a NFP silicon product as used in SmartNICs or Smart Edge devices. Multiple product options, including software, are available.
2. As composable IP blocks that can be licensed to build your own custom SOC. Proven IP blocks are available in different process nodes.
3. Your custom logic can be made available by you as a Logic Block or Design Block and integrated in future silicon produced by Netronome. Details on Logic and Design Blocks are available in the "NFP Theory of Operation" document.

Can I develop my own data plane features using the NFP?

Netronome enables a flexible and highly performant programming environment for customers to develop their own preferred data plane with NFP and SmartNIC solutions. Developers can develop code in C directly, or C through the Linux kernel in eBPF, P4 or in assembly. C through the Linux kernel in eBPF using the LLVM compiler, and P4 using the P4.org

front end compiler are open source and vendor-agnostic ways of programming supported with the NFP.

What is the programming model of the NFP-based system? Does a user have to explicitly manage the parallelism and data across the NFPs?

The processing engines in an NFP are called Flow Processing Cores. The programmer does have to manage parallelism. The NFP includes hardware, firmware and tools to make implementing one parallelism model - a sequence of work queues - easier. Each work queue is serviced by a "sea of workers" (code running on Flow Processing Cores or other chip processing resources) with each worker executing in a run-to-completion model. Netronome has implemented multiple production-quality offload packages in the "sea of workers" model.

For a "sea of workers" model, developers can create code in C directly (an optimized version of C that can compile in a no-OS environment) or C through the Linux kernel in eBPF, P4 or in assembly. With any of these models, Netronome provides packet management infrastructure firmware so that a developer only needs to specify code for the work queues modifying packet content. All other work queues related to packet and memory management are handled by infrastructure firmware.

How does one deal with long latency operations, such as a read-modify-write of a DRAM location?

The NFP provides hardware support for read-modify-write operations at internal and external memories. A programmer implements a read-modify-write as a single line swap command. Developers typically use this operation to manage shared resources.

In addition, the architecture minimizes the impact of long latency with two other technologies:

1. All the processing cores and the memory controllers are highly multi-threaded. This allows at least one thread to be active in a processing core at any instant.
2. Memory controllers provide support for logic operations (for example vector-adds, complex look-ups) directly at the memory. This completely eliminates data movement latency.

How much bandwidth does each IP block in the composable architecture have?

Each DSF interface currently supports 128Gb/s of throughput. A Logic Block with six of them, as shown in the "NFP Theory of Operation" document, supports 768Gb/s.

Traditionally, significant internal resources, such as SRAM, an embedded FPGA or hardware accelerators have been implemented as single large multi-port blocks. With Netronome's design approach, these resources are actually distributed across multiple Logic Blocks. The aggregate bandwidth available to that resource is much higher than in a normal single-block implementation. For example, in NFPs, internal SRAM is physically distributed

across up to 10 Logic Blocks. The aggregate bandwidth available to internal memory across these Logic Blocks is now more than 1Tb/s with a clock frequency of 1GHz. Other DSA implementations, with embedded FPGA blocks, for example, can similarly be distributed across multiple Logic Blocks to increase the aggregate bandwidth available to functions implemented in the FPGA.

Can you provide a list of applications and examples of code that have been developed and/or deployed using the NFP?

The following are examples of proven NFP code and applications:

- NGFW-related code in production, for enterprise security appliance
- SSL visibility-related code in production, for enterprise visibility appliance
- Data center interconnect (VXLAN-MPLS) code managed using SDN controller, for service-provider environment
- Disaggregated storage, congestion management and tail latency reduction related code, for hyperscale data center environment
- NGFW and UTM-related code, for enterprise security appliance
- Virtual switching and stateful security, for hyperscale data center IaaS environment
- SSL visibility-related code, for enterprise visibility appliance

In addition, the following vendors have certified NFP-based firmware and software:

- Red Hat OpenStack platform for Open vSwitch offload with Netronome SmartNICs
- Juniper Contrail Cloud platform for vRouter offload with Netronome SmartNICs
- The Linux kernel community has accepted Netronome PF, VF, and DPDK PM device driver software, OVS-TC firmware, eBPF JIT compiler and eBPF-related firmware

Finally, researchers from around the world have also developed firmware for Netronome SmartNICs for a variety of applications and details can be found at Open-NFP.org.

IN OUR DESIGN APPROACH, FULL-CHIP BACKEND IS SIMPLIFIED BECAUSE WITH OUR COMPOSABLE ARCHITECTURE, ALL CONNECTIONS FOR DATA FABRIC, CLOCK AND TEST ARE HIERARCHICAL WITH DIRECT CONNECTIONS SIMPLY BETWEEN PHYSICAL NEIGHBORS.

Netronome composable IP blocks include Logic Blocks. If I knew exactly which of your Logic Blocks to compose together, how long would it take to get to GDSII files?

In our design approach, the full-chip backend is simplified because with our composable architecture, all connections for data fabric, clock and test are hierarchical with direct connections simply between physical neighbors. Full-chip GDSII files are developed by creating GDSII files for individual Logic Blocks and composing a full chip by physically abutting Logic Blocks. This approach has led to a more deterministic full-chip backend process and resulted in a predictable labor cost and duration for the design effort. The methodology has been verified across numerous devices in multiple process nodes.

Consider two options in the answer:

1. A new IC that entirely reuses existing Logic Blocks. In this approach, the time to composing a full IC is approximately three months. The steps for the three-month process are shown in the table below.
2. A new IC that is a mixture of existing and newly-developed Logic Blocks. The time to develop the RTL for the new Logic Blocks is a function of their complexity. Post RTL, the GDSIIs for the new Logic Blocks need to be generated. The new blocks are then composed with reused Logic Blocks to form the whole chip.

Step	Approach	
	Fully reuse existing Logic Blocks	Mixture of reused and new Logic Blocks
New Logic Block RTL	N/A	Function of Block complexity
New Logic Block GDSII		4-6 weeks/new Logic Block
Floor plan	4 weeks	
Full chip assembly (Switch Fabric, DFT tree)	4 weeks	
Layout and DRC/LVS/NAC	4 weeks	
Tapeout	1 week	
Total time	13 weeks	17-19 weeks backend (+frontend effort)

What IP blocks are available from Netronome and how can they be used to develop a SoC design?

Please refer to “Applying the Netronome Composable IP Blocks to Modern SoC Designs.”

NETRONOME

Netronome Systems, Inc.
2903 Bunker Hill Lane, Suite 150 Santa Clara, CA 95054
Tel: 408.496.0022 | Fax: 408.586.0002
www.netronome.com