NETRONOME

WHITE PAPER

Benefits of a Composable Silicon Architecture

NETRONOME HAS DEVELOPED A COMPOSABLE ARCHITECTURE IN WHICH CROSS-CHIP FUNCTIONS ARE IMPLEMENTED ACROSS A DISTRIBUTED SET OF COOPERATING LOGIC BLOCKS. THIS COMPOSABLE ARCHITECTURE ENABLES SCALABLE DESIGNS THAT CAN BE IMPLEMENTED RAPIDLY.

CONTENTS

1. INTRODUCTION

The increasing rate of change in protocols and algorithms related to networking, security and machine learning, coupled with the slowing down of Moore's Law have prompted growing interest in coprocessor-based SoCs to offload and accelerate domain-specific workloads. Data movement efficiency across the processing cores and memory in such SoC designs is becoming paramount. Coprocessor architectures need to be scalable to handle performance requirements both at the edge and core of the network. Lower development costs are needed for implementations to adapt to multiple domains and workload evolution. Netronome has developed a composable architecture in which cross-chip functions are implemented across a distributed set of cooperating logic blocks. This composable architecture enables scalable designs that can be implemented rapidly.

2. COMPOSABLE ARCHITECTURE

A Netronome SoC is built around a set of logic and memory elements that are instantiated multiple times in the design. Figure 1 builds a functional tree of the IP blocks as they are applied to the SoC architecture. The IP is classified as Design Blocks, Logic Blocks and Cross-Chip Functions. Multiple instances of one or more Design Blocks are combined to form a Logic Block. Multiple Logic Blocks are combined to form a SoC die.

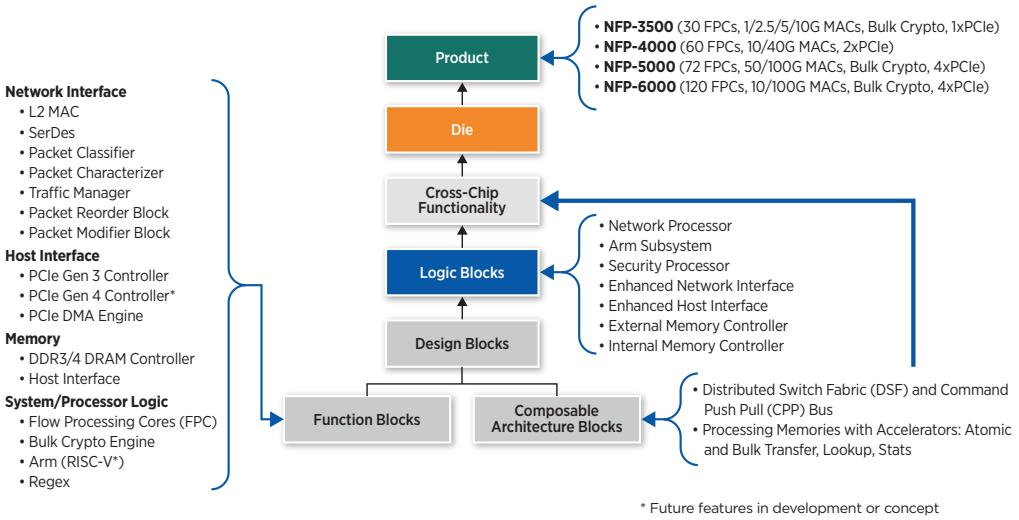

Figure 1. Functional tree of the IP blocks

2.1 Composable Blocks for Cross-Chip Functions

In SoCs built with Netronome IP, special cross-chip functions are composed from unique Composable Architecture Design Block instances across the SoC. To build a cross-chip function, every Logic Block in a SoC will need one or more instances of a Composable Architecture Block contributing to that function. At the chip level, multiple instances of these Design Blocks work together to provide these functions. By building functions from distributed instances, designs can be scaled efficiently. The NFP contains two important cross-chip functions: (a) A Distributed Switch Fabric (DSF) and (b) Processing Memory.

2.2 Distributed Switch Fabric

The main global bus for full-chip communication, the DSF, is built from such Composable Architecture Blocks.

Design

As shown in Figure 2, an NFP is physically implemented as a tiled array of Logic Blocks (or Islands). Each Logic Block contains a Bus Agent Design Block. The Bus Agent provides interfaces to other Logic Blocks (see Figure 2) and also for Design Blocks within the Logic Block (not shown in Figure 2). All physical connections between Logic Blocks in a Netronome-IP based SoC are limited to near neighbors. A Logic Block is not directly connected to any Logic Block that is not its physical neighbor.

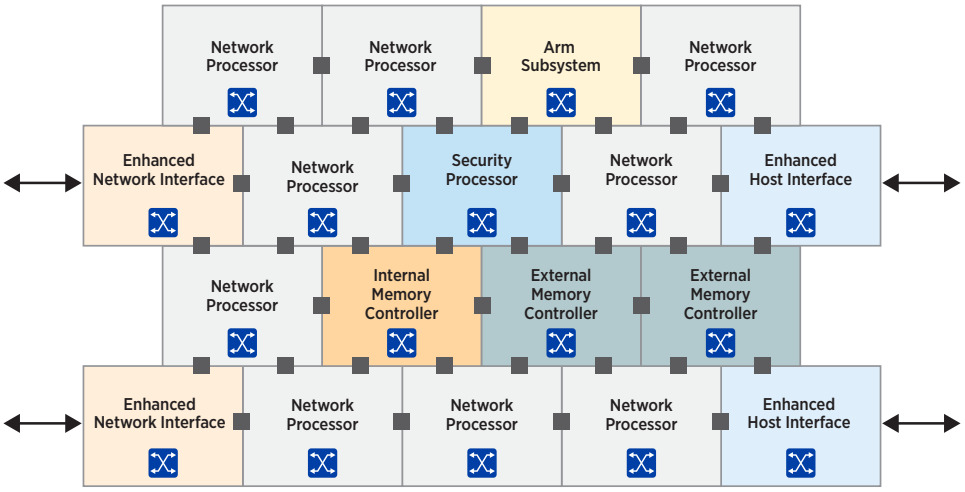

Figure 2. The Distributed Switch Fabric and tiled design

Data Routing

The DSF is logically global but has no central logic. Across the chip, the bus agents in each Logic Block coordinate to enable cross-chip data transfers between any pair of Logic Blocks. The distributed nature of the DSF enables the creation of an identical overlay mesh for all Logic Blocks as shown in Figure 3. Configurable routing across the Logic Blocks permits application-driven traffic balancing, which enables efficient routing of data across the DSF. Paths are configured a-priori, optimized for the specific applications the NFP device supports. Figure 3 shows two example routes for data transfer. Latency of access to data across the die (in a typical Netronome product) composed of the Logic Blocks varies between 5 and 20 nanoseconds. Critical paths that affect latency adversely can be shortened by changing routing.

Actual data transfer on the DSF is orchestrated programmatically from processing cores in the NFP. The DSF command syntax is extensible and supports commands for data transfer and even processing at remote bus agents in a different island. For example, processing cores can: (a) create data transfers such that accelerator blocks can have data transferred to/from memory and/or external I/O directly and (b) issue commands to process data at a remote location where it resides, minimizing data movement and processing time.

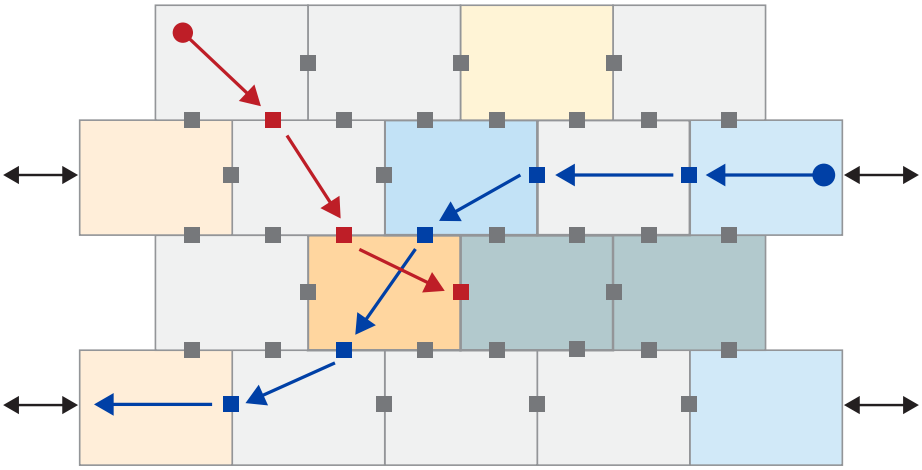

Figure 3. Routing data efficiently over the DSF

2.3 Distributed Control and Test

This class of Design Blocks realizes control functions to configure and operate the switch fabric and other Design Blocks in each Logic Block. This is implemented in a distributed manner similar to the switch fabric. A second set of Design Blocks implements a distributed JTAG and self-test functions. These two types of Design Blocks enable a Logic Block to be configured and tested when integrated with adjacent Logic Blocks in an NFP.

2.4 Processing Memory

Processing Memory is the second example of a function formed from Composable Architecture Blocks. Processing Memory enables programmable cores to perform arithmetic and logic operations on data while it still resides in the memory, without requiring a physical transfer over the data bus. It is implemented by adding Processing Design Blocks at the interfaces to internal and external memory.

The NFP has unique Processing Memories that combine data storage, access and compute. Each Processing Memory comes with multiple engines to support an extensive array of functionality for data storage and manipulation right in the memory (see Figure 4). At every memory unit, in addition to a traditional controller, compute engines support a variety of operations, each processing data based on the commands received over the DSF. When combined with the ability to execute remote commands on the DSF, Processing Memory enables significant compute at the memory, including atomic operations, bulk transfer, distributed statistics collection and a wide range of table and trie data structures. Many network functions can be executed on data where it resides in memory, reducing data movement costs and processing time.

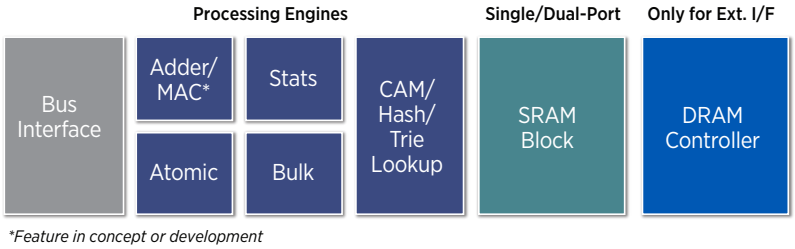

Figure 4. Processing memory architecture

There are two types of processing memories present in any NFP: A high-bandwidth multiport internal memory and an interface to external memory. The internal memory (SRAM) is physically distributed across multiple Logic Blocks. The internal memory components local to each Logic Block are referred to as Cluster Local Scratch (CLS) and Cluster Target Memory (CTM). Local memories in Logic Blocks are tightly coupled with the local processing logic (the FPCs) for quick command execution and data transfers. The External Memory Controller Unit adds the infrastructure for accessing external memory (DRAM) through DDR interfaces and also has a 3MB data cache.

2.5 Advantages

The distributed implementation of the DSF and Processing Memory offers several advantages:

Performance: The parallelism enables high aggregate bandwidth comparable to a crossbar because many transactions can be executed in parallel. For example, internal memory in an NFP is realized as multiple physical memories, one in each Logic Block. The memories in each

Logic Block can all be accessed simultaneously. The DSF has separate data/command paths and masters for each island. With distributed arbitration, many near-neighbor connections can operate simultaneously.

Scalability: The DSF infrastructure is scalable across many different Logic Block instances, supporting rates peaking at billions of commands and terabits of data transfers per second, giving this distributed bus throughput comparable to a crossbar. The design can be scaled by scaling the number and throughput of near-neighbor connections. For example, in the specific example shown in Figure 2, each Logic Block has six DSF interfaces (K=6). Each 64-bit wide DSF interface link at 1GHz delivers 128Gb/s of bidirectional bandwidth into each Logic Block at the nodal point. So, a total of K*128Gb/s throughput is theoretically possible across each Logic Block. In the Figure 2 example, each Logic Block can support 768Gb/s of bidirectional bandwidth.

Design Efficiency: In a DSF implementation, all direct logical connections between Logic Blocks are physically restricted to near neighbors. With this approach, these connections can be realized by simply abutting neighboring Logic Blocks. Realizing connections through simple abutment greatly simplifies routing and timing analysis for top-level design. A physically distributed bus consumes less silicon die area than a globally routed bus. The bus can be clocked at a higher frequency because all wires are local.

Power Efficiency: With almost no global physical connections, the DSF dissipates less power and can be clocked faster than a global bus.

Data Movement Efficiency: As mentioned previously, internal memory in an NFP is realized as multiple memories in multiple Logic Blocks. However, all memory in an NFP operates as part of a global address space. The DSF enables high-bandwidth and low-latency data transfers between any data processing element and memory. Therefore, application logic on processing cores and other processing elements on the DSF can operate with relaxed coherence, with a single copy of all data. Any processing element operates on a single copy of the data. Relaxed coherence eliminates the costs associated with hardware or software to maintain multiple coherent copies of data.

The DSF combined with Processing Memory enables data to be processed in-situ through remote commands from processing elements. This reduces the need for data movement for data-intensive compute.

2.6 Scaling and Reuse

The concepts used in the NFP architecture are potentially relevant to accelerating any data-intensive application. Specifically, cross-chip functions and relaxed coherence can enable power-efficient chips for applications such as machine learning (learning and inferencing) and distributed data processing as in Big Data.

- The relaxed coherence model for data vs. control can be extended to inferencing, which like networking requires processing data from multiple independent flows.
- The DSF Design Block can be used to enable inferencing ASICs with a high-bandwidth fabric. Data conditioning on ingress in the Enhanced Networking Interface Logic Block can be used to accelerate distributed data applications.
- Processing at memory can be used to reduce data movement in high fan-out applications.

For example, a multiply-accumulate (MAC) unit added to a processing memory can accelerate learning applications.

Figure 5 shows an example of how new and existing Logic Blocks can be combined to form a new product. The distributed implementations for the DSF and Processing Memory reduce the impact of new Logic Blocks on the elements of a SoC that are reused. Many varieties of Logic Block contents can be reused as GDS II if needed. Also, individual Logic Blocks, such as the Network Processor instances or the Enhanced Host Interface, can be upgraded without perturbing the rest of the system.

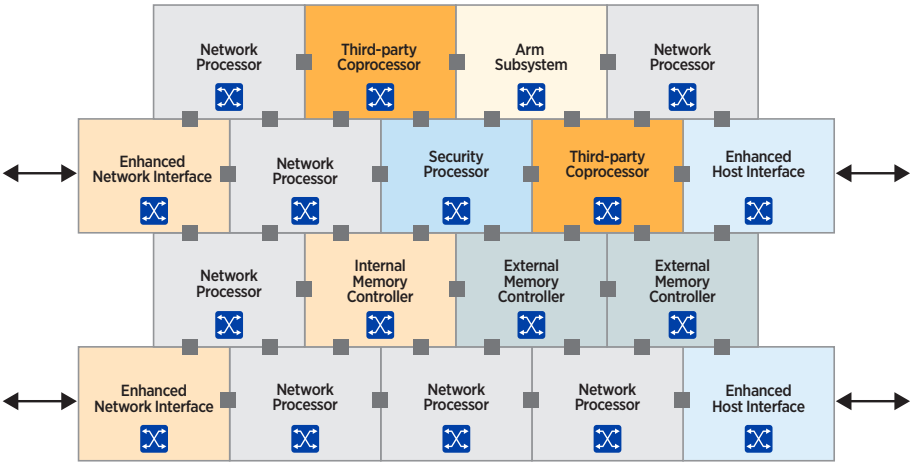

Figure 5. Adding third-party coprocessor IP Blocks

3. CONCLUSION

The increasing importance of coprocessors in new server architectures is driving the need for more complete and sophisticated IP blocks for use in modern SoC designs. While coprocessors are optimized to accelerate specific workloads such as networking and machine learning or inferencing, the efficiency of data movement in a SoC design has a significant bearing on performance and costs related to silicon die. Netronome has developed a unique Composable Architecture to meet these requirements. In this architecture, special cross-chip functions are implemented as functions across cooperating distributed Design Blocks. The composable architecture is used to implement a scalable distributed switch fabric and processing memory. The composable architecture enables efficient data movement and relaxed coherence to improve the power-performance of a programmable architecture for data-intensive applications. The composable architecture offers higher bandwidth, design and data-movement efficiency and scalability relative to conventional implementations. The IP blocks in the NFP SoC can be replaced and upgraded independently to scale the architecture in function and/or performance. The architecture can be reused in new silicon devices to accelerate data-intensive applications such as machine learning and distributed data processing.

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