

Chapter 2

JUDICIOUSLY USING BENCHMARKING

Mel Tsai¹, Niraj Shah¹, Chidamber Kulkarni¹, Christian Sauer²,
Matthias Gries¹, Kurt Keutzer¹

¹*University of California at Berkeley
Electronics Research Laboratory*

²*Infineon Technologies
Corporate Research
Munich, Germany*

Related work in benchmarking research usually focuses on uni-processor performance only. In this case, the optimization of computing performance is the primary concern. Common benchmarks contain an executable specification of the application with corresponding data sets for this purpose. We would like to extend benchmarking to evaluating full systems, such as programmable platforms. Due to the diversity and heterogeneity of system architectures and constantly evolving applications, it is currently a challenge to compare or evaluate systems quantitatively. In this chapter, we present principles of a benchmarking methodology that aim to facilitate the realistic evaluation and comparison of system architectures. A key aspect of our approach is the separation of function, environment, and the means of measurement within the benchmark specification. We pay particular attention to system-level interfaces and define a normalizing environment to quantitatively compare programmable platforms. Looking at the example of network processor architectures, we present a set of representative benchmarks and motivate their selection based on a taxonomy of network equipment. Finally, we provide a proof-of-concept of our methodology by specifying and implementing three benchmarks on a network processor.

1. Introduction

The most established application domain of benchmarking is general-purpose and high-performance computing. In these domains benchmarks are mainly being used to evaluate the quality of computing resources. Application-specific platforms however, such as the Philips Nexperia™ for media processing and the Intel IXP for network processing, have to consider additional constraints on timing, costs, and power dissipation since they are usually used as embedded system. As a consequence of these tight constraints, their system architecture consists of several and heterogeneous components, varying from hard-wired co-processors to programmable cores. We can no longer assume that apart from computation all other components are overprovisioned, as general-purpose computing benchmarks do.

We believe that a disciplined methodology is needed to bring architectural aspects and constraints of the environment into benchmarking. Since programmable platforms can be customized to the anticipated application domain, and the requirements can vary considerably from application scenario to application scenario, the accuracy of the benchmark specification becomes increasingly important. Particularly, two important goals of system-level benchmarks are:

- They must provide results that are *comparable* across different platforms.
- They must provide results that are *representative* and *indicative* of real-world application performance.

Benchmarking plays a critical role in the architectural development process of application-specific instruction processors (ASIPs). In the development of an ASIP, architects can apply a benchmarking methodology that facilitates the evaluation of a prospective architecture and thus enables efficient exploration of the space of potential architectures.

Compared with traditional benchmarking of general purpose processors, we particularly recognize the following issues with providing a system-level benchmark for an application-specific domain:

- *Heterogeneous system resources* – Application-specific platforms achieve performance through a number of architectural mechanisms, which are not necessarily similar across different platforms. In order to maintain comparability of results, a benchmark specification cannot always allow complete freedom in the use of resources.
- *Performance classes* – In the realm of DSPs, for instance, one would not compare an inexpensive low-power DSP to the latest sub-gigahertz DSP designed for multi-channel base stations. These two DSPs are designed for different target systems and exist at two different power and price points. Yet, both processors can perform the same kind of algorithms.

- *Interfaces* – In the domain of network processing, for instance, networks utilize a number of transport media and protocols, including SONET, Frame Relay, ATM, Ethernet, FDDI, and Token Ring. Thus, network processors have become increasingly flexible in the types of network interfaces they support. Unfortunately, every NPU is different, and so are their supported interfaces.
- *Choosing the right metrics* – Platform performance can be measured in a number of ways. Typical metrics include throughput, latency, resource usage, and power consumption. However, there are a variety of other performance metrics that are useful. These include derived metrics, such as cost effectiveness (performance/cost), ease of programming and debugging, and API support. Thus, the set of possible ways to evaluate performance quickly becomes large, and the “right” evaluation criteria may differ across architects, system designers, and customers.

To meet the goals of system-level benchmarking, we present a methodology that allows benchmark results to be comparable, representative, and indicative of real-world application performance. Comparability of results is achieved by precise specification and separation of benchmark functionality, environment, and means of measurement. Functionality captures the important aspects of the benchmark’s algorithmic core. In contrast, the environment supplies the normalizing test-bench in which the functionality resides and allows results to be compared across platforms. Finally, guidelines for the measurement of performance ensure that quantitative results are consistently measured across implementations.

The specification of benchmark functionality, environment, and means of measurement is only one part of the methodology. For benchmarks to be representative of real-world applications, a correct way to select benchmarks is important. Thus, not only does our methodology describe how to select a realistic suite of benchmarks based on an application-domain analysis, but also how to choose the appropriate granularity of a benchmark so that results will be indicative of real-world performance.

The rest of this chapter is organized as follows: Next, in Section 2 we describe our generalized benchmarking methodology. As a demonstration, we apply it to the network processing domain in Sections 3 and 4. Section 5 presents our proposed NPU benchmarking suite and includes an example benchmark specification for IPv4 packet forwarding. We demonstrate results for the Intel IXP1200 network processor in Section 5.3. In Section 6 we summarize previous and ongoing work relating to benchmarking. Finally, this chapter is concluded in Section 7.

2. A Benchmarking Methodology

In this section, we motivate and present principles of a generalized benchmarking methodology that apply to programmable platforms in general. Next, we demonstrate these principles for network processors and illustrate our methodology on an IPv4 packet forwarding benchmark.

Before introducing our methodology, we first define the meaning of a benchmark.

DEFINITION 2.1 (BENCHMARK) *A benchmark is a mechanism that allows quantitative performance conclusions regarding a computing system to be drawn.*

Complete benchmarking methodologies produce benchmark results that are comparable, representative, indicative, and precisely specified, as discussed in the following sections.

2.1 Comparable

A quantitative performance claim that a benchmark provides is not enough. Benchmarks also need to be comparable across system implementations with heterogeneous interfaces. The benchmark specification cannot simply be a functional representation of the application; it must also model the system environment.

- (1) *Quantitative benchmark results must be comparable across a range of system implementations.*

2.2 Representative

To properly characterize a particular application domain, a benchmark suite must cover most of the domain and provide results that correlate to real-world performance. While the number of benchmarks chosen should adequately represent the application domain, the usefulness and feasibility of implementation is questionable for large benchmark suites. A suite of benchmarks must be chosen based on careful application-domain analysis.

- (2) *A set of benchmarks must be representative of the application domain.*

2.3 Indicative

A representative benchmark will not necessarily produce results that are indicative of real-world application performance. For benchmark results to be indicative, the benchmark must be specified with the right granularity. The granularity of a benchmark is its relative size and complexity compared to other applications in a particular domain. In the domain of network processing,

NetBench and NPF-BWG (see Section 6), for example, implement benchmarks according to three different granularities: Small (micro-level), medium (IP or function-level), and large (application or system-level).

The right benchmark granularity is determined by finding the performance bottlenecks of the application and architecture. In some cases, a particular subset of the application (e.g. a small application kernel) may dominate the performance of the larger application. However, in many cases bottlenecks cannot easily be identified or are heavily influenced by the architecture on which the application is implemented. In such cases, choosing a benchmark granularity that is too small may lead to performance results that are not indicative of real-world application performance.

- (3) *The granularity of a benchmark must properly expose application and architectural bottlenecks of the domain.*

2.4 Precise Specification Method

A key component of our methodology is the precise specification of a benchmark. A benchmark specification requires functional, environment, and measurement specifications communicated in both a descriptive (e.g. English) and an executable description.

To separate concerns of benchmarking, we distinguish between functional, environment, and measurement specifications.

- The *functional specification* describes the algorithmic details and functional parameters required for the benchmark. This specification is independent of the architecture.
- In contrast, the *environment specification* describes the system-level interface for a specific architectural platform. This ensures comparability of results across multiple architectural platforms. The environment specification also defines the test-bench functionality, which includes stimuli generation.
- The *measurement specification* defines applicable performance metrics and the means to measure them. At a minimum, a method to determine functional correctness should be provided.

These specifications should be communicated in two forms: A descriptive (e.g. English) specification and an executable description. The English description should provide all of the necessary information to implement a benchmark on a particular system. This description also provides implementation guidelines, such as acceptable memory/speed trade-offs (e.g. it may be unreasonable for a benchmark to consume all on-chip memory), required precision of results, and acceptable use of special hardware acceleration units. Besides the

English description, the specification includes an executable description. This allows rapid initial evaluation, precise functional validation, and facilitates unambiguous communication of the requirements. Note that neither description eliminates the need for the other. We state the final tenet of our benchmarking methodology:

- (4) *Each benchmark should include functional, environment, and measurement specifications in an English and executable description.*

3. A Benchmarking Methodology for NPUs

Due to the heterogeneity of platform architectures, it is necessary to separate benchmarking concerns to ensure comparability of results. In this section, we present our benchmarking methodology for network processors as an illustrative example that separates the functional, environment, and measurement specifications.

3.1 Functional Specification

The role of the functional specification for our NPU benchmarking methodology is three-fold:

First, it must specify the benchmark requirements and constraints, such as the minimum allowable routing table size supported by the implementation, the minimum quality of service (QoS) that must be maintained, or restrictions on where tables can be stored. Requirements and constraints may vary significantly across the suite of benchmarks.

Second, the functional specification must describe the core algorithmic behavior of the benchmark along with any simplifying assumptions in an English description.

Finally, the functional specification should include an executable description written in the Click Modular Router Language [130]. While the algorithmic and functional behavior of a benchmark can unambiguously be communicated using standard languages such as C or C++, Click is a parallel, modular, and natural language for the rapid prototyping of network applications. Using Click, the programmer connects elements (application building blocks such as IP routing table lookups, queues, and packet sources/sinks) and runs simulations on this description to design and verify network applications. Click is also extensible; new elements can be added to serve the needs of different applications.

3.2 Environment Specification

Previous approaches to NPU benchmarking lack environment specifications that are critical for comparability. We present different models to specify the environment: The NPU line card model, the gateway model, and the NIC model.

These models overcome many of the difficulties associated with comparing benchmark results from different network processors.

Currently, major targets of network processors are core routers and edge network equipment. In these segments, routing and switching are the most common tasks. These tasks are performed by systems based on line cards with switched back-plane architectures. A close examination of how a network processor sits in a line card assists in defining the boundary between functionality and environment. Later we illustrate the gateway model, more suited to the low-end access segment of network equipment.

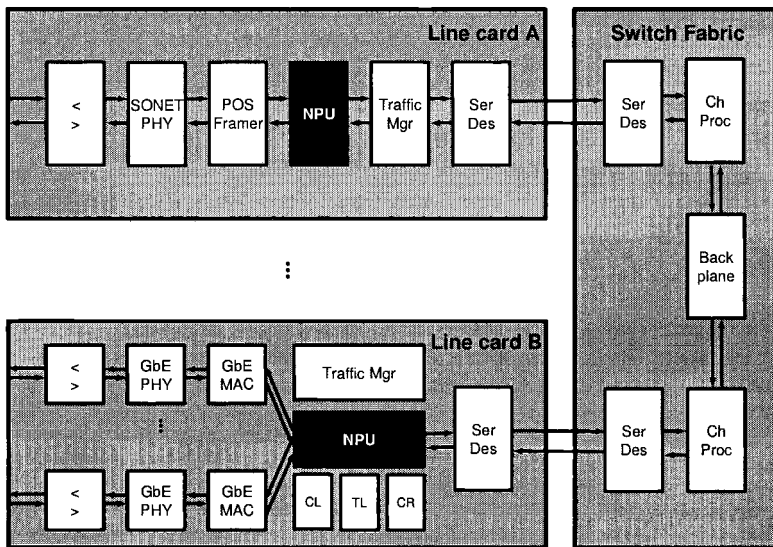


Figure 2.1. An example of a router architecture.

Line cards manage the physical network interfaces of a router. Typical router architectures, such as the one presented in Figure 2.1, contain a variable number of router line cards [52, 119]; up to 16 or more line cards are included in larger routers. Each line card is also connected to a switch fabric that passes packets to other line cards or control processors.

Figure 2.1 demonstrates two different line card deployment scenarios. The network processor in line card A is configured in a serial fashion to its surrounding components, while the network processor in line card B is connected to multiple Gigabit Ethernet MAC units in a parallel fashion. Network processors within line cards may have different interface configurations. In this example, line card A supports an OC-192 packet-over-SONET interface, while line card B supports 10x1 Gigabit Ethernet interfaces.

Table 2.1. Common router line card configurations.

<i>Number of Ports</i>	<i>Type of Ports</i>	<i>Max Bandwidth (Aggregate), in Mb/s</i>
16	Fast Ethernet	100 (1,600)
24	Fast Ethernet	100 (2,400)
48	Fast Ethernet	100 (4,800)
8	Gigabit Ethernet	1,000 (8,000)
16	Gigabit Ethernet	1,000 (16,000)
8	OC-3 POS	155 (1,242)
4	OC-12 POS	622 (2,488)
4	OC-12 ATM	622 (2,488)
1	OC-48 POS	2,488 (2,488)
4	OC-48 POS	2,488 (9,953)
1	OC-192	9,953 (9,953)

3.2.1 Interfaces. In the environment specification it is vital to capture interface differences between line cards. Without capturing these aspects, comparability of benchmark results is not ensured because the network processor is not doing the same work if its line card configuration is different. In Figure 2.2, the system-level and architectural interfaces of an NPU are further illustrated. The system-level interfaces consist of the network (physical and fabric) and control interfaces; these interfaces directly influence the benchmark specification due to their tremendous impact on benchmark functionality and performance. Accordingly, this chapter focuses on system-level network interfaces of an NPU. The memory and coprocessor interfaces are architectural interfaces that are not visible at the system-level, and are not considered here.

Based on an examination of over twenty NPU interfaces [218], we found two major distinguishing characteristics: 1) Some NPUs integrate network interfaces, while others do not and 2) of those that do, the supported interfaces vary greatly. For instance, some network processors contain integrated MAC units for Ethernet interfaces and some do not. If the network processor does not include an integrated MAC unit, one must be added to the hardware or software simulator. If the network processor includes an integrated MAC, it must be configured to conform to the specification. The MAC unit(s) must support the required per-port data rate and number of ports. Further network interface restrictions may be specified by individual benchmarks (e.g. buffer size limits).

For the environment specification, the network interfaces in Figure 2.2 have three important parameters: The number of ports, the type of ports, and the

speed of each port. The functional specification must also be made aware of these parameters as they directly impact functionality. The number and type of active ports in the system must specifically be defined in order to program where, when, and how packets flow within the network processor. In most cases, optimal performance cannot be achieved without customization of the benchmark software to suit a particular port configuration. For example, a programmer of an Intel IXP1200 must treat slow ports (i.e. 10/100 Fast Ethernet) significantly differently than fast ports (i.e. Gigabit Ethernet) [108].

3.2.2 Line Card Configurations. Line card configurations from router vendors, such as Cisco [52] and Juniper Networks [119], have a wide range of performance requirements, as shown in Table 2.1. Hence, we propose two classes of line card port configurations - one that targets lower performance line card deployment scenarios and one that targets higher performance scenarios. For this reason, each benchmark should specify two valid port configurations: One that models a low-end line card configuration and one that models a high-end configuration.

While each benchmark may specify its own port configuration, a uniform configuration should ease implementation complexity. A standardized port configuration also facilitates the understanding of the overall performance of a network processor.

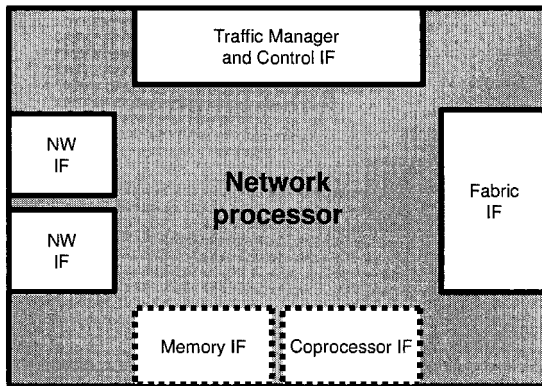


Figure 2.2. Network processor interfaces.

For the low-end line card configuration, we recommend a standard configuration of 16 Fast Ethernet ports, a relatively low-bandwidth configuration supported by many router vendors. For the high-end line card configuration, we recommend a standard configuration of four OC-48 packet-over-SONET ports (10 Gb/s of aggregate bandwidth). OC-48 interfaces are common in high-end line card configurations [171].

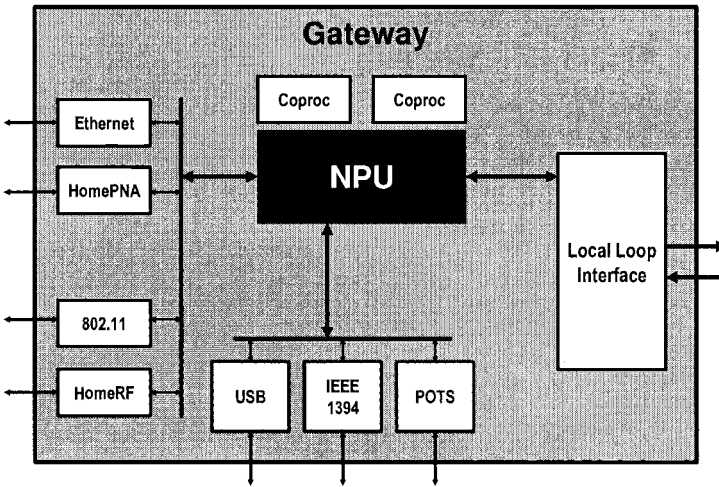


Figure 2.3. Gateway environment model.

While the NPU line card model is a realistic environment for the core and edge segments, it does not accurately represent equipment within the access segment like gateways. A gateway acts as a boundary between an external network (e.g. the Internet) and an internal network (e.g. home or office network). Rich application services are performed in a gateway such as line termination, DHCP, network address translation, and firewall security. To represent this type of functionality and environment we define the gateway deployment scenario, presented in Figure 2.3. A handful of network processors target home and office gateway equipment [106, 34, 114].

The network processor inside the gateway acts as a centralized connection point for internal network devices to the local loop interface. The gateway connects multiple interface media such as wireless (e.g. 802.11b) and wireline (e.g. Ethernet) devices to an external broadband connection (e.g. xDSL or Cable).

Based on Figure 2.3, three important distinctions can be drawn between the gateway deployment scenario and the line card deployment scenario. First, in the gateway, the internal network interfaces are heterogeneous. Second, in addition to packet data, the gateway must also support non-packetized data from interfaces such as USB, IEEE 1394, or even analog phone line connections. Third, NPUs designed for gateway applications, such as those from Brecis [34], include payload-processing accelerators to support DSP (e.g. Voice over IP) and encryption (e.g. 3DES) tasks. A network processor without these accelerators or co-processors may be unable to perform such tasks. Thus, comparing network processors using the gateway model will not only require accounting for interface differences, but also differences in payload accelerators and co-processors.

In the future, a potential application of network processors is the Network Interface Card (NIC). A NIC connects a PC or workstation to a LAN, and has two primary interfaces: The network interface and the PC bus interface. Today, NICs are low-cost ASIC-based products designed for simple buffering and data transfer. However, evolving applications such as virtual private networks (VPN) at large data rates ($>1\text{Gb/s}$) will place greater processing requirements on the PC. Offloading of tasks to the NIC may be a necessary result of tomorrow's network processing requirements. An area of future work will be to develop the NIC environment model for network processors.

3.2.3 Traffic Sources and Sinks. Additional considerations of the environment specification are the packet sources and sinks. The IETF Benchmarking Methodology Workgroup recommends [32] exercising network equipment using the following test setup: A tester supplies the device under test (DUT) with packets, which in turn sends its output back to the tester. In this case, the tester serves as both a packet source and sink. Alternately, a packet sender (source) supplies packets to the DUT, whose output is fed into a packet receiver (sink) for analysis. Either approach will serve the needs of our benchmarking methodology, as long as the packet traces meet the requirements of the benchmark environment specification.

According to the IETF, the data set for network interconnected devices, such as routers and switches, must include a range of packet sizes, from the minimum to maximum allowable packet size according to the particular application or network medium (i.e. Ethernet). Specifically, they recommend using evenly distributed Ethernet packet sizes of 64, 128, 256, 512, 1024, 1280, and 1518 bytes [32].

For protocols that use variable size packets, network processor vendors often report packet throughput using minimum-size packets. Because many network applications operate primarily on fields in the packet header, more processing must be performed on a large group of minimum-size packets than on a small group of large packets. Using only minimum-size or maximum-size packets can test the corner cases of benchmark performance, but these are not realistic indicators of overall performance and we defer to the IETF's recommendations and Newman's work for packet sizes.

Newman [171] observed packet sizes based on live Internet samples from the Merit network over a two-week period. The top four IP packet sizes were 40, 1500, 576, and 52 bytes (56%, 23%, 17%, and 5% of the total, respectively). Newman uses this proportion of packet sizes to develop a traffic pattern called the "Internet mix" (Imix).

Applications are still evolving in the gateway deployment scenario. Benchmarking for the gateway deployment scenario requires workload characterization and remains an open problem.

3.3 Measurement Specification

A benchmark implementation must be functionally correct before performance can be measured. In many cases, functional correctness can be observed by comparing the trace output of the network processor to the output of a reference implementation (e.g. the Click executable description). However, other measures of functional correctness (such as real-time constraints or allowable packet drop rate) may also be specified by the benchmark.

Once a benchmark implementation is shown to be functionally correct, its quality of results can be measured. For many network processor benchmarks, line speed (throughput) is the most important measurement of performance. There are two different units used to measure line speed: Packets per second and bits per second. While the former provides insight into computational performance, the latter provides a clearer understanding of throughput. However, with knowledge of the packet size distribution, one can be converted to the other.

According to RFC 1242 [31], throughput is defined as the maximum rate at which none of the offered packets are dropped by the device. We extend this definition so that some packets may be dropped in accordance with the benchmark specification. The measurement of throughput is challenging. According to [32], throughput is measured by sending packets to the DUT at a specific rate. If packets are incorrectly dropped by the DUT, the rate is successively throttled back until packets are no longer dropped. Using this procedure in our methodology ensures that the measurement of throughput is consistent across implementations.

Besides line speed, there are other useful performance metrics. For example, in a benchmark with real-time constraints, packet latency may be an important performance metric. Derived metrics such as cost effectiveness (i.e. performance/cost) may also be important. The notion of time is central to the measurement of many of these metrics. At the very least, a cycle-accurate software simulator of the network processor architecture is required.

First introduced by NPF-BWG [11] and discussed by Nemirovsky [170], the concept of headroom is loosely defined as the amount of available processing power that could be used by other tasks in addition to the core application. In theory, headroom is useful to evaluate support for additional functionality on top of the core application. Unfortunately, headroom is difficult to define and measure.

4. An Example Benchmark Specification

To illustrate our methodology we present a benchmark specification in the domain of network processing for IPv4 packet forwarding. Using our tem-

plate for network processor benchmarks shown in Figure 2.4, we motivate and summarize the specification for this benchmark.

4.1 Functional Specification

Our functional specification of this benchmark is based on RFC 1812, Requirements for IP Version 4 Routers [16]. The main points of the English description are:

Functional Specification

- Requirements and Constraints
- Behavior
- Click Implementation

Environment Specification

- Network Interface
- Control Interface
- Traffic Mix and Load Distribution

Measurement Specification

- Functional Correctness
- Quality of Results

Figure 2.4. Template for NPU benchmarking.

- A packet arriving on port P is to be examined and forwarded on a different port P'. The next-hop location that implies P' is determined through a longest prefix match (LPM) on the IPv4 destination address field. If P = P', the packet is flagged and forwarded to the control plane.
- Broadcast packets, packets with IP options, and packets with special IP sources and destinations are forwarded to the control plane.
- The packet header and payload are checked for validity, and packet header fields' checksum and TTL are updated.
- Packet queue sizes and buffers can be optimally configured for the network processor architecture unless large buffer sizes interfere with the ability to measure sustained performance.
- The network processor must maintain all non-fixed tables (i.e. tables for the LPM) in memory that can be updated with minimal intrusion to the application.
- Routing tables for the LPM should be able to address any valid IPv4 destination address and should support up next-hop information for up to 64,000 destinations simultaneously.

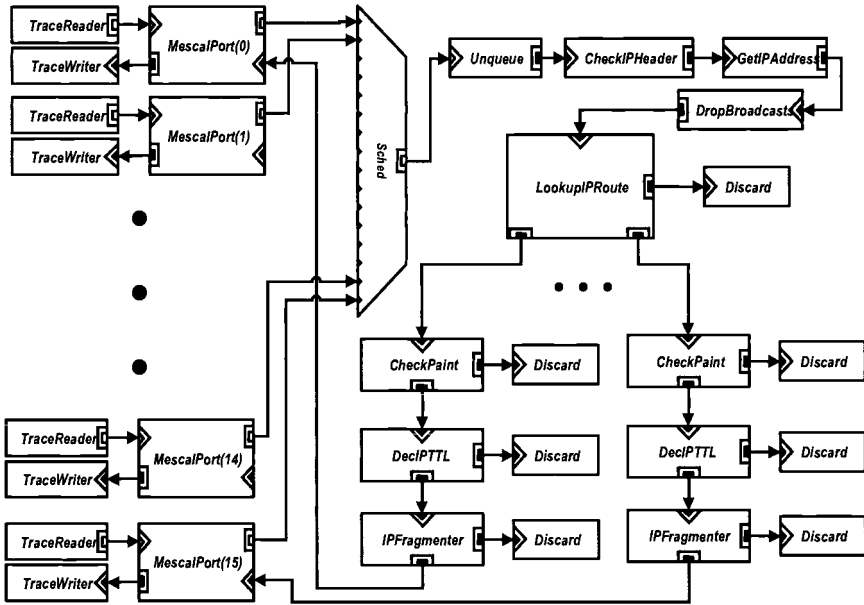


Figure 2.5. IPv4 packet forwarding Click diagram (low-end configuration).

In addition to the written description, Figure 2.5 shows the graphical representation of the accompanying Click description. This Click description also includes components of the environment specification (see below).

4.2 Environment Specification

For the network interfaces, we require the recommended 16 Fast Ethernet ports for the low-end configuration and four OC-48 POS ports for the high-end configuration. A simplifying assumption of our benchmark is that it is a stand-alone configuration. As a result, we do not require a fabric interface.

We define a simple control interface that drops all incoming control packets and does not generate any control packets.

The traffic mix for our benchmark contains destination addresses evenly distributed across the IPv4 32-bit address space. Packet sizes are evenly distributed across 64, 128, 256, 512, 1024, 1280, and 1518 bytes, and 1% of the packets are hardware broadcast packets [32]. In addition, 1% of the packets generate IP errors.

There is a single packet source for each input port that generates an evenly distributed load. Also, the range of destination addresses and associated next-hop destinations provide an evenly distributed load on every output port. Figure 2.5

contains packet sources and sinks that model network interfaces of the low-end configuration.

4.3 Performance Measurement

In order to prove the functional correctness of a benchmark implementation, one must compare all output packets (including error packets) from the Click description to that of the DUT.

Overall performance should be measured as the aggregate packet-forwarding throughput of the network processor such that no valid packets are dropped. This is measured by successively lowering the bandwidth at which packets are sent to each MAC port simultaneously until the network processor does not drop valid packets. This measurement is obtained by using packet traces and port loads specified by the environment.

5. The NPU Benchmark Suite

After having shown the specification of one particular benchmark, we would like to define a representative set of benchmarks for a defined deployment scenario. Before we introduce our suite of benchmarks for network processors, we first discuss benchmark granularity for our methodology.

5.1 Granularity

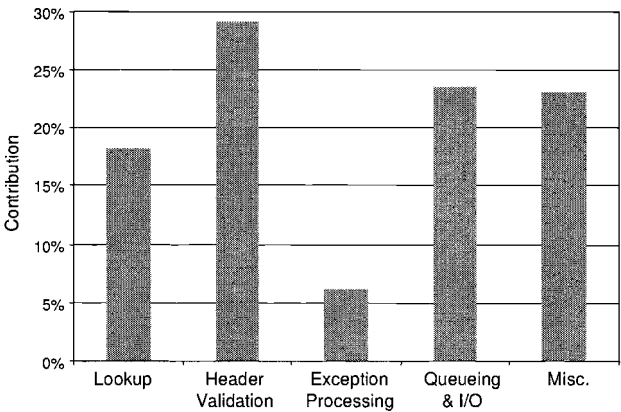


Figure 2.6. Profile for IPv4 packet forwarding on the IXP1200 (64 byte packets).

As introduced in Subsection 2.3, the appropriate benchmark granularity must be chosen after careful application-domain analysis and the identification of performance bottlenecks. Previous benchmarking approaches often chose benchmarks with relatively small granularities, e.g. micro-level. In our experience,

typical applications running on a network processor must perform a large number of diverse tasks on packets. Nearly all such applications must perform micro-level operations, such as input queuing, table lookups, bit-field replacements, filtering rule application, appending bytes to packets (resizing), and checksum calculations. While it is possible that any one of these functions may become a performance bottleneck, it is difficult to determine a priori which micro-level tasks will dominate a particular application on a particular network processor. As an example, Figure 2.6 shows a micro-level breakdown of IPv4 packet forwarding benchmark on the IXP1200 for packet traces consisting only of 64 byte packets. Even when processing small packets, which heavily stresses the header processing ability of a network processor, LPM routing lookup consumes only 18% of the processing time on an Intel IXP1200 microengine. As Figure 2.6 shows, no single micro-level task dominates execution time, thereby supporting the chosen granularity of this benchmark.

We believe the performance bottlenecks of network applications are not appropriately represented by micro-level or function-level benchmarks. Hence, network processor benchmarks should be specified at the application-level.

5.2 Choosing the Benchmarks

Some existing approaches to NPU benchmarking define criteria for differentiating the characteristics of benchmarks (i.e. header vs. payload) and choose benchmarks based on these criteria. However, we strongly believe that the choice of benchmarks should be driven by an application-domain analysis. For this analysis, we created a classification of applications based on network equipment and chose benchmarks that were characteristic of these classes. This equipment-centric view provides a representative set of applications to evaluate system architectures.

We identify three major segments within the network application domain. First, equipment designed for the Internet *core segment* are generally the highest throughput devices found in networking. Core equipment is usually responsible for routing high-speed WAN/MAN traffic and handling Internet core routing protocols, such as BGP.

Second, the *network edge segment* comprises equipment that operates at lower speeds (with shorter links) than equipment in the core. These devices include a variety of mid-range MAN packet processors, such as routers, switches, bridges, traffic aggregators, layer 3-7 devices (such as web switches and load balancers), and VPNs/firewalls.

Finally, the *network access segment* is usually comprised of low- to high-speed LAN equipment. This segment includes equipment such as LAN switches (Fast Ethernet, Gigabit Ethernet, FDDI, Token Ring, and so on), wireless de-

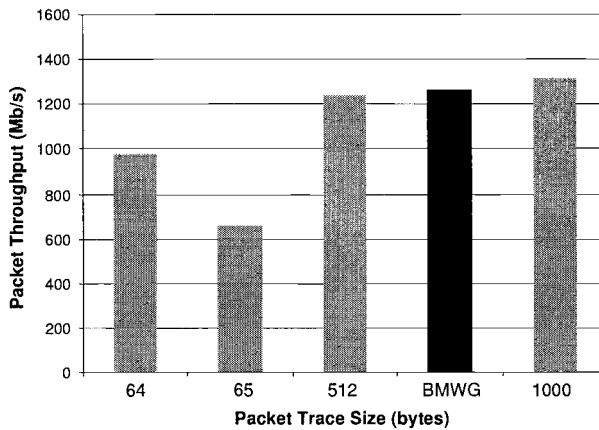


Figure 2.7. IPv4 packet forwarding throughput on the IXP1200.

vices (802.11 a & b), integrated access devices, access concentrators, and cable modems.

Based on this classification, we identify five major application categories in which network processors are likely to play a role. Ten benchmarks were chosen based on their relative significance within the network.

- *LAN/WAN Packet Switching*
 - Ethernet Bridge
 - IPv4 Packet Forwarding
 - ATM Switch
 - MPLS Label Edge Router
- *Layer 3+ Switching*
 - Network Address Port Translation (NAPT)
 - HTTP (Layer-7) Switch
- *Bridging and Aggregation*
 - IP over ATM
 - Packet over SONET Framer
- *QoS and Traffic Management*
 - IPv4 Packet Forwarding with QoS and Statistics Gathering
- *Firewalls and Security*
 - IPv4 Packet Forwarding with Tunneling and Encryption

5.3 Benchmarking Results for Intel IXP

We have completed three benchmark specifications for demonstrating the benchmarking suite: Apart from IPv4 packet forwarding described earlier, we also implemented MPLS and NAPT. These benchmarks were implemented in assembly language on the Intel IXP1200 network processor.

To specify our benchmark functionality and environment we augment an English language description with an executable specification in the Click Modular Router framework [130]. Describing network processor benchmarks using Click has certain key advantages. First, the intended functionality of the benchmark can be verified using Click's Linux[®]-based simulation tools. Second, the extensibility of Click allowed us to write a number of new elements and tools. For example, we have written a port element that models the ports of a router line card. Port elements are used in conjunction with new elements that read and write packet trace files (in a format similar to TCPDump [158]) and supply the Click simulator with realistic packet data from our packet-generation utility (written in C).

The same packet trace interface used by the Click simulator is used for the network processor simulation environment, which aids verification of benchmark functionality. In our benchmark experiments with the IXP1200, a Windows dynamic-linked library was written to interface with the IXP1200 Developers Workbench tools.

Performance on the IXP1200 was measured using version 2.0 of the Developer Workbench software assuming a microengine clock rate of 200 MHz and an IX Bus clock rate of 83 MHz. Intel's IXF440 MAC devices are modeled within the Developer Workbench and configured for 16 Fast Ethernet ports.

Figure 2.7 shows the results for IPv4 packet forwarding. A variety of packet sizes were tested, including the mix of packet sizes recommended by the IETF Benchmarking Methodology Workgroup (BMWG) in [32]. With 16 100 Mb/s ports, the maximum achievable throughput is 1600 Mb/s, yet for the range of packet sizes the IXP1200 cannot sustain this throughput due to the computational complexity of the benchmark specification.

As shown in Figure 2.8, the IXP1200 achieves noticeably higher packet throughput for the range of packet sizes on Network Address Port Translation (NAPT). Incoming packet headers are used to hash into an SRAM-based session table, an operation that benefits greatly from the IXP1200's on-chip hash engine.

As shown in Figure 2.9, the IXP1200 achieves the highest performance on our MPLS benchmark. This is due to the lower computational requirements of our MPLS specification compared to the IPv4 and NAPT benchmarks. This benchmark exercises all three modes of an MPLS router (ingress, egress, and transit) in a distributed fashion.

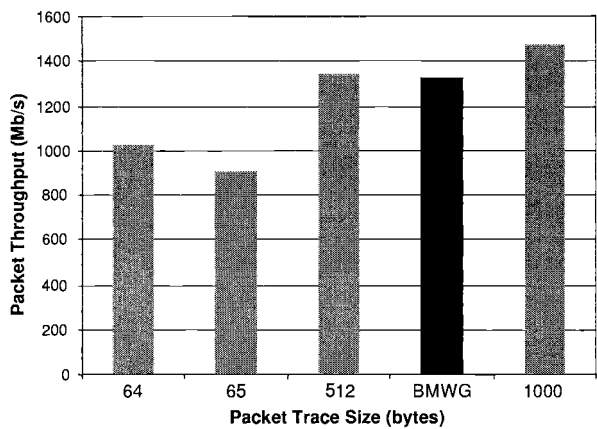


Figure 2.8. NAPT on the IXP1200.

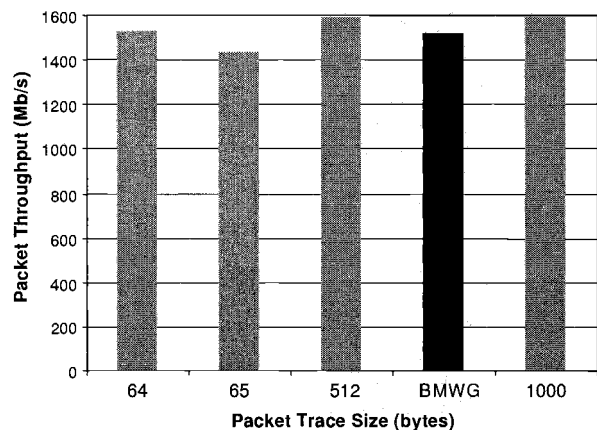


Figure 2.9. MPLS on the IXP1200.

In theory, as packet size increases, packet throughput should also increase. In practice, we observe a reduction in throughput between 64 and 65 byte packets, 128 and 129 byte packets, and so on across all three benchmarks. This is due to the 64 byte alignment of the receive and transmit FIFOs on the IXP1200. Extra processing and transmission time are required for non-aligned packet segments.

This type of information provides insight into the architectural nuances of network processors. It can be seen that actually several different benchmarks representing the application domain are required to analyze the sensitivity of the platform to certain architectural choices. Judiciously using benchmarking

is thus an integral part of a systematic development of ASIPs and programmable platforms.

6. Related Work on Benchmarking

Available benchmarks can be classified according to their application domain. Examples are:

- *General purpose computing:* Benchmarks for general-purpose and scientific computing are published by the Standard Performance Evaluation Corp. (SPEC¹). The Business Applications Performance Corporation (BAPCo²) focuses on benchmarks for personal computers and notebooks.
- *Embedded systems:* Benchmarks for embedded systems including automotive, telecommunication, consumer, and control systems can be found in MiBench [91] and are also defined by the Embedded Microprocessor Benchmark Consortium (EEMBC [61]).
- *Multimedia-centric computing:* Benchmarks focusing on multi-media processing can be found in MediaBench [142] and in the DSP-centric BDTI benchmarks from Berkeley Design Technology, Inc.
- *Database and transaction processing:* Business-oriented transactional server benchmarks are defined by the Transaction Processing Performance Council (TPC³). SPECjAppServer2002 is a client/server benchmark from SPEC for measuring the performance of Java enterprise application servers in end-to-end web applications.
- *Parallel Computing:* Examples of benchmarks for parallel computing and multi-processing are SPEC OMP (OpenMP Benchmark Suite), Stanford Parallel Applications for Shared Memory (SPLASH2 [247]), and PARallel Kernels and BENCHmarks (Parkbench⁴).
- *Network processing:* Related work on defining benchmarks for network processors include CommBench [245], NetBench [160], and activities of the Network Processor Forum (NPF⁵). Related work on disciplined approaches for evaluating network processors can be found in [45]. The Internet Engineering Task Force (IETF) has a work group on benchmarking methodologies (BMWG) of internetworking technologies.

¹<http://www.spec.org>

²<http://www.bapco.com>

³<http://www.tpc.org>

⁴<http://www.netlib.org/parkbench>

⁵<http://www.npforum.org>

Popular benchmarking approaches, such as SPEC and MediaBench, work well because their intended architectural platform is homogeneous. In other words, these and other related approaches focus solely on the functional characteristics of the application and do not emphasize the system-level aspects (e.g. interfaces) of the architectural platform. As a result, these approaches do not work well for application domains with widely heterogeneous architectures.

Benchmarking efforts related to system-level architectures can be found in the context of embedded systems and network processing: CommBench, EEMBC, MiBench, NetBench, NPF Benchmarking Working Group, Linley-Bench, and by Intel Corp. In general, these approaches do not emphasize the system-level interfaces of a system. Since the performance of, for instance, a network processor is heavily dependent on system-level interfaces, such as the network and control interfaces, it is crucial to account for such differences in a corresponding benchmarking methodology. For example, how does one compare the Freescale C-PortTM C-5 (which contains a programmable and flexible on-chip MAC unit) to the Intel IXP1200 (which does not contain an on-chip MAC unit)? Without special consideration of these environmental issues, fair comparisons between the C-PortTM C-5 and the IXP1200 cannot be drawn. As determined earlier, judiciously using benchmarking also requires a specification methodology, an executable description, a method for performance measurement, and a motivation for selection of benchmarks. In order to underpin the necessity of the principles, we will have a closer look at some of the more recent benchmarking efforts.

CommBench

CommBench [245] specifies eight network processor benchmarks, four of which are classified as “header-processing applications” and the other four as “payload-processing applications”. CommBench motivates their choice of benchmarks, but provides no methodology of specification or consideration of system-level interfaces. In addition, the implementation and analysis of CommBench currently targets general-purpose uniprocessors. It is unclear how their benchmark suite can be applied in the context of heterogeneous platforms.

EEMBC

The Embedded Microprocessor Benchmark Consortium (EEMBC [61]) defines a set of 34 application benchmarks in the areas of automotive/industrial, consumer, networking, office automation, and telecommunication domains. Due to this wide focus, the benchmark suits per application domain are not necessarily representative. In the networking domain, for instance, EEMBC defines only three simple benchmarks (Patricia route lookup, Dijkstra’s OSPF algorithm, and packet flow between queues) with little notion of a method-

ology that can be applied to network processors. The benchmarks include a measurement specification.

MiBench

MiBench [91] follows a concept similar to EEMBC, but is composed from freely available sources. A set of 36 task-level benchmarks is defined in areas of automotive/industrial, consumer, office, networking, security, and telecommunication. The benchmarks include small and large data sets for each kernel as workload.

NetBench

NetBench [160] defines and classifies a set of nine network processor benchmarks according to micro-level (e.g. CRC32), IP-level (e.g. IPv4 routing), and application-level (e.g. encryption) granularities. By using this classification, NetBench acknowledges the heterogeneous aspects of network processor micro-architectures and has results for the SimpleScalar [13] simulator and three results for the Intel IXP1200. However, as with EEMBC and [245], NetBench does not provide a methodology that considers the heterogeneity of platform interfaces.

NPF Benchmarking Working Group

The NPF Benchmarking Working Group (NPF-BWG, [11]) defines benchmarks based on micro-level, function-level, and system-level applications. As of this writing, three function-level benchmarks are available: IP forwarding, MPLS, and a switch fabric benchmark. NPF-BWG addresses environment and system-related benchmarking issues, such as workload and measurement specifications. However, no source code is available.

LinleyBench

The Linley Group benchmark⁶ builds on the IPv4 definition of the NPF-BWG. In addition, DiffServ classification and marking, and an optional ATM-IPv4 interworking test are defined. To date, results on the IBM network processor are published.

Intel Corporation

Intel's [45] benchmarking approach focuses on the IXP1200 network processor. They define a four-level benchmark hierarchy: Hardware-level, micro-

⁶<http://www.linleygroup.com/benchmark>

level, function-level, system-level. Hardware-level benchmarks are designed to test functionality specific to a particular NPU (e.g. memory latencies). They provide some results for IPv4 forwarding on the Intel IXP1200. Intel's executable description and method for performance measurement are specific to the IXP1200.

Intel's work recognizes the need for benchmark comparability by describing a benchmarking reference platform. This model wraps black-box functionality (the network processor, co-processors, and memory) with a set of media, fabric, and control interfaces. However, these interfaces are left unspecified so that customers can modify the reference platform to suit their particular needs. As we have argued earlier, these interfaces must be appropriately specified because they are critical for benchmark comparability. Intel's published work does not indicate whether they provide a methodology for benchmark specification.

NpBench

Lee and John [141] address the lack of control plane benchmarks for network processing. They define a set of 26 functions in three classes: Traffic management and Quality of Service, security and media processing, and packet processing). These benchmarks also include processing in the data plane and refer to CommBench, EEMBC, MiBench for this purpose.

Other Work

Nemirovsky [170] recognizes the needs and requirements of network processor benchmarking. Although Nemirovsky provides insight into the methods for quantifying network processor performance, he does not provide precise information about a viable benchmarking approach.

Crowley et al. [55] present an evaluation of theoretical network processor architectures based on a programmable network interface model and detailed workload analysis. However, their model does not include many aspects of current target systems for network processors. Also, their model does not fully separate functional and environment concerns.

Summary of Existing Approaches

In Table 2.2 we compare existing approaches to platform benchmarking, particularly for network processors. In this domain, the environment specification includes the specification of external network traffic. The granularity column distinguishes between micro-kernel benchmarks that focus only on the most compute-intense part of an application and more complex function benchmarks.

As this table shows, none of the existing approaches meet all the requirements of a benchmarking methodology. In particular, there are a number of deficien-

Table 2.2. Characteristics of benchmark definitions.

	<i>CommB.</i>	<i>NetBench</i>	<i>EEMBC</i>	<i>NPF</i>	<i>Intel</i>	<i>Linley</i>	<i>MiBench</i>	<i>NpBench</i>
<i>Measurement Specification</i>	No	No	Yes	Yes	IXP1200-specific	Yes	No	No
<i>Traffic Spec.</i>	No	Yes	Yes	Yes	N.A.	Yes	Yes	no
<i>Interface Spec.</i>	No	No	No	N.A.	Yes	No	No	No
<i>Source Code</i>	C	C	C	No	μ E-C	No	C	C
<i>Availability</i>	on req.	free	license	spec only	No	license	free	on request
<i>Granularity</i>	micro	micro/func	micro	function	function	function	micro	micro

cies including: Lack of specification methodology and lack of consideration of interfaces. In order to compare different platforms and achieve functional correctness, executable system benchmarks, environment, and measurement specifications are required.

7. Conclusion

In this chapter we have reviewed existing benchmarking efforts and revealed flaws why they cannot be applied to benchmarking of whole systems. We have derived four principles of a generalized benchmarking methodology considering heterogeneous system architectures:

- Comparability across different system implementations,
- Representativeness of the application domain,
- Indicative of real performance,
- Precise specification.

The methodology defines a template for a benchmark consisting of functional, environment, and measurement specifications. As an illustrative example, we have tailored this methodology to the specific requirements and goals of network processor benchmarking. However, other application domains (e.g. telecommunications, multimedia, automotive) also benefit from this work.

Our key contributions, illustrated for network processing, are the emphasis of the system-level interfaces for NPUs based on our line card model, the utility of a Click executable description, motivation for application-level benchmarks, and a disciplined approach to identifying a set of benchmarks. In summary, we believe we have identified key issues in benchmarking ASIPs as well as programmable platforms and embodied them in a methodology for judiciously using benchmarking.



<http://www.springer.com/978-0-387-26057-0>

Building ASIPs: The Mescal Methodology

Gries, M.; Keutzer, K.

2005, XXXII, 350 p., Hardcover

ISBN: 978-0-387-26057-0