A Comparison of Network Processor Programming Environments

Niraj Shah
William Plishker
Kurt Keutzer

Chess Review
May 10, 2004
Berkeley, CA

Outline

- Motivation
- Study
  - Overview
  - Performance measurement
  - Development Process
  - Resource Usage
- Conclusions
- Future Work

Motivation

- Electronic systems designers want silicon customization
- Application Specific IC design becoming increasingly risky
  - Costly
  - Unpredictable
- Fuels the rise of programmable devices or ASIPs (Application Specific Instruction Processors)
  - Networking
  - Multimedia
  - Graphics
- ASIPs
  - Architectures have been explored in great depth
  - Modest progress on programming environments
  - But, the success of users is dependent on their ability to program these effectively

Existing ASIP Programming Environments

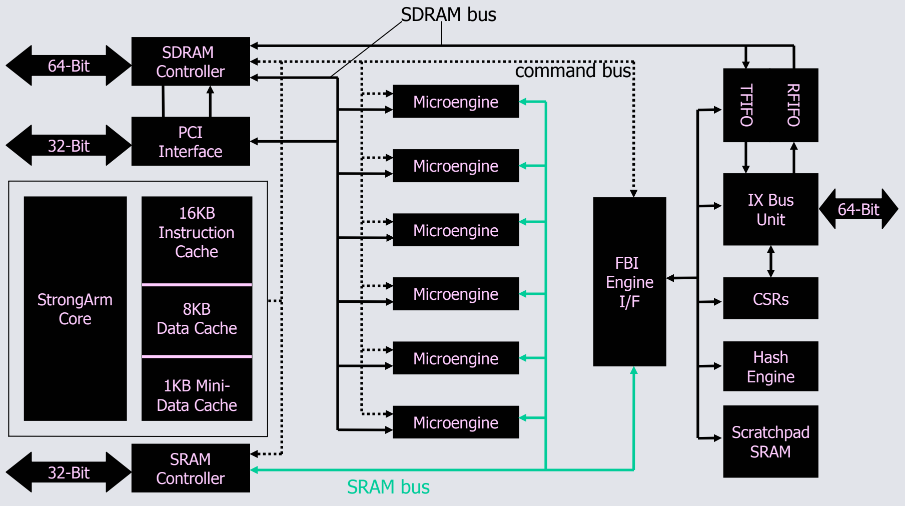
- Macro assembler
- C language variants
  - Subset: no function pointers or recursion
  - Superset: language features for threads, processes, and data allocation
- Higher level programming environments
  - Application domain-specific, block based

The Study

- Focus on Networking*
- Use a representative network processor
- Implement two applications
- With two programming environments
- Compare
  - Achievable performance
  - Development process
  - Resource usage

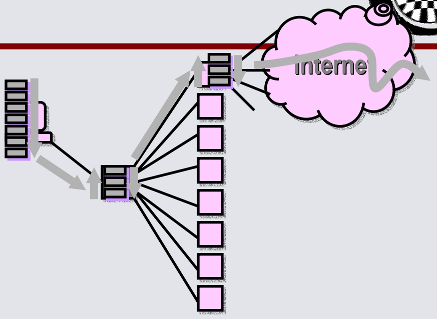
* First presented at the Workshop on Productivity and Performance in High-End Computing (P-PHEC), February, 2004. (N. Shah, W. Plishker, K. Keutzer. *Comparing Network Processor Programming Environments: A Case Study*. 2004)

Representative NP: Intel IXP1200

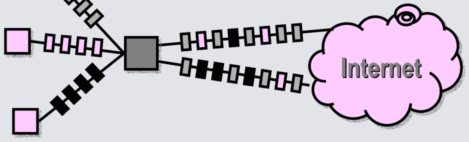


Applications Overview

- IP version 4 (IPv4) packet forwarding
  - Data plane of router
  - Baseline functionality for network equipment



- Differentiated Services (DiffServ)
  - Quality of service in IP
  - Provision network resources for categories of traffic

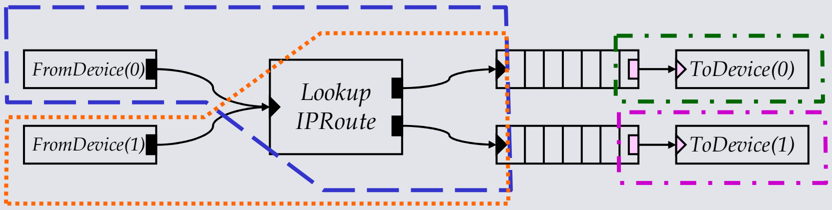


Programming Environment: IXP-C

- Subset of C
  - Loops, conditionals, function calls
  - Basic and abstract data types
- Explicit parallelism – each microengine programmed independently
- Explicit threading model
- User defined data allocation

Programming Environment: NP-Click

- Developed here*
- Click[†] concepts
  - elements
  - communication via push and pull of packets
- Combined with salient features of underlying hardware
  - threads
  - data layout
  - arbitration of shared resources

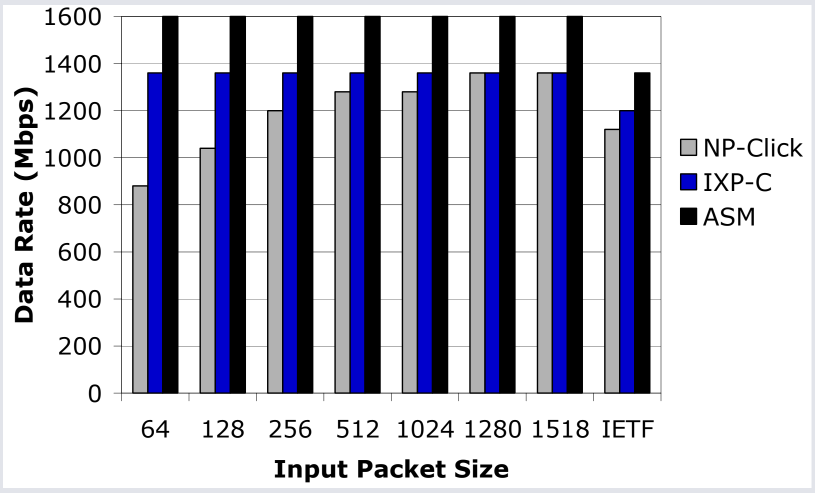


*Niraj Shah, William Plishker, Kurt Keutzer. *NP-Click: A Programming Model for the Intel IXP1200.* P. Crowley, M. Franklin, H. Hadimioglu, P. Onufryk, 9, 181-201, 1, 2, Elsevier, 2004.
†E. Kohler et al. *The Click Modular Router.* ACM Transactions on Computer Systems. 18(3), pg. 263- 297, August 2000.

Evaluation

- IPv4 Performance Measurement
  - 16x100Mbps IPv4 packet forwarder
  - 1000 entry route table
  - Maximum sustainable data rate on a variety of packet sizes as proxy for performance
- DiffServ Performance Measurement
  - 4x100Mbps DiffServ node
  - Ingress: a variety of packet classes (Assured Forwarding, Best Effort, Expedited Forwarding).
  - Measure egress data rate of constituent flows

IPv4 Forwarding Performance



Performance Summary

- IPv4 Forwarding Performance
  - IXP-C: 75-85% of line rate
  - NP-Click: 55-85% of line rate, 93% of IXP-C (IETF)
  - Assembler: Performs at 85-100% of line rate
- DiffServ Performance
  - Both implementations received packets at line rate and perform similarly
  - NPClick within 10% of IXP-C for higher priority flows

Development Process Comparison

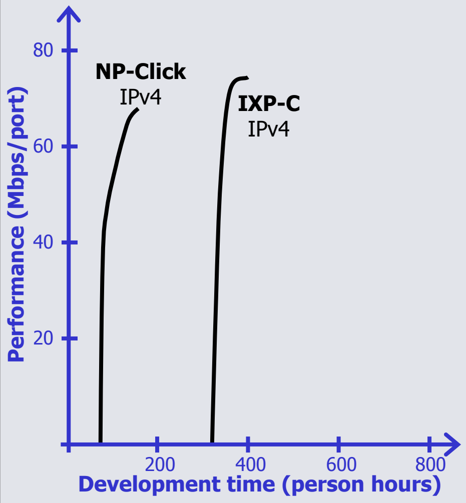
NPClick

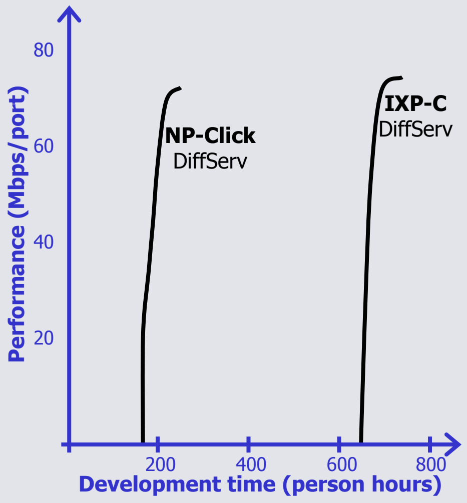
- Functional correctness in a few days
- Poor initial performance
- Most of time spent profiling/optimizing
  - Changing mapping to microengines
  - Element implementations
  - Arbitration schemes
- IPv4: 100 person hours
- DiffServ: additional 120 person hours

IXP-C

- Mostly spent reaching functional correctness
- High initial performance
- Isolating & fixing multi-threading bugs
- Profiling and optimizing difficult
  - Only incremental changes
- IPv4: 400 person hours
- DiffServ: additional 320 person hours

Development Process Summary





Summary: IXP-C

- Principal advantages
  - Resource usage: smaller code size, fewer microengines
  - Performance edge: slightly higher data rate
- Drawbacks
  - Much longer development effort
  - Long time to reach functional correctness
  - Performance tuning difficult → only incremental changes
  - Final implementation largely dependent on initial implementation

Summary: NP-Click

- Primary advantages
  - Very short time to functional correctness
  - Explore design space of implementations
  - 2.6-4x shorter overall design effort
- Disadvantages
  - Resource usage: much larger code size
  - Small performance overhead
    - IPv4: 7% less than IXP-C for IETF traffic mix
    - DiffServ: within 10% for higher priority flows

- More effort using NP-Click spent can reduce performance gap and resource usage

Conclusions

- Embedded software development
  - Performance-focused
  - Effort bound
- Best use of IXP-C
  - Squeeze the most performance from chip
  - Tight resource constraints
- Best use of NP-Click
  - Effort bound projects
  - Little intuition about design space
- These results are applicable to current and newer network processors

Future Work

- Automation of mapping application to architecture
  - Task Allocation was recently done*
  - Memory and Communication are underway
- Perform similar study on newer, larger scale NP architectures
  - Will highlight NP-Click's design space exploration
  - Even more difficult to program in IXP-C

* William Plishker, Kaushik Ravindran, Niraj Shah, and Kurt Keutzer. *Automated Task Allocation on Single Chip, Hardware Multithreaded, Multiprocessor Systems.* Workshop on Embedded Parallel Architectures (WEPA-1), February, 2004.