

# Performance of Embedded System Application on Network Processor

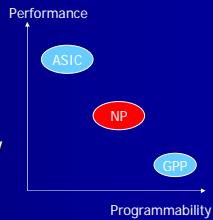
2006 Spring Directed Study Project  
 Danhua Guo  
 University of California, Riverside  
[dguo@cs.ucr.edu](mailto:dguo@cs.ucr.edu)  
 06-07-2006

# Road Map

- Motivation
- Intel IXA NP: A Quick Look
- Environment: Intel IXA SDK
- MiBench Overview
- Experiment

# Motivation

- NP Overview
  - Programmability (GPP)
  - Performance (ASIC)
- Project Motivation
  - Architectural Portability



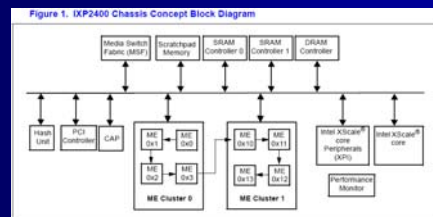
# Motivation (Cont.)

- NP has been proved to be a successful design for packet processing.
- NP architecture (XScale + MEs) is similar to Embedded Processor architecture (ARM + DSP).
- Embedded application performance on NP?

# Road Map

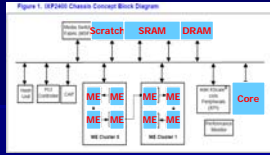
- Motivation
- Intel IXA NP: A Quick Look
- Environment: Intel IXA SDK
- MiBench Overview
- Experiment

# Intel IXA 2400 NP Architecture



Intel Corporation - Intel® IXP2400 Network Processor - 2nd Generation Intel® NPUs

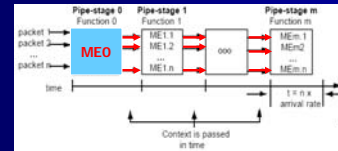
## IXA 2400 NP Resources Summary



Name	Size (Bytes)	Transfer Size (Bytes)	Reference Latency (Cycles)	Application
<b>XScale Core</b>	32-bit general-purpose processor			Initialize and manage the chip
<b>GPR</b>	256*4	4	1	Incurs no penalty for context switch
<b>Xferr</b>	512*4	4	1	Hold data while asynchronous I/O is pending
<b>NWR</b>	128*4	4	1	Transfer data from previous/next neighbor ME
<b>LM</b>	640*4	4	3	Caching data needed by ME
<b>Scratch</b>	16K	4	60	General purpose use with atomic operations and ring support
<b>SRAM</b>	64M	4	90	Control information storage
<b>DRAM</b>	2G	16	120	Data buffer storage

## Programming Model

### Context Pipeline

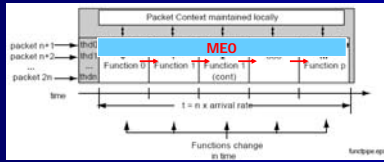


- Context is passed through the pipeline
- A single ME is dedicated entirely to a single function

Intel Corporation - Intel® IXP2400 Network Processor - 2nd Generation Intel® NPUs

## Programming Model (Cont.)

### Functional Pipeline



- Functions are passed through pipeline
- A single ME can constitute a functional pipeline

Intel Corporation - Intel® IXP2400 Network Processor - 2nd Generation Intel® NPUs

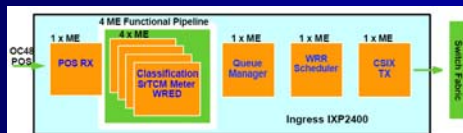
## Programming Model (Cont.)

### Context Vs. Functional

- |   |  |
|---|--|
| <p>+++</p> <ul style="list-style-type: none"> <li>• Good for programs with a large code size</li> <li>• Good for on chip storage of vectors/tables</li> </ul> <p>---</p> <ul style="list-style-type: none"> <li>• Bad for large context passing</li> <li>• All pipe stages must execute at minimum packet arrival rates. (hard to partition)</li> </ul> | <p>+++</p> <ul style="list-style-type: none"> <li>• Supports a longer execution period than context pipe-stages</li> <li>• Exe time for each pipe-stage is flexible</li> </ul> <p>---</p> <ul style="list-style-type: none"> <li>• ME must support multiple functions</li> <li>• Mutual exclusion may be more difficult</li> </ul> |
|---|--|

## Programming Model (Cont.)

- Which one is better?
  - A combination of Context and Functional Model



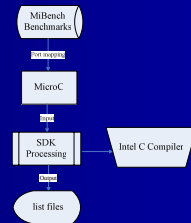
Intel Corporation - Intel® IXP2400 Network Processor - 2nd Generation Intel® NPUs

## Road Map

- Motivation
- Intel IXA NP: A Quick Look
- Environment: Intel IXA SDK
- MiBench Overview
- Experiment

## Environment: Intel IXA SDK Workbench

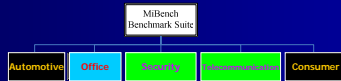
- A software development kit for NP programming
- Input: MicroC or MicroCode
- Configuration: Data Flow, ME distribution, etc.
- Output: .list file for each ME



## Road Map

- Motivation
- Intel IXA NP: A Quick Look
- Environment: Intel IXA SDK
- MiBench Overview
- Experiment

## MiBench Overview



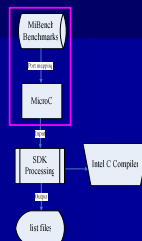
- Type of Embedded Application
  - **Control** Intensive: branch instructions
  - **Computational** Intensive: integer and floating point ALU operations
  - **I/O** Intensive: memory (load and store)

## Road Map

- Motivation
- Intel IXA NP: A Quick Look
- Environment: Intel IXA SDK
- MiBench Overview
- Experiment

## Experiment Outline

- Port map MiBench to SDK
- Configure SDK Environment
- Feed .list into SDK
- Analyze and Evaluate Result



## Port mapping

- Why is it required?
  - The C compiler is not a complete ANSI C implementation.
  - Things the compiler does not support
    - the full standard C runtime library.
    - automatic parallelization of code.
    - C++
    - floating point data types (float and double).
    - function pointers and recursion.
    - functions with a variable number of arguments (varargs).

## Port mapping (Cont.)

- How is it working?
  - Data Allocation
    - Register Regions: GPR, NN, Xfer
    - Memory Regions: LM, Scratch, SRAM, DRAM
    - Allocation attributes (`__declspec`) can describe where the variable is allocated.
  - Register Spillage
    - Default spill order:
      - Next Neighbor registers
      - Local Memory
      - SRAM Memory
  - Functions
    - Inline function (function calling overhead)
    - Optimizing pointer arguments (callee access overhead)

## Port mapping (Cont.)

- Optimizing Your Code
  - Minimize variable allocation to memory.
    - Taking the address of a variable or declaring it with a memory region attribute causes allocation to memory.
    - Structures/arrays larger than 64 bytes (or 128 bytes in 4-context mode) are allocated to memory.
  - Minimize access to memory variables.
  - When possible, declare a variable as a local variable in `main()` rather than as a global or a static variable so that unnecessary initialization is avoided.
  - Efficient Structure Access
    - Sizing of Structure Members
      - A multiple of four bytes in size
      - Falling on a four byte offset from the start of a structure
      - Do not cross a 4 byte boundary

## SDK Configuration

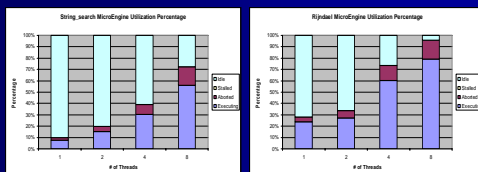
- Library directory
  - Which library to include
- Memory information
  - Which part of memory segments is reserved for variables
- Thread information
  - Which mode to use
  - # of contexts (threads) to run
- ME information
  - Which .list file to be fed into which ME

## Environment

- Intel IXP SDK 4.1 (Select IXP2400)
- 600MHz ME configurations
- 200-MHz SRAMs
- 150-MHz RDRAMs
- Executed in Multi-threads
- Executed in Multi-MicroEngines

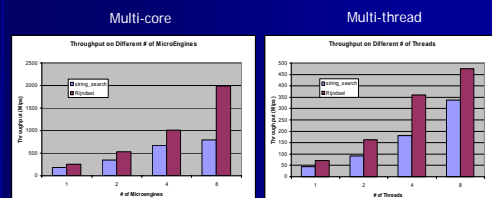
## Experiment Result

- MicroEngine Utilization Percentage
  - On 8-Context Mode on 1 ME



## Experiment Result (Cont.)

- Throughput (Mips)



## References

- Intel C Compiler User's Guide
- Intel IXA Development Tools User's Guide
- Matthew R. Guthaus et. al., *"MiBench: A free, commercially representative embedded benchmark suite"*
- Zhangxi Tan et. al., *"Optimization and Benchmark of Cryptographic Algorithms on Network Processor", 2004*
- Intel Corporation – Intel® IXP2400 Network Processor - 2nd Generation Intel® NPU
- Yan Luo et. al., *"NePSim: A Network Processor Simulator with a Power Evaluation Framework", 2004*
- Dominic Herity, *"Network Processor Programming", <http://www.embedded.com/story/OEG20010730S0053>*

## Thanks to

- Prof. Bhuyan for the idea of simulation
- Prof. Luo for the suggestions on SDK
- Mr. Piyush Satapathy for the experience on MicroC coding
- Everyone for giving up your valuable "afternoon-coffee" time

Questions or Comments?