

# **Peppermint and Sled: Tools for Evaluating SMP Systems based on IA-64 (IPF) Processors**

---

**Sujoy Basu<sup>1</sup>**

**Tom Fisher<sup>2</sup>**

**Sumit Roy<sup>1</sup>**

**Bruce E. Blaho<sup>2</sup>**

**Raj Kumar<sup>1</sup>**

**(1)HP Labs (2)Technical Computing Division**

**Hewlett-Packard Company**



# Introduction

- System architects have large and complex technical applications
- Focus on aspects of system design that have maximum impact on performance
- Applications have already been compiled for IA-64 architecture
- Long traces available from these applications
- Reasonably fast simulation: performance model

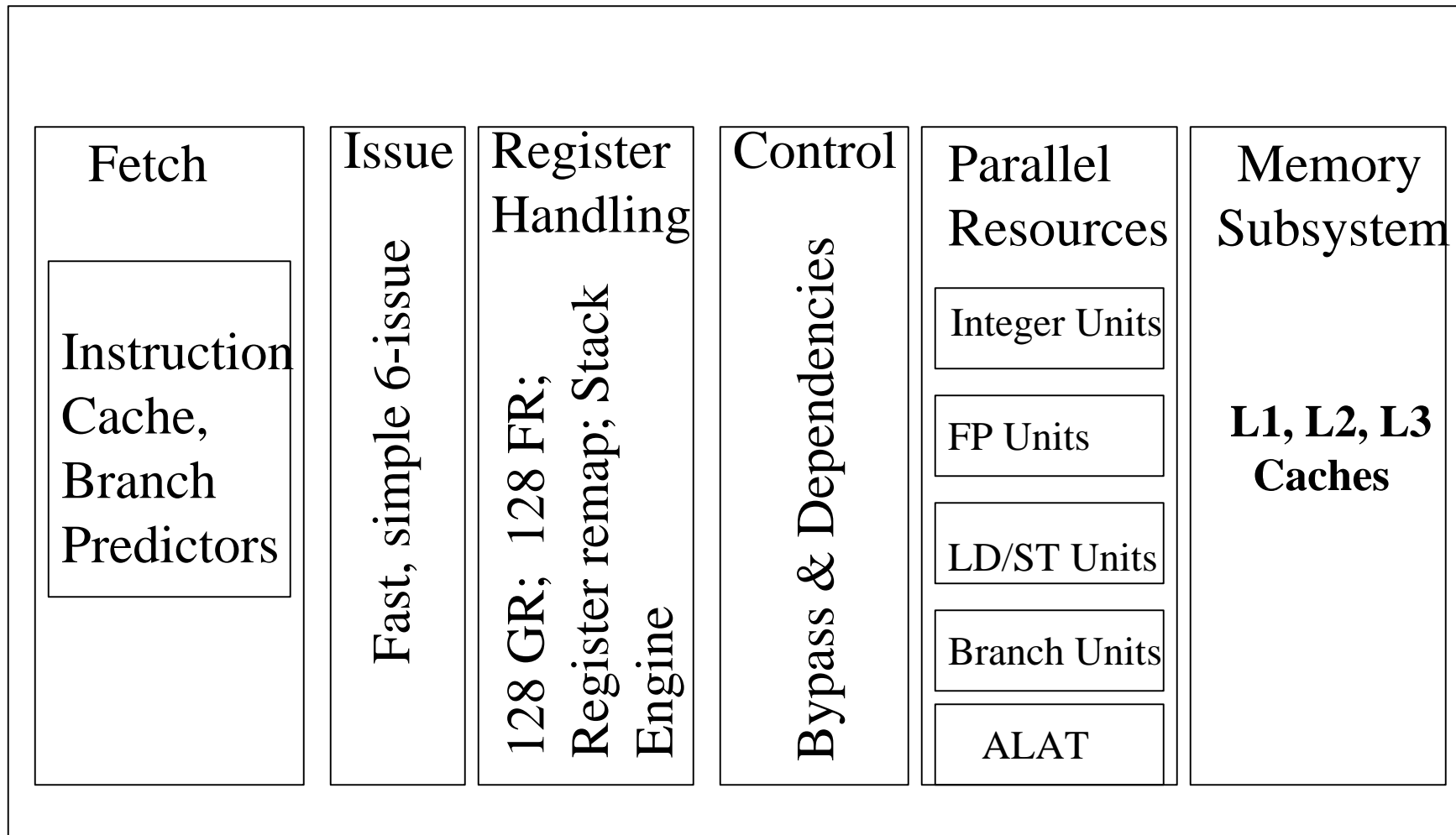
# Peppermint

- Takes traces generated on IA-64 systems as input
- Models components of small SMPs:
  - 1 or more IA-64 processors with 3 levels of cache
  - System bus
  - Memory controller
  - DRAM
- Uses a configuration file to change parameters of these components

# Organization of This Talk

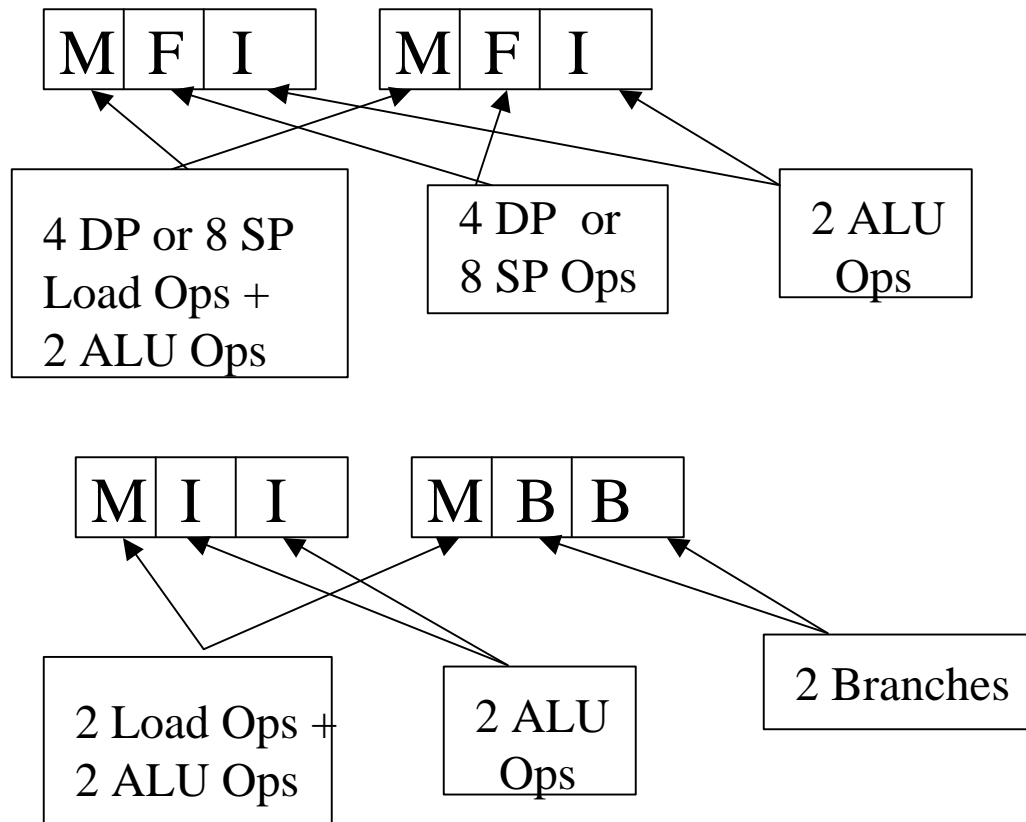
- Peppermint
  - Architectural Model
  - Performance & Validation
- Sled
  - Design Issues
  - Performance
- Itanium vs McKinley
  - Configuration, Applications
  - Simulation Results

# Conceptual View of IA-64 H/W



# Instruction level Parallelism

6 instructions can provide:



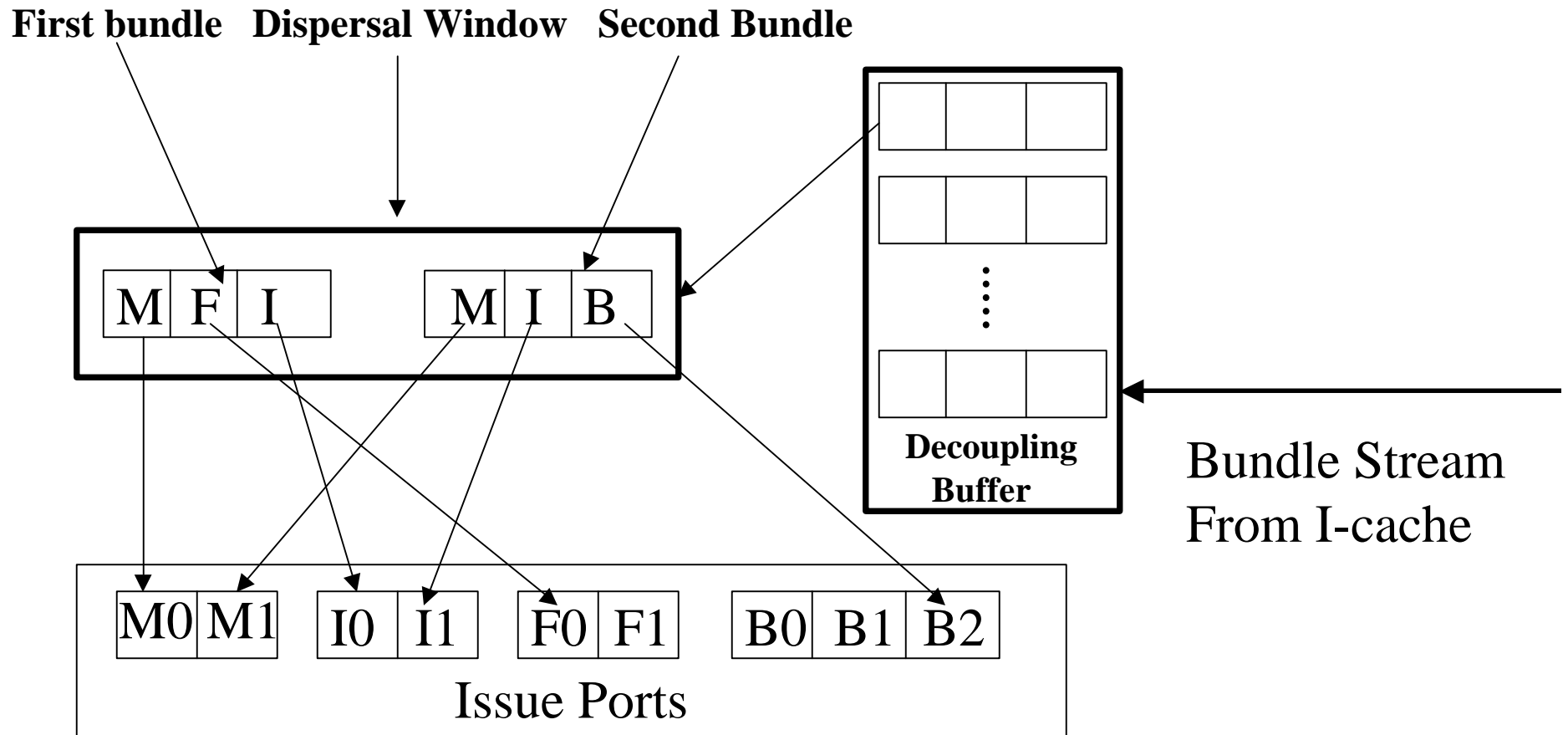
**Total:**

- 20 Parallel SP Ops/Clock
- 12 Parallel DP Ops/Clock

**Total:**

- 8 Parallel Ops/Clock

# IA-64 Instruction Dispersal



# Processor Architecture in Peppermint

- IA-64 instruction dispersal mechanism:
  - Instructions are fetched from I-Cache into the decoupling buffer as long as there is space
  - Instructions are shifted from the buffer into instruction dispersal window every cycle if needed
  - 2 instruction bundles in dispersal window are available every cycle (6 slots)
  - Dispersal rules determine:
    - Which issue port each instruction is sent to
    - When dispersal stops without issuing all 6 instructions

## Examples of Instruction classes supported on issue ports

Instruction Class	M0	M1	I0	I1	F0	F1	B0	B1	B2
ALU (Add, compare)	•	•	•	•					
Sign/zero extend			•	•					
Fixed extract/deposit			•						
Load/Store	•	•							
Memory Management	•								
Fixed Multiply					•	•			
FP Compare					•				
Call/Return							•	•	•
Loop-type branch									•

# Instruction Stalls in Peppermint

```
{.mii
  add  r1=r2,r3 // 0
  add  r4=r5,r6 // 0
  sub  r7=r8,r9 // 0
}
{.mii
  ld4  r14 = [r56] // 0
  add  r15 = r31, r34 // 1
  add  r16 = r18, r19 // 1
}
```



# Instruction Stalls in Peppermint

- Not all 6 instruction slots may be dispersed in any cycle (*split-issue*):
  - × Issue port of right type is not available
  - Explicit stop-bit is inserted by compiler to preserve dependencies
  - Certain sequence of bundles will always split-issue
  - Dependencies between instructions require stall for few cycles
  - Less than 2 bundles in dispersal window, due to underflow of decoupling buffer

# Instruction Stalls in Peppermint

```
{.mii
  add  r1=r2,r3 // 0
  add  r4=r5,r6 ;; // 0
  sub  r7=r8,r9 // 1
}
{.mfi
  ld4   r14 = [r56] // 1
  fadd  f10 = f12,f13 // 1
  add   r16 = r18,r19 // 1
}
```



# Instruction Stalls in Peppermint

- Not all 6 instruction slots may be dispersed in any cycle (*split-issue*):
  - × Issue port of right type is not available
  - × Explicit stop-bit is inserted by compiler to preserve dependencies
    - Certain sequence of bundles will always split-issue
    - Dependencies between instructions require stall for few cycles
    - Less than 2 bundles in dispersal window, due to underflow of decoupling buffer



# Instruction Stalls in Peppermint

- Not all 6 instruction slots may be dispersed in any cycle (*split-issue*):
  - × Issue port of right type is not available
  - × Explicit stop-bit is inserted by compiler to preserve dependencies
  - × Certain sequence of bundles will always split-issue
    - Dependencies between instructions require stall for few cycles
    - Less than 2 bundles in dispersal window, due to underflow of decoupling buffer

# Instruction Stalls in Peppermint

- Not all 6 instruction slots may be dispersed in any cycle (*split-issue*):
  - × Issue port of right type is not available
  - × Explicit stop-bit is inserted by compiler to preserve dependencies
  - × Certain sequence of bundles will always split-issue
  - × Dependencies between instructions require stall for few cycles
  - Less than 2 bundles in dispersal window, due to underflow of decoupling buffer

# Instruction Stalls in Peppermint

- Not all 6 instruction slots may be dispersed in any cycle (*split-issue*):
  - × Issue port of right type is not available
  - × Explicit stop-bit is inserted by compiler to preserve dependencies
  - × Certain sequence of bundles will always split-issue
  - × Dependencies between instructions require stall for few cycles
  - × Less than 2 bundles in dispersal window, due to underflow of decoupling buffer

# Cache Hierarchy in Peppermint

- Cache characteristics such as size, associativity, latency are parameterized
- Integer loads: 2(L1D), 6(L2), 21(L3) cycles
- FP loads: 9(L2) or 24(L3) cycles
- Instruction fetch: 1(L1I), 5(L2), 20(L3) cycles
- Predicated off loads are not simulated
- Advance (speculative) loads are simulated

# System Bus in Peppermint

- Pipelined split-transaction system bus is modeled
- Round-robin arbitration
- Frequency is parameterized
- Bus Width & Number of Cycles required for transfer of a cache line can also be varied

## Memory Controller & DRAM Model

- Bit positions and field width for row address, column address, bank are parameters
- Number of open pages and size of read and write queues are also parameters
- Latencies such as precharge time and RAS to CAS delay are modeled as parameters
- Scheduling policy can be varied by writing function for selecting among queues

# Performance & Validation

- Peppermint
  - Throughput ranged from 11 to 22 K instructions per second on traces of length 1 million
  - Measured on a system with 733 MHz Pentium III processor with 256 Mbytes of RDRAM
  - Will be lower for smaller traces
  - Cache models validated against Dinero
  - Complete system model validated against a similar Itanium-based system within 20%
  - More exhaustive validation intended against a McKinley-based system

- Generates IA-64 trace by:
  - Controlling execution using *ptrace* or *ttrace*
  - Uses hardware debug registers of *Itanium* processor to collect trace between two breakpoints
  - Within breakpoints, the process is single-stepped to collect trace records
  - Single-stepping not required if only performance counters are only sought

# Sled Performance

- In tracing mode, information gathered by Sled includes:
  - Instruction pointer
  - Contents of the instruction bundle
  - Address of load/store, if present
  - Content of predicate register
- Slowdown over native execution: 60 – 200
- Typical range for single-stepping: 100 – 10000
- Acceptable, since the trace is collected once, and simulated several times



## Sled Performance (Contd.)

- On a dual 800 MHz Itanium-based system, slowdown ranged from 60 to 200
- Slowdown reported in the literature for trace collection based on single-stepping ranges from 100 to 10000
- We also observed that the overhead of trace-collection over single-stepping is 20-30%
  - Overhead of system calls
  - Optimization: bundle cache

# Configuration Files

<b>Parameter</b>	<b>Itanium</b>	<b>McKinley</b>
Proc. Frequency	800 MHz	900 MHz
L1 I Line Size	32 bytes	64 bytes
L1 I Assoc.	4	4
L1 I Access	2 cycles	1 cycle
L1 I Size	16 KB	16 KB
L1 D Line Size	32 bytes	64 bytes
L1 D Assoc	4	4
L1 D Size	16 KB	16 KB

# Configurations (Contd.)

Parameter	Itanium	McKinley
L1 D Ports	2	4
L1 D Access	2 cycles	1 cycle
L2 Line Size	64 bytes	128 bytes
L2 Size	96 KB	256 KB
L2 Ports	2	4
L2 Access (Int)	6 cycles	5 cycles
L2 Access (FP)	9 cycles	6 cycles
L2 Assoc	6	8

# Configurations (Contd.)

<b>Parameter</b>	<b>Itanium</b>	<b>McKinley</b>
L3 Size	4 MB	3 MB
L3 Line Size	64 bytes	128 bytes
L3 Associativity	4	12
L3 Access (Int)	21 cycles	12 cycles
L3 Access (FP)	24 cycles	16 cycles
Bus Frequency	133 MHz	200 MHz
Bus Width	64 bits	128 bits
DRAM Freq	100 MHz	150 MHz

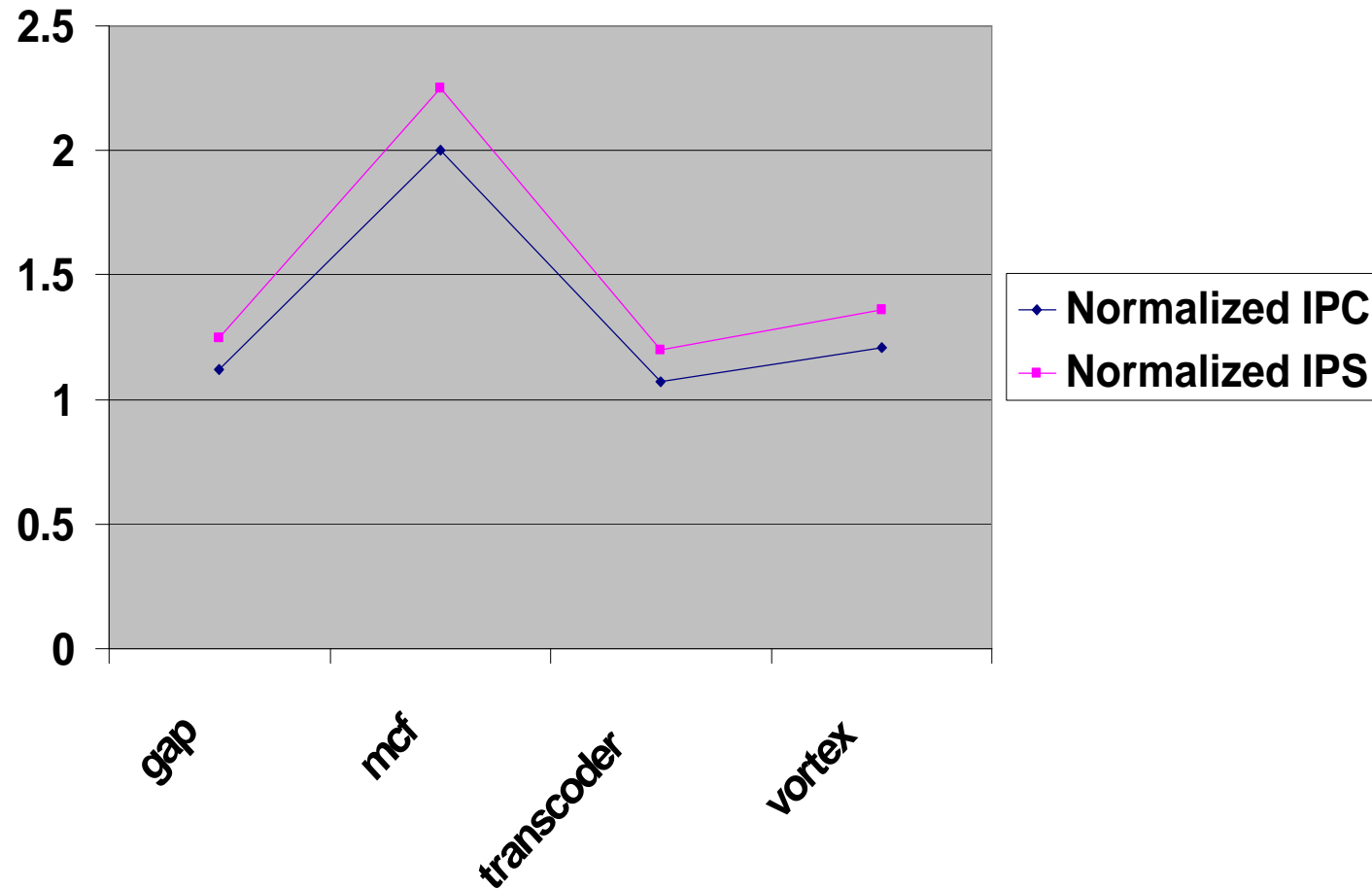
# Configurations (Contd.)

Parameter	Itanium	McKinley
RAS-CAS Delay	20 ns	20 ns
CAS Latency	2 DRAM cycles	2 DRAM cycles
Double Data Rate	No	Yes

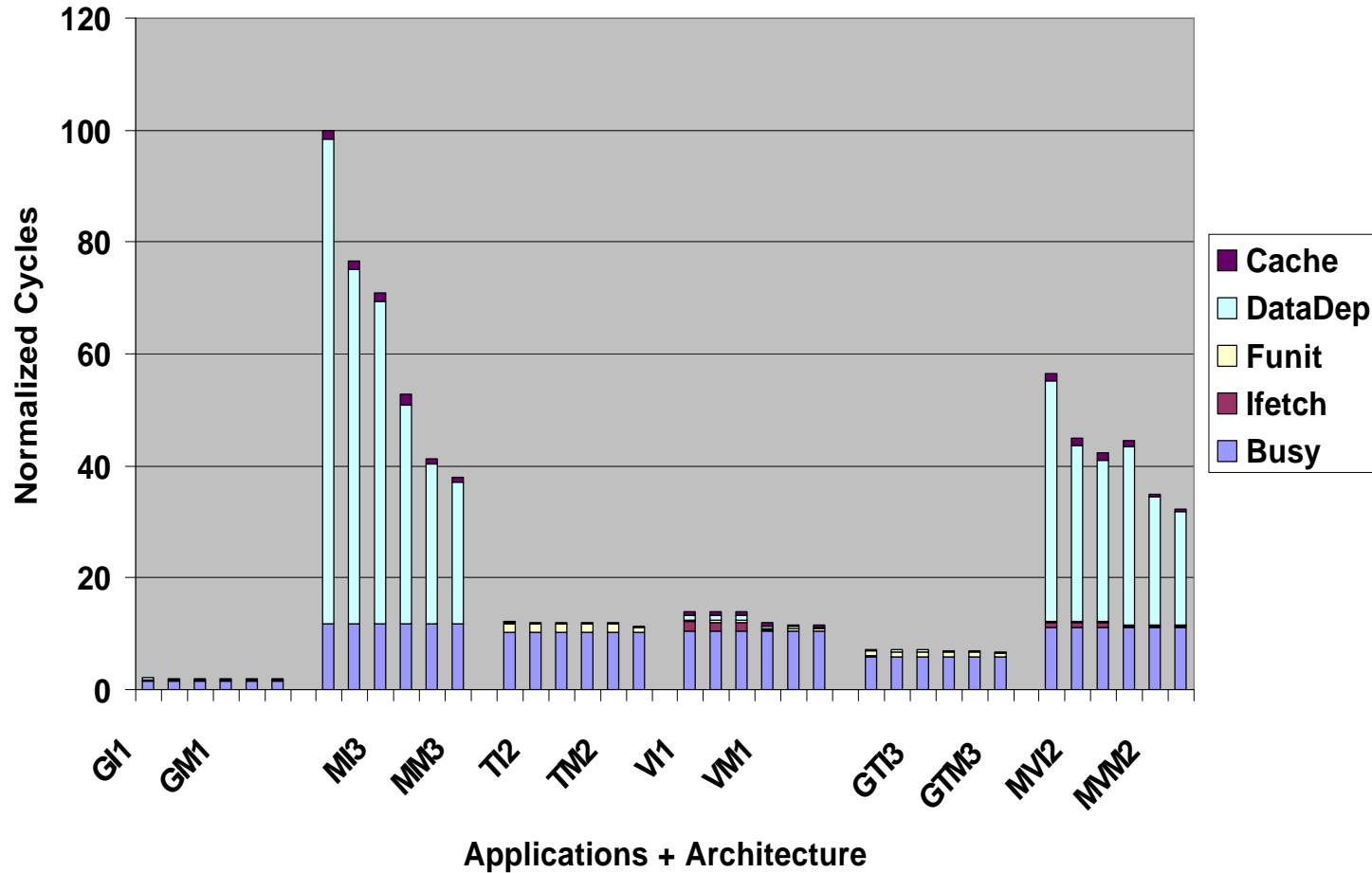
# Applications Used

<b>Application</b>	<b>Description</b>
Mcf (SPEC2000)	<b>Combinatorial optimization; network simplex algorithm</b>
Vortex (SPEC2000)	<b>Lookups, insertions &amp; deletions in object-oriented database; no I/O</b>
Gap (SPEC2000)	<b>Computational discrete algebra</b>
Transcoder (In-house)	<b>Down-scale transcoding of compressed video</b>

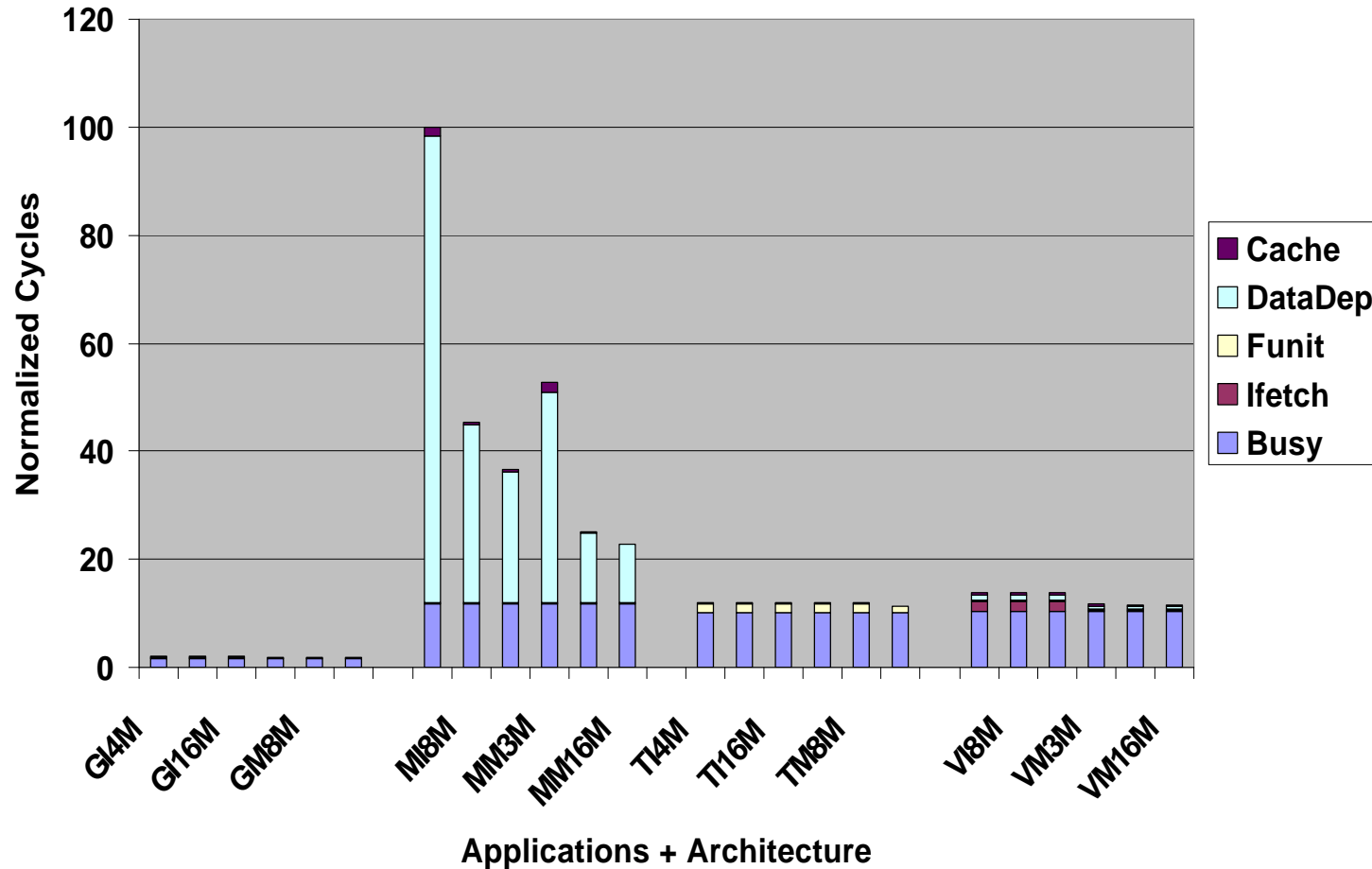
# IPC & IPS: Itn vs Mck



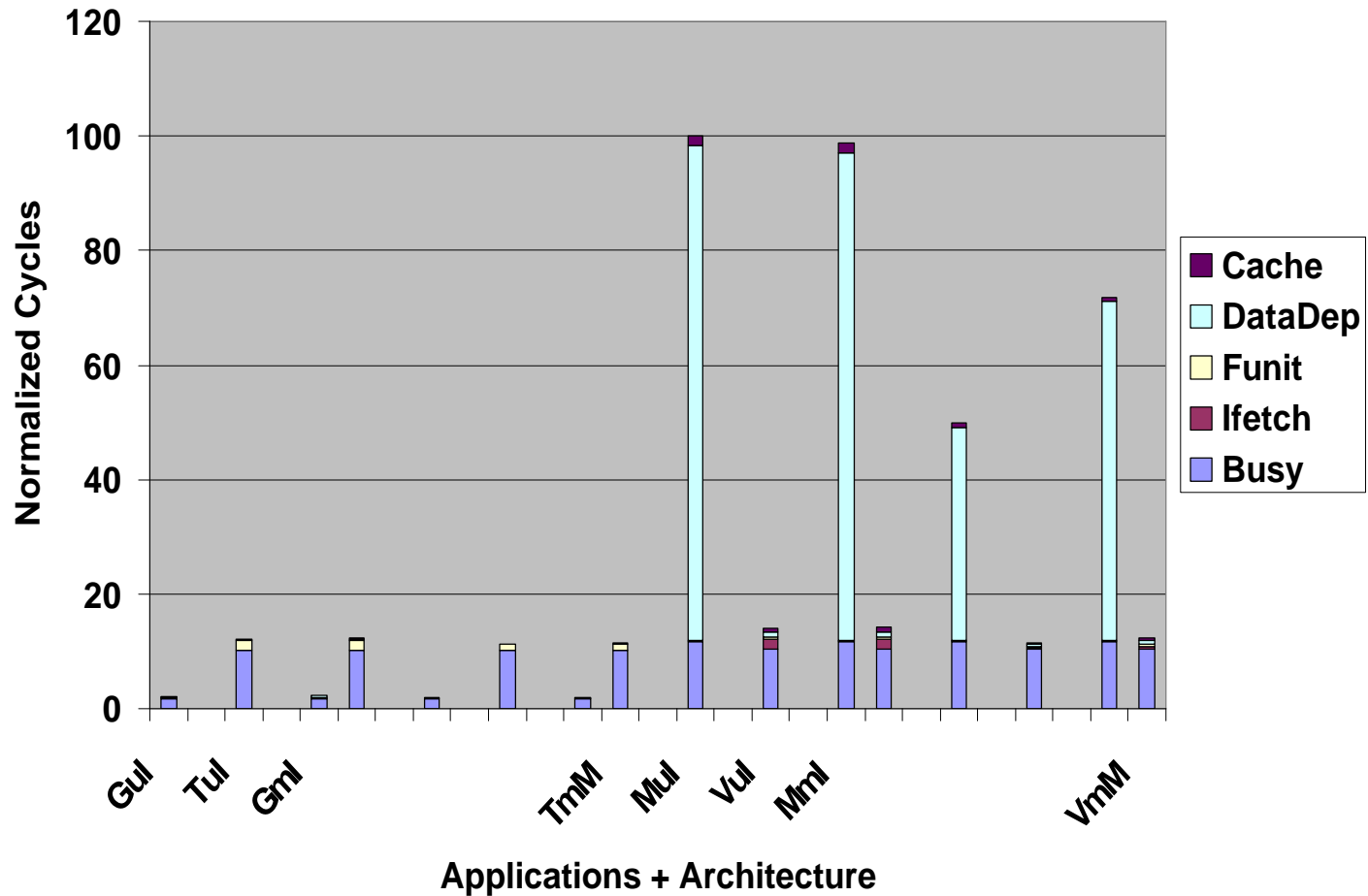
# Execution Time vs Bus Speed



# Exec Time vs L3 Size



# UNI vs SMP



# Summary of Results

- **IPC of McKinley over Itanium can range from 7% to 100%**
- **IPS is even better, due to higher frequency**
- **Compute-intensive applications like gap and transcoder benefit from larger number of functional units in McKinley**
- **Better cache and memory system of McKinley benefit applications like gap:**
  - **Larger cache and cache line size**
  - **DDR SDRAM with higher frequency**
- **High system bus utilization and memory controller queue overflow observed in some SMP runs**

# For more information

- Email me: [sujoy\\_basu@hp.com](mailto:sujoy_basu@hp.com)
- Get HP Labs Technical Report Number HPL-2002-14
  - Go to [www.hpl.hp.com](http://www.hpl.hp.com)
  - Click on “Technical Reports” on the left panel
  - Select “2002 Technical Reports”
  - Select “HPL-2002-14”