The Hobbit Microprocessor for Personal Communicators

Robert J. Scavuzzo

AT&T Bell Laboratories, Inc.
Allentown, Pennsylvania

Hot Chips Symposium
August 10, 1992

---

Agenda

- Personal Communications
- Why Hobbit
- Hobbit History
- CRISP/Hobbit Comparison
- Hobbit Code Density
- Hobbit Performance
- Power Reduction
The AT&T
Personal Communications Vision

Pen Computer
Fax

Phone
Cellular Phone
Video Phone

Anytime, Anyplace Access to Voice, Data, Fax, Video

A Personal Communicator Design

Power Subsystem
Digitizer
Modem
Memory Subsystem
System Controller
Hobbit
Display Subsystem
I/O Subsystem
Microprocessor Attributes Needed for PCS

- High Code Density
- High Performance
- High Integration Level
- Low Power Dissipation
- Low Cost

Hobbit Evolution

1980 C Machine 1985 Now

CRISP Hobbit

C Machine Research Methodology
1) Propose an Architecture
2) Create C Compiler
3) Develop Simulator
4) Evaluate Performance
CRISP/Hobbit Architecture
High Level Attributes

- CISC Like
  - Memory-to-Memory Architecture
  - Variable Instructions
  - High Code Density
- RISC Like
  - Small Instruction Set
  - Pipelined Architecture
  - Single-Cycle Execution
  - High Performance

The CRISP/Hobbit Architecture Combines the Best of CISC and RISC
## CRISP → Hobbit Changes

### Technology/Silicon Design:

<table>
<thead>
<tr>
<th>CRISP</th>
<th>Hobbit</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 V</td>
<td>2 V to 5.0 V</td>
</tr>
<tr>
<td>12 MHz, Selected</td>
<td>8 MHz to 30 MHz Operation</td>
</tr>
<tr>
<td>No Standby Mode</td>
<td>Low Power Standby Mode</td>
</tr>
</tbody>
</table>

### Architecture:

<table>
<thead>
<tr>
<th>CRISP</th>
<th>Hobbit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Big Endian Data Byte Ordering</td>
<td>Big/Little Endian Data Byte Ordering</td>
</tr>
<tr>
<td>512 Bytes, Direct Mapped, Encoded Instruction Cache</td>
<td>3K Byte, 3-way Set Associative, Encoded Instruction Cache</td>
</tr>
<tr>
<td>128 Byte Stack Cache</td>
<td>256 Byte Stack Cache</td>
</tr>
<tr>
<td>No MMU/TLBs</td>
<td>MMU/TLBs</td>
</tr>
<tr>
<td>No Data Loop Back</td>
<td>Data Loop Back</td>
</tr>
</tbody>
</table>
Code Density

- High Code Density
  - Memory-to-Memory Architecture
  - Variable Length Instructions
  - Carefully Crafted Instruction Set

Instruction Size Distribution for PCC

<table>
<thead>
<tr>
<th>Instruction Size</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-Byte</td>
<td>15,000</td>
<td>68%</td>
</tr>
<tr>
<td>6-Byte</td>
<td>6,000</td>
<td>27%</td>
</tr>
<tr>
<td>10-Byte</td>
<td>1,000</td>
<td>5%</td>
</tr>
</tbody>
</table>
### Code Density

#### PCC Comparisons

#### Size Comparisons of Cross Compiled Hobbit PCC

<table>
<thead>
<tr>
<th>Machine</th>
<th>Bytes</th>
<th>Relative</th>
</tr>
</thead>
<tbody>
<tr>
<td>VAX</td>
<td>72K</td>
<td>1.0</td>
</tr>
<tr>
<td>Hobbit</td>
<td>74K</td>
<td>1.0</td>
</tr>
<tr>
<td>M68000</td>
<td>86K</td>
<td>1.2</td>
</tr>
<tr>
<td>R3000</td>
<td>128K</td>
<td>1.8</td>
</tr>
<tr>
<td>IBM 370</td>
<td>134K</td>
<td>1.9</td>
</tr>
<tr>
<td>SPARC</td>
<td>141K</td>
<td>1.9</td>
</tr>
</tbody>
</table>

---

### CRISP/Hobbit

#### Performance Related Features

- High Code Density
- Synchronous but Independent Prefetch Decode and Execution Units
- Single Cycle Instruction Execution
- Encoded and Decoded ICs
  - Stack Cache
  - Fast Procedure Call
  - Branch Folding/Branch Prediction
  - Read Cancelling with Two Stage Bypassing
  - MMU and TLBs
  - Graceful Performance Degradation with Increasing Memory Wait States
CRISP/Hobbit
Performance Related Features

- High Code Density
- Synchronous but Independent Prefetch Decode and Execution Units
- Single Cycle Instruction Execution
- Encoded and Decoded ICs
- Stack Cache
- Fast Procedure Call
  - Branch Folding/Branch Prediction
  - Read Cancelling with Two Stage Bypassing
  - MMU and TLBs
  - Graceful Performance Degradation with Increasing Memory Wait States

Stack Cache

- A Different Approach to Register Allocation
- Automatically Maps the Stack onto a Circular Buffer of Registers
  - Looks like Registers to Hardware
  - Looks like Cache to Software
- Buffer Maintained by Head and Tail Pointers
  - Stack Pointer
  - Maximum Stack Pointer
- Stack Cache Hit if SP ≤ Address < MSP
- Low Order Virtual Address Bits Used as Index
Stack Cache Results

- Captures 80% of Data References
- Three Times Fewer References than VAX PCC
- Best of Registers
  - Fast Access
  - No Tags
  - No Tag Compares
- Best of Cache
  - Better Allocation than Software
  - Simplifies Compiler Design
  - Can Hold Strings, Arrays
  - Transparent of Software ⇒ Scalable

Fast Procedure Call

- Four Instructions Used in Procedure Call
  - **Call**: (1 Clock Tick)
    Places Return Address in SC and Branches to Target Address
  - **Enter**: (1 Clock Tick Minimum)
    Allocates Space for New Procedure
  - **Return**: (2 Clock Ticks)
    Deallocates Current Stack Frame
  - **Catch**: (1 Clock Tick Minimum)
    Guarantees SC is Filled
- Procedure Calls are Typically 5 Clock Ticks
### CRISP/Hobbit

**Performance Related Features**

- High Code Density
- Synchronous but Independent Prefetch Decode and Execution Units
- Single Cycle Instruction Execution
- Encoded and Decoded ICs
- Stack Cache
- Fast Procedure Call
- **Branch Folding/Branch Prediction**
  - Read Cancelling with Two Stage Bypassing
  - MMU and TLBs
  - Graceful Performance Degradation with Increasing Memory Wait States

---

### Branch Problems

- Tend to "Break" Pipeline
- RISC Machines Typically use Delayed Branch
  - 1 Cycle for Branch
  - 1 - 2 Cycles for nop’s
  - Problems with short Basic Blocks
Branch Folding

- ~ 1/3 of Instructions are Branches
- Branch Prediction is ~ 90% Accurate
- PDU Can Simultaneously Decode up to Two Instructions
- In Branch Folding, Target Address of Leading Instruction can be replaced by Target Address of Following Branch Instruction
- Branch Instruction is Executed in "Zero" Time
- Conditional Branches Provided for
  - Alternate Next-Address Field
  - Branch Prediction
  - Branch Spreading
- Reduces Number of Instructions Needed to Execute Program
- MIP Rate > Clock Rate

CRISP/Hobbit
Performance Related Features

- High Code Density
- Synchronous but Independent Prefetch Decode and Execution Units
- Single Cycle Instruction Execution
- Encoded and Decoded ICs
- Stack Cache
- Fast Procedure Call
- Branch Folding/Branch Prediction
- Read Cancelling with Two Stage Bypassing
- MMU and TLBs
- Graceful Performance Degradation with Increasing Memory Wait States
Read Cancelling

- Dependences Cause Problems for Pipelines
  
  \[ a \leftarrow b + 1; \]
  
  \[ c \leftarrow a + 3; \]

- Hazards are Resolved by Bypassing
- Read is Cancelled with Bypass Occurs
- Hobbit Performs Better with Hazards

Low Power Dissipation

- Low Voltage Design
- DC Standby Mode
- Integrated Resources
  - Caches
  - MMU/TLBs
- Reduced I/O
  - Caches
  - Read Cancelling
  - Stack Cache Architecture
- High Code Density
  - Fewer Text I/O Transactions
  - Less System RAM
- Data Loop-Back
  - Resistor-less System Bus
Data Loop-Back

Data In

Input Buffer

Data Loop-Back Control

Input Latch

Data Out

Tri-State Driver

Data Bus

Hobbit for
Personal Communicators

- High Code Density
  - Less RAM
  - Lower Power Consumption
  - Lower System Cost
- High Performance
  - Faster System Response
  - Enables Use of CPU Intensive Applications
- High Integration
  - Lower Power Consumption
  - Lower System Cost
  - Longer Battery Life
  - Lighter/Smaller Systems
- Low Power
  - Longer Battery Life
  - Lighter/Smaller Systems
"The Right Choice"