The New IT Engine –
HISC: Object-Oriented
Computer for the 21st Century

Anthony Fong  Ph.D.
Department of Electronic Engineering
City University of Hong Kong

Confidential and proprietary  -
© 2011 Copyrights; all rights reserved by A. Fong
Computer Evolution

- **1940s** Pioneer – Univac, Control Data, IBM
  Instruction Sets different for individual machines, software incompatible

- **1960s** CISC – IBM System 360 family
  Family of machines → made IBM the industry leader

- **1980s** RISC – PowerPC, MIPS, SPARC
  Filled in the gap between minicomputers and PC → workstations and servers

- **2000s** OOP – HISC !
  Object-Oriented Programming: Java, J2EE, C#, .NET, C++
  Internet computing
Agenda

- Object-Oriented Programming (OOP)
- Problems of today’s computers with OOP
- HISC – Operand Descriptor machine
- jHISC – Architecture, Design, & System Software
- Performance Evaluation and Functionality Comparison
- Project status
- Future development
Procedure-Oriented Programming

- The domination of the programming language styles in the past
- Every user develops his/her own software
  - Less efficient/productivity
  - Quality problems
  - Non-sharable, hard-to-reuse
  - Non-transportable
Object-Oriented Programming (OOP)

- In early 1970s, U.S. DoD commissioned a task force to investigate why her IT budget went out of control, w/o much to show for.

- The findings:
  - 80% budget went to software.
  - More than 80% of software budget went to maintenance.
  - Hardware components need to do once, and they can be applied to various products, and the integrity of the components only affects the parts themselves.
  - Software procedures were often non-sharable nor reusable. Every programmer does his/her own routines. Software faults could impact other programs running in computers.
OOP cont.

Solution:
Make Software behave like hardware

Thus Object-Oriented Programming was coined.
OOP cont.

- Strictly enforce input and output variables (instance variables) as input/output respectively, similar to HW input/output signals
- Strictly limit the SW execution within its domain (methods) with its resource
- All these mean “ACCESS CONTROL”.
OOP Characteristics

- **Abstraction**
  To separate the essential from the nonessential characteristics of an entity, to make reuse easy.

- **Encapsulation**
  - **Security** – To improve reliability by localizing programming errors and mistakes to individual objects

- **Polymorphism**
  - **Sharing** – Multiple service providers can honour the contractual interface, and they can be interchanged dynamically without affecting the clients

- **Dynamic binding**
  - **Sharing** – To link methods (procedures) at run-time

- **Inheritance**
  - **Sharing** – To extend data types and functionality to a software system
OOP Characteristics cont.

An object (Class and instance) comprises of:

- **Instance variables/fields** ("data" in non-OOP)
- **Methods** ("procedure" in non-OOP)

OOP access control:

Objects communications and data exchanging via Methods
OOP meets Today’s Computing Needs

- Software modules can be shared, and re-used
  - Productivity increases, quality improved

- Portability to multiple platforms
  - Networking/Internet, heterogeneous, productivity

- A module cannot affect other modules if faulty
  - High security, high quality, behaving like hardware
OOP vs Procedural-Oriented Programming

Object-Oriented Programming

- Colour
- Wheels
- Doors
- Speed
+ Run()
  + Accelerate()
  + Stop()

MyCar
YourCar

Traditional Data/Procedure Oriented Programming

Data input
Processing
Data Output
OOP in Hardware

OOP needs access control hardware

Von Neumann Processor

Original von Neumann programming model

Von Neumann Processor with OOP system attribute support

Object-oriented programming model

© 2011 Copyrights, all rights reserved by A. Fong
JAVA

- JAVA is a successful OOP language, popular in Internet, mobile, and other applications.
- Inherit OOP advantages for security/protection, sharing, and re-usability.
- High Portability and Platform Independent
- Automatic Memory Management
JAVA cont.

- JAVA objects protect their own properties via a techniques called encapsulation
- Each property defines its own access right level by Access Modifiers
Access Modifiers in JAVA

- **Public**: A class, field variable or method is accessible from any class.
- **Protected**: A field variable or method is accessible to any class that is part of the same package as the class in which the variable is declared.
- **Private**: A field variable or method is only accessible in the class in which it is declared.
- **Default (package)**: A field variable or method is accessible in any class that is part of the same package as the class which the variable or method is declared.
Agenda

- Object-Oriented Programming (OOP)
- **Problems of today’s computers with OOP**
- HISC – Operand Descriptor machine
- jHISC – Architecture, Design, & System Software
- Performance Evaluation and Functionality Comparison
- Project status
- Future development
Overheads of OOP

- Extra access control checking of objects communications
- Extra data for storing object information
- Extra load for resolving objects
- A complex memory management scheme is required for dynamic memory allocation
Existing Hardware Support for OOP

- Traditional computing system favors to procedure-oriented programming
- Extra software overheads to handle features of OOP
- Memory access control of existing system is memory page based, cannot utilize OOP security features
Solution

- To perform OOP tasks as much as possible by hardware
  - Hardware is relatively cheap, and increases fast in #gates per unit die area, and #gates, and decreases in cost per gate
  - The silicon gate productivity in today’s computers is notoriously poor
  - Apply the hardware technology to solve OOP performance problems
Agenda

- Object-Oriented Programming (OOP)
- Problems of today’s computers with OOP
- **HISC – Operand Descriptor machine**
- **jHISC – Architecture, Design, & System Software**
- Performance Evaluation and Functionality Comparison
- Project status
- Future development
Past attempts for OO Processors

- Intel i432
- IBM System 38
- Univac
- Burroughs
- Data General FHP

Only System 38 and i432 went to production, and only system 38 did ok in market.
i432 development team and Dave Rogers of DEC (PDP-11/70 and Vax-11/780) formed Sequent Machine Inc.
High-level Instruction Set Computer

- An OOP computer architecture
- Extended from descriptor computers to provide OOP support
- Operand Descriptor tables are used to carry out the object concept
HISC Operand Descriptor

- Operand descriptor works as TAG to operand data in memory, thus as tag directly to programming variable.
- The tag consists of system attributes, for individual programming variable.
- The tag is specified by compiler, supported by operating system, and can be dynamically updated.
HISC Operand Descriptor cont.

- System attributes include:
  - Access control
  - Caching options
  - System support functions
  - Hardware acceleration features such as vector and variable data length
Procedure-oriented Programming Model by von Neumann processors
Object, security, sharing execution by von Neumann processors

Data describe other data which need to be interpreted by programs

multiple layers of -->

© 2011 Copyrights, all rights reserved by A. Fong
OOP Execution with Hardware Support
(HISC as processor for von Neumann OOP)

- Opcode
- RI

Instruction

Operand Descriptor
Table in Caches

Memory

Data

Program

Hardware support
for attributes for data in the operands
Operand Descriptor Format

<table>
<thead>
<tr>
<th>address</th>
<th>access</th>
<th>caching</th>
<th>timeliness</th>
<th>addr. mode</th>
<th>extensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

1. Access control for object computing
2. Caching options for multiprocessing
3. Timeliness for network/Internet computing
Four U.S. patents issued

- U.S. Patent #6,292,879, 18 Sept 2001
  Method and Apparatus of Access Control List and Cache Enabling and Cache Coherency Requirement Enabling on Individual Operands of an Instruction of a Computer

- U.S. Patent #6,941,444, 6 Sept 2005
  Specifying Branch Prediction for Operands

- U.S. Patent #7,146,479, 5 Dec 2006
  Method and Apparatus of Storage Allocation/De-allocation in Object-Oriented Programming Environment

- U.S. Patent #7,487,507, 3 Feb 2009
  Secure Control Transfer in an Information System
One U.S. patent pending

- Specifying Data Timeliness requirement and Trap Enabling on Instruction Operands of a Processor".
Agenda

- Object-Oriented Programming (OOP)
- Problems of today’s computers with OOP
- HISC – Operand Descriptor machine
- jHISC – Architecture, Design, & System Software
- Performance Evaluation and Functionality Comparison
- Project status
- Future development
JAVA Weak Points

- The unsatisfied performance of JAVA is due to its extra overheads comparing with traditional programming

- Overheads are mainly due to:
  - Automatic Memory Management such as GC
  - JAVA Virtual Machine
  - Runtime bytecode conversion
  - Access Control Checking
JAVA Weak Points cont.

- As existing hardware cannot utilize the security features provided by OOP and JAVA, access control is handled by software
- No instruction to support object-oriented related operations, which are the most time-costly operations in OOP
jHISC

- A version of HISC optimized for Java
- 32-bit processor
- RISC-like core
- HISC operand descriptor with access control for OOP instructions
- Hardware readable data structure, Operand Descriptor Table, to descript object
jHISC Instruction Set

RISCInstructionSet

CPU Load, Store, and Memory Control Instructions
CPU Trap Instructions
FPU Load, Store, and Memory Control Instructions

jHISCInstructionSet

CPU Arithmetic Instructions
CPU Branch and Jump Instructions
CPU Instruction Control Instructions
CPU Logical Instructions
CPU Move Instructions
CPU Shift Instructions
FPU Arithmetic Instructions
FPU Branch Instructions
FPU Compare Instructions
FPU Convert Instructions
Privileged Instructions

CPUObjectData Access Instructions
FPUObjectData Access Instructions
Object Communication Instructions
Object Manipulation Instructions

© 2011 Copyrights, all rights reserved by A. Fong
### jHISC Instruction Format

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE</td>
<td>31-24</td>
</tr>
<tr>
<td>IMM10</td>
<td>23-14</td>
</tr>
<tr>
<td>IMM16</td>
<td>22-14</td>
</tr>
<tr>
<td>R0</td>
<td>8-7</td>
</tr>
<tr>
<td>R1</td>
<td>8-7</td>
</tr>
<tr>
<td>R2</td>
<td>8-7</td>
</tr>
</tbody>
</table>

© 2011 Copyrights, all rights reserved by A. Fong
jHISC Operand Descriptor Tables
Object Representation

Object 1 context

Object 1
OODT

Persistent variables

CODT

Class variables

Method 1

Method 2

Object 2 context

Object 2
OODT

Persistent variables
jHISC Java Object Representation

Object ODT
- Bounded Address
- Class Reference: ParentClass
- Runtime Package Identity
- Public int a
- Protected int b
- int y

Class ODT
- Bounded Address
- Class Reference
- Class Identity
- Public Static int c
- Public Static Method_A()
- Public Parent Class()
- Public Method_B()

ParentClass
- Method_A()
- ParentClass()
- Method_B()

Method_B()
- Bounded Address
- Local Var. Stack Size
- Method Code Base Address
- int i
- Private int x
- Method Code for Method_B()

Stack
- 0
- 0
- 0

Memory
- 0
- 0
- 0

Direct addressing
Stack relative addressing
ODCHECK stage

- OD for operand 0 (Register)
  - Verification Unit for OD0
    - Result
  - Decoded information for instruction

- OD for operand 1 (Register)
  - Verification Unit for OD1
    - Result

- OD for operand 2 (Register)
  - Verification Unit for OD2
    - Result

- Runtime Package Identity for current class (Register)
- Runtime Package Identity for target class (Register)

- Same Package?
- Access Modifier of target class (Register)

- Final Decision
jHISC 5 pipelining stages

- Instruction fetching
- Instruction decoding
- Data fetching
- Execution
- Write back
Overall Board Diagram

- IFETCH Stage Controller
- Instruction Cache
- Instruction Queue
- Branch Prediction Unit

- IDECODE Stage Controller

- DFETCH Stage Controller
  - Data Cache
  - Data Buffer
  - Register Stack Engine

- EXEC Stage Controller
  - Arithmetic and Logic Unit
  - Floating Point Unit

- WBACK Stage Controller

- Stage Controllers
- Cache & Registers
- Execution Units

© 2011 Copyrights, all rights reserved by A. Fong
Hardware readable Object structure

- Hardware readable data structure, Operand Descriptor Table, to implement object
- The Operand Descriptor is designed in a similar structure of JAVA described in Sun Microsystems JAVA specification
- The overhead to convert high-level object into machine code can be minimized
Object-oriented instructions to manipulate OOP

- In JAVA, there is over 40% operations are the memory load/store operations and not more than 30% are object-oriented related operations
- The load/store operations use around 1 CPU cycle while object-oriented related operations use more than 100
Object-oriented instructions to manipulate OOP cont’

Average Bytecode Distribution

<table>
<thead>
<tr>
<th>Operation</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constants Operation</td>
<td>5.0%</td>
</tr>
<tr>
<td>Load Operation</td>
<td>45.0%</td>
</tr>
<tr>
<td>Store Operation</td>
<td>15.0%</td>
</tr>
<tr>
<td>Stack Operation</td>
<td>10.0%</td>
</tr>
<tr>
<td>Arithmetic and Logic Operation</td>
<td>20.0%</td>
</tr>
<tr>
<td>Type Conversion Operation</td>
<td>25.0%</td>
</tr>
<tr>
<td>Control Flow Operation</td>
<td>30.0%</td>
</tr>
<tr>
<td>Method Handling Operation</td>
<td>35.0%</td>
</tr>
<tr>
<td>Objects and Arrays Operation</td>
<td>40.0%</td>
</tr>
<tr>
<td>Other Operation</td>
<td>45.0%</td>
</tr>
</tbody>
</table>
Object-oriented instructions to manipulate OOP cont.

- jHISC provides selected OOP manipulation instructions to speed up OOP processing
- Currently, these OOP manipulation instructions cover over 66% of OO related operations in JAVA
Object Boundary Registers

- Each object associates with a pair of memory boundaries (upper and lower boundaries)
- Ensure security of different data structures in the system and avoid out-of-boundary access of data structures
Object Boundary Registers cont.

- Upper Reg.
- Lower Reg.
- Object A
- Object B
  - Out of Boundary
  - Illegal Access
Inhibiting Illegal Memory Access

- Provide access control checking during object invocations/revocations
- Inhibit the use of pointers to eliminate illegal accesses between modules.
- Inhibit code bypassing with the rigid rules defined in invoke and revoke instructions.
Inhibiting Illegal Memory Access cont.

Traditional Computing System

- Intrusion program
- Security checking
- Protected Area

bypass security checking by jump/branch instructions
Inhibiting Illegal Memory Access cont.

jHISC System

Protected Area

Only valid entry point

A method

Security checking

Intrusion program

Not allowed!!
Embedded Object Access Modifier

- Embedded flags to identify the public, private, protected and default modifier in operand descriptor
- Public and private access modifiers are checked by hardware during property accessing
- Others are checked by software
~90% of access modifiers belong to public and private, so that about 90% of access checking overheads are reduced by hardware assistance.
Hardware Concurrent Garbage Collector

- In JAVA, Garbage Collector is provided for memory allocation
- Garbage Collection is activated whenever memory is not sufficient for memory allocation
- The whole execution will pause (stop-the-world) during the processing
Hardware Concurrent Garbage Collector cont.

- More than 90% of objects are small in size and short in life
- A Hardware Concurrent Garbage Collector for cache is provided for Garbage Collection in jHISC
Hardware Concurrent Garbage Collector cont.

- Activated during idle states of the processor
- The Garbage Collection process is speeded up and its performance impact to the normal executions is minimized
Runtime Hardware-Assist Bytecode Translations

- A runtime hardware Bytecode-to-NativeCode Translator is provided.

- Bytecodes can be translated into jHISC native codes from 1-to-1 to N-to-1 by a simple hardware logic circuit.

- The overhead of the translation is reduced without losing the portability of JAVA.
Runtime Hardware-Assist Bytecode Translations cont.

1. **Java Class File**
2. **Java Bytecode**
3. **Hardware Translator**
4. **jHISC Instruction**
5. **jHISC Execution Core**
Agenda

- Object-Oriented Programming (OOP)
- Problems of today’s computers with OOP
- HISC – Operand Descriptor machine
- jHISC – Architecture, Design, & System Software
- **Performance Evaluation and Functionality Comparison**
- Project status
- Future development
Performance Evaluation

- A performance analysis is conducted to demonstrate the expected performance gain

Methodology

- Java benchmark program
- HISC emulator vs. UltraSPARC processor
- results scaled to same clock rate (100MHz) and normalized to slowest configuration
Performance Evaluation cont.

Systems Under Test

- UltraSPARC System
  - 270MHz UltraSPARC IIi processor, 256KB external cache, 128MB 60 ns RAM

- HISC System
  - Assumed 5 stage pipeline:

```
Stage 1: Instruction Fetch
Stage 2: Instruction Decode
Stage 3: DataFetch
Stage 4: Execute
Stage 5: WriteBack
```
Benchmark Program in Java

class TestAtom
{
    protected long invokeCount;
    public long Execute(int iterations)
    {
        int i;
        invokeCount = 0;
        for(i = 0; i < iterations; i++)
        {
            SeriesA(i);
            SeriesB(i);
        }
        return invokeCount;
    }
    public int SeriesA(int a)
    {
        invokeCount++;
        if(a > 0)
            return a + SeriesA(a - 1);
    }
    public int SeriesB(int b)
    {
        
        invokeCount++;
        if(b >= 0)
        {
            if((b & 1) != 0)
                b = b + SeriesB(b - 1);
            else
                b = 1 + b + SeriesB(b - 1);
            return b;
        }
        else
            return 0;
    }
}

Performance Evaluation cont.
## v.1 Spec Simulation Results

### Execution Times

<table>
<thead>
<tr>
<th>Method</th>
<th>System</th>
<th>Execution Time (ms)</th>
<th>Scaled Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Native</td>
<td>HISC</td>
<td>671</td>
<td>671</td>
</tr>
<tr>
<td>JIT</td>
<td>Ultra-5</td>
<td>3362</td>
<td>9077</td>
</tr>
</tbody>
</table>

### Speedup Factor – better than 10-to-1

<table>
<thead>
<tr>
<th>Method</th>
<th>System</th>
<th>Speedup Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Native</td>
<td>HISC</td>
<td>13</td>
</tr>
<tr>
<td>JIT</td>
<td>Ultra-5</td>
<td>1</td>
</tr>
</tbody>
</table>
Other Java processors

- picoJava2 from Sun Microsystems
- XPRESSOcore from Zucotto
- aJ-100 from aJile
- JA108 from Nazomi
- Moon2 from Vulcan Machines
- Lightfoot from DCT
- Espresso from Aurora VLSI
- JVXtreme Accelerator from Synopsys
## Functionality Comparison

<table>
<thead>
<tr>
<th></th>
<th>jHISC</th>
<th>others</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operand descriptor</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>protection</td>
<td>Individual operands</td>
<td>most on pages or no paging</td>
</tr>
<tr>
<td>HW access control</td>
<td>~90% by HW</td>
<td>partial hardware</td>
</tr>
<tr>
<td>HW-assist OOP instructions</td>
<td>66%+</td>
<td>Microprogramming or SW traps</td>
</tr>
<tr>
<td>HW garbage collection</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>
Agenda

- Object-Oriented Programming (OOP)
- Problems of today’s computers with OOP
- HISC – Operand Descriptor machine
- jHISC – Architecture, Design, & System Software
- Performance Evaluation and Functionality Comparison
- **Project status**
- Future development
Implementation Methodology

- jHISC core is being implemented on Nallatech FPGA board with Xilinx Virtex XCV 800-6
- Provides software library for PCI communication
- Extendable Nallatech DIME modules with XCV 2000
jHISC Project Status

- Behavioral VHDL functional simulation has been completed
- A Nallatech Ballynuey2 card with four Xilinx Virtex XCV 2000E-6 FPGAs and PCI interface to PC for a HISC model is in progress
- Basic input/output testing-vector and integer instructions can be executed on the FPGA board
- Object-oriented instructions are being evaluated
jHISC Project Status cont.

- The performance of the core is being optimized
- Native code compiler is being developed
- Benchmark executing platform is under development
- The structure of the basic kernel is being defined
ACM Special Interest Group (SIG) Computer Architecture News publications


Student Competition Prices

Winner of Second Class Price in the Challenger’s Cup in Xi’an, China, Sept 2001, by MOK Pak Lun, titled:
新世纪计算机体系结构-支持面向物件的微处理器设计与实现

Winner of Second Class Price in the 2005 Challenger’s Cup in Shanghai, China, by Yau Chi Hang and Tan Yiyu, titled:
一种带实时编译器的新型Java处理器
(A Java Processor with Real-time Instruction Folding)

Winner of Second Class Price in the 2009 Challenger’s Cup in Beijing, China, by Yau Chi Hang, titled:
An Efficient Java Processor with Hardware-Software System Co-Design
Winners of 2008 and 2004 CityU EE Final-Year-Project Competition Championships

- **May 2008** – WONG Pak Shing
  An Effective Model of Cache Coherence Protocol with VHDL Simulation

- **May 2004** – YAU Chi Hang
  A Portable Compiler with Efficient Class Loading and Memory Optimization for Mobile Embedded Object-Oriented Computing
Agenda

- Object-Oriented Programming (OOP)
- Problems of today’s computers with OOP
- HISC – Operand Descriptor machine
- jHISC – Architecture, Design, & System Software
- Performance Evaluation and Functionality Comparison
- Project status
- Future development
Future R&D

- Support the J2ME CLDC and MIDP profiles (libraries)
- Provide real-time application support
- Enhance the instructions to support more OOP such as .NET frameworks
- Extend the jHISC architecture to support enterprise server processor such as J2EE
Development Plan

- Start with small-scale microprocessor with TCP/IP capacity for Internet
- Low-cost custom chips for special applications: smart-phones, smart-card systems, testers, financial system front-end, medical equipment, etc.
- Coupled with Linux as operating system
- Internet/Cloud computing
- Timeliness options and other OD extensions
Cloud Computing

- Computing resources – processing, storage, messaging, database are from external data centers thru networking, and pay only what one uses.
- Like other utilities such as telephone, electrical power, and VISA credit cards: world-wide functional.
  - Like electrical power. Away from individuals’ own power generators to get power from utility energy companies.
- The actual channeling of work and resource assignments are hidden from the users.
- Users generally do not affect others.
Cloud Computing cont.

- Cloud Computing is to get computing resources, both hardware and software, from a provider, so that costs and management will be shared and minimized.

- It is like power companies.

  In the early days, all factories had their own power generators. Today they get power from power companies. All power generator issues are hidden from them. The clients just pay what they use.

  With Cloud, all IT infrastructure will be shared and hidden from the users.

- Distributed computing and object computing are the keys for Cloud.
Conclusion

- Superior Security and OOP support
- OO performance advantage
- Better memory allocation and memory garbage collection with hardware support
- Translation from Java bytecode to HISC codes is more efficient and direct
The New IT Engine – HISC

- Computer is the engine for IT
- Powerful computing systems are built on individual processors
- Four U.S. patents granted, and one pending
- Proven advantages on object computing
- The operand descriptor can be extended for future needs
jHISC
Object-Oriented Computer for the 21st Century

mobile
small core size, small memory footprint

desktop
multimedia, high performance

MARKETS

server
data coherence, secure environment

© 2011 Copyrights, all rights reserved by A. Fong
HISC Development Group
Thank you!

Q & A

HISC Website: http://www.ee.cityu.edu.hk/~hisc/
e-mail: anthony.fong@cityu.edu.hk