Innovative System and Application Curriculum on Multicore Systems

Pangfen Liu ¹, Greg C. Lee ², Jenq Kuen Lee ³, and Cheng-Yen Lin ³

¹ National Taiwan University, Taipei, Taiwan.
² National Taiwan Normal University, Taipei, Taiwan.
³ National Tsing Hua University, Hsinchu, Taiwan.

MOE Embedded Software Consortium, Taiwan.
Outline

• Taiwan ESW Consortium
• Motivation
• Lab modules with Multicore systems
• Discussions
• Conclusion
Taiwan Embedded Software Consortium

• The Embedded Software Consortium, established in February 2004 in Taiwan, is the consortium funded by the Ministry of Education.

• The ESW Consortium focuses the development of embedded software curriculum.

• We hope to provide a reference curriculum for universities in Taiwan to develop their embedded program.

• Currently conjunction with National Communications Program (NCP)
ESW Consortium

- Hands-on Lab modules
- Focus:
  - Basic/Advance embedded curriculum
  - Domestic embedded processor platforms
  - Open platforms (Android)
  - Multicore/Embedded Multicore platforms
- ESW promotion
- Embedded System Hardware/Software Design Contest

Diagram:
- MOE Advisory Office
- NCP Consortium
- Advisory Committee
- ESW consortium
- Other consortiums
- ES Design contest
- Partner Universities
- Collaboration With TEIA
- Collaboration with NSC NCP Program
- Collaboration with International Companies
Motivation

• The multicore architecture are increasingly important in system design
  – Amend traditional content of system education for multicore software development
• Shared ground between embedded multicore systems and high performance parallel systems
• Help students to learn the parallel design patterns of parallel programming to lay the foundations for advanced multicore system research.
• Government Support and Intel Collaboration
Multicore/Embedded Multicore Trends

- High bandwidth interconnect with multi-channel memory support
- MPU 0, MPU n
- Memory Controller: ExMem, ExMem
- Display, Mem
- sDMA
- Hardware Accelerator
- Image Signal Processor (SIMD processor)
- RAM, SP
- 2D/3D Graphic Accelerator
- RAM, RAM
- DSPs, Accelerators
- CryptoDMA, Security Accelerators
- eDMA, MMU
- Baseband Processor
- Wireless, GPS, Bluetooth, ...
- Multi-core
- Trend: From processors to multi-PE (SDR)

External chips

Baseband Processor

Trend: From processors to multi-PE (SDR)
### Course Introduction

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Introduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>ES-Y11-A1</td>
<td>Embedded application/software studio</td>
<td>Introduction to Android programming</td>
</tr>
<tr>
<td>ES-Y11-A2</td>
<td>Introduction to marketing place</td>
<td>Developing and business trend of embedded application will be introduced</td>
</tr>
<tr>
<td>ES-Y12-A1</td>
<td>Augmented reality applications for embedded systems</td>
<td>Advanced embedded software for augmented reality application and lab modules based on embedded hardware with sensors support</td>
</tr>
<tr>
<td>ES-Y12-A2</td>
<td>Android application design</td>
<td>Mobile application design and implementation in the Android environment</td>
</tr>
<tr>
<td>ES-Y13-A1</td>
<td>Mobile voice based application design</td>
<td>The introduction of voice recognition and UI control techniques based on mobile devices</td>
</tr>
<tr>
<td>ES-Y13-A2</td>
<td>Mobile embedded software design</td>
<td>Mobile applications design and lab modules on based domestic embedded platforms</td>
</tr>
</tbody>
</table>
# Course Introduction

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Introduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>ES-Y11-M1</td>
<td>Parallel programming in multicore systems</td>
<td>Principles and practice for parallel programming. Software studio are evaluated on Intel MTL.</td>
</tr>
<tr>
<td>ES-Y11-M2</td>
<td>Introduction to real world application in MTL environment</td>
<td>Wide range of real world multicore applications will be introduced.</td>
</tr>
<tr>
<td>ES-Y12-M1</td>
<td>Mobile + Cloud applications</td>
<td>The design and implementation of mobile cloud applications.</td>
</tr>
<tr>
<td>ES-Y12-M2</td>
<td>Mobile + Cloud programming and system software</td>
<td>Understand the system software and programming tools for mobile cloud applications.</td>
</tr>
<tr>
<td>ES-Y13-M1</td>
<td>Virtualization on embedded multicore systems</td>
<td>The advanced virtualization techniques for embedded system will be introduced.</td>
</tr>
<tr>
<td>ES-Y13-M2</td>
<td>Multicore programming and power optimization</td>
<td>The power optimization techniques for multicore platform.</td>
</tr>
</tbody>
</table>
## Course Introduction

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Introduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>ES-Y11-S1</td>
<td>Embedded compilers</td>
<td>Compiler techniques and system software development flow for embedded systems will be introduced</td>
</tr>
<tr>
<td>ES-Y11-S2</td>
<td>Innovative multimedia labs</td>
<td>The design and implementation of portable multimedia applications</td>
</tr>
<tr>
<td>ES-Y12-S1</td>
<td>Embedded multicore programming</td>
<td>Introduction to embedded multicore platforms and programming techniques</td>
</tr>
<tr>
<td>ES-Y12-S2</td>
<td>Embedded operating systems</td>
<td>Introduction to embedded operating systems</td>
</tr>
<tr>
<td>ES-Y13-S1</td>
<td>Dynamic compilers</td>
<td>Advanced dynamic compilation techniques will be introduced</td>
</tr>
<tr>
<td>ES-Y13-S2</td>
<td>Innovative Android system optimizations</td>
<td>Mobile system optimization techniques based on Android will be introduced</td>
</tr>
</tbody>
</table>
This semester

• Virtualization techniques on embedded systems
  – Process virtualization: language level, OS level, Cross-ISA
  – Device virtualization
  – System virtualization
  – Facilities
    • Prof. Wei-Chung Hsu (NCTU), Prof. Chi-Sheng Shih (NTU),... etc.
  – Labs
    • QEMU/LnQ
    • JVM/CVM/KVM
    • Xen/Opennebula
    • OpenStack/Ubuntu UEC
    • GPU+VMGL
    • Hadoop+HDFS

Outline
- Introduction to virtual machines
- Fast emulation: dynamic binary translation
- Full virtualization and paravirtualization
- Architecture supports for virtualization
- KVM-based virtualization for ARM-based embedded systems
- Optimizations
- Hypervisor for embedded real-time systems
- Emerging applications of embedded system virtual machines
This semester

• Embedded multicore programming
  – Embedded multicore architectures
  – GPU/GP-GPU
  – OpenCL
  – Facilities
    • Prof. Ching-Hsien Hsu(CHU), Prof. Kuan-Ching Li(PU), Prof.Yuan-Shin Hwang(NTUST) ...etc.
    – Labs
      • OpenCL programming on X86/ATI/NVIDIA

Outline

➢ Introduction to parallel programming
➢ GPU Architecture
➢ Introduction to OpenCL
➢ The programming model of OpenCL
➢ Synchronization
➢ Debugging techniques
➢ Case study for optimizations
This semester

- **Android programming**
  - Understand the Android system
  - Mobile applications
  - Sensor applications
- **Facilities**
  - Prof. Gwan-Hwan Hwang(NTNU), Prof. Shih-Hao Hung(NTU), Prof. Shang-Hung Wu Hwang(NTHU) ...etc.
- **Labs**
  - Android app widget
  - GPS applications
  - Android graphic applications

Outline

- Introduction to Android system
- Understand the development of Android app
- The app widget
- The Android UI design
- GPS/Sensor Applications
- Optimization techniques
- Advanced graphic applications
- Cloud and Devices
This semester

• **Embedded system applications**
  – The development of embedded applications
  – Domestic embedded platforms
  – Network & multimedia applications
  – **Facilities**
    • Prof. Jyh-Cheng Chen (NCTU), Prof. Jing Chen (NCKU), Prof. Jyh-Shing Jang (NTHU) ... etc.
  – **Labs**
    • Wireless applications
    • Speech and Audio applications

Outline

- Embedded software design and practice
  - Cross compilation
  - Embedded OS
  - Device driver
- Wireless applications
  - 802.1X
  - Socket programming
  - Security standards
- Speech and Audio processing
  - Audio signal processing
  - Speech features
  - Recognition techniques
- PAC/PAC-DUO Introduction
  - Toolchain
  - Media codecs
  - DVFS
Intel MTL/MOE Curriculum Program

• Intel collaborates with Taiwan MOE
• Intel provides the resources, trainings, tools and community
• Course materials design by Taiwan professors
• Two Lab Modules:
  – Lab Modules for Parallel Programming in Multicore Systems
  – Lab Modules of Parallel Programming on Real-World Applications
• 14 Faculties: Pangfeng Liu (NTU), Jen-Wei Hsieh (NTUST), Rong-Guey Chang (CCU), Chao-Tung Yang (THU), Chih-Ping Chu (NCKU), Wuu Yang (NCTU), Greg Lee (NTNU), Tsung-Che Chiang (NTNU), Che-Rung Lee (NTHU), Li-Chun Wang (NCTU), Charles Wen (NCTU), Ren-Guey Lee (NTUT), Chung-Chih Lin (CGU), Che-Lun Hung (PU)
Intel MTL Environment

- License Server
- File Server
- Batch Node
- Batch Node
- Batch Node
- Batch Node

High Speed Network

Login Node

Software Environment and Tools
- Linux
- C/C++ compilers
- Performance Analyzers
- Thread builder
- Runtime library support

Four Intel® Xeon® Processor (E7-4860) - Providing 40 cores
Multicore Curriculum

• Software Tool Lab Modules for Parallel Programming in Multicore System
  – Principle of parallel programming
  – Programming models
  – Parallel algorithms
  – Tools for parallel programming

• Lab Modules for Real world Application
  – Real world application
    • Infection simulations, logic simulation, healthcare applications, monitoring applications with air quality…
  – Parallel design patterns
Principle of Parallel Programming

• Multicore Systems
  – Shared/Distributed memory
• Level of Parallelism
  – Data parallelism
  – Task parallelism
• Parallel Programming Models
  – Pthread
  – OpenMP
  – MPI
  – CUDA
  – MapReduce
• Amdahl's law
Amdahl's Law

\[
\text{Speedup} = \frac{T_1(n)}{T_p(n)} = \frac{T_{seq}(n)}{T_{seq}(n)} + \frac{T_{parallel}(n)}{T_{parallel}(n)} \Rightarrow \\
\text{Assume } p \rightarrow \text{Infinite} \\
\text{Speedup} = 1/r \\
\text{r} = \frac{T_{seq}}{T} = \frac{T_{seq}(n)/T + T_{parallel}(n)/(T*p)}{1} \\
\text{r} + \frac{(1-r)/p}{1}
\]
Sandia’s Version of Amdahl’s Law

\[
\text{Speedup (n)} = \frac{T(n)}{1 + \frac{T_p(n)}{T_{seq}(n)} + \frac{T_{parallel}(n)}{n}}
\]

As \( n \to \infty \), \( \text{Speedup(n)} \to p \)
Design Patterns

• Brief introduction to design patterns
  – Proposed by C. Alexander for city planning and architecture.
  – Introduced to software engineering by Beck and Cunningham.
  – Become prominent in object-oriented programming by GoF.

• Design patterns describe “good solutions” to recurring problems in a particular context.
  – Patterns for object-oriented programming
    • Creational patterns, Structural patterns, Behavioral patterns, etc.
  – Patterns for limited memory systems
    • Compression, Small data structures, Memory allocation, etc.
  – Patterns for parallel programming
    • Finding concurrency, Algorithm structure, Supporting structures and Implementation mechanisms.
Parallel Design Patterns

Parallelization can be a process to transform problems to programs by selecting appropriate patterns.

- **Finding Concurrency**
  - Decomposition of problems
  - Dependency analysis patterns: {group tasks, order tasks, data sharing}
  - Design evaluation pattern

- **Algorithm Structure**
  - How the given problem is organized?
    - By tasks: {task parallelism, divide & conquer}
    - By data decomposition: {geometric, recursive}
    - By flow of data: {pipeline, event-based coord.}

- **Supporting Structures**
  - Software constructs to express parallel algorithms
    - Program structures: {SPMD, master/worker, loop parallelism, fork-join, client-server, SIMD}
    - Data structures: {shared data, shared queue, distributed array}
    - UE management: {thread/process creation/destr.}
    - Synchronization: {barrier, mutex, mem fence}
    - Communication: {msg passing, collective comm}

- **Implementation Mechanisms**
  - Design Space of Parallel Patterns
  - Decomposition patterns: {data, task}
  - Dependency analysis patterns: {group tasks, order tasks, data sharing}
  - Design evaluation pattern

These patterns are summarized from the book, “Patterns for Parallel Programming” by Mattson et al.
Lab Module: Principles of Parallel Programming

- Basic of Parallel Programming
  - Programming Models
    - Open MP
  - Runtime library
  - Intel TBB
  - Fork-Join Parallelism

- Generic Parallel Algorithms
  - parallel_for, parallel_for_each*
  - parallel_reduce
  - parallel_scan
  - parallel_do
  - pipeline
  - parallel_sort
  - parallel_invoke*

- Synchronization Primitives
  - atomic, mutex, recursive_mutex
  - spin_mutex, spin_rw_mutex
  - queuing_mutex, queuing_rw_mutex
  - null_mutex*, null_rw_mutex*

- Generic Parallel Algorithms
  - parallel_for, parallel_for_each*
  - parallel_reduce
  - parallel_scan
  - parallel_do
  - pipeline
  - parallel_sort
  - parallel_invoke*

- Concurrent Containers
  - concurrent_hash_map
  - concurrent_queue
  - concurrent_vector

- Task scheduler
  - task
  - task_scheduler_init
  - task_scheduler_observer

- Memory Allocation
  - tbb_allocator
  - cache_aligned_allocator
  - scalable_allocator

TBB Components

Threads
tbb_thread
Lab Module: Multicore Programming with Thread Profiler

- Performance bottleneck analysis with software tools
  - Critical section
  - Load balance
  - Waiting External event
  - Synchronization overhead
- Sequential -> Parallel

Project: Try the parallel flash memory simulator FAST

Block: The basic unit for erase operation
Page: The basic unit for read/write operation

Block 0
Block 1
Block 2
... Block 8191

2K Bytes
64 Bytes

Embedded Software Consortium

Workshop on Embedded Systems Education, 2011
Lab Module: Air Quality Monitoring System over MTL platform

- Air Quality Monitoring with Sensors
  - CO, CO₂, O₃
- Data Analyzing on Cloud System
Lab Module: Development of short-term memory assessment tools and for healthcare applications

• Dementia Diagnosing System
  – PDA/hand held devices
  – Memory assessment system to help diagnosing dementia disease

• Cloud Database System
  – Global rating analysis for Alzheimer disease
  – Elder degenerative trend analysis
Lab Module: Infection Simulations on Multicore Processors

- Mote Carlo Simulation
  - Predict the Infection trend
    - Build up the analytical model
  - Sequential -> Parallel
    - OpenMP
    - Pthread
    - Intel TBB
  - Amdahl's Law

Infection/Recover

Vaccination

Movement

Statistics

Taiwan Flu Statistics

Simulation Results
Lab Module: Implementation of Parallel Algorithm for Logic Simulation on Switching Network

• HOPE
  – A fault simulator for synchronous sequential circuits
  – Parallel fault simulation techniques

• Performance analysis
  – gprof
  – Identify the bottlenecks

Diagram:

- Fault list $F$
- Fault-free simulation for all pattern $T$
- Get fault $f$ from $F$
- Next pattern?
- Fault simulation for pattern $p$ from $T$
- Mismatch?
- Delete $f$ from $F$
- No more fault → END

Simple fault simulation flow
Lab Module: Parallel Computing for Wireless Simulation Platforms

• Apply the concept of parallel computing to expedite comprehensive wireless system simulations.

• Dynamically changing environments
  – Radio channel characteristics
  – Terminal mobility

• Complex scenarios—Many “multiples”
  – Multiple paths
  – Multiple base stations
  – Multiple users
  – Multiple antenna
  – Multiple networks
Lab Module: Parallelizing Memetic Algorithm Using OpenMP

- **Metaheuristics**
  - **Characteristics**
    - Category of approximation algorithms for optimization problems.
    - Tool for solving hard optimization problems
      - Scheduling, routing, clustering
    - Iterative processes
    - Memetic Algorithm
    - Genetic Algorithm
  - **Components**
    - Solution encoding/decoding schemes
    - Neighborhood functions
    - Selection/acceptance criteria
    - Stopping criteria

![Flowchart diagram of the Memetic Algorithm process](chart.png)
Online Judge System

- ACM online judge style system
- Check and grade programming homework
- A platform for in-class worksheets and midterm evaluation
- Parallel processing
Making Progress

- Understand Students’ progress by their submission results of worksheets

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td># of students solved the problem</td>
<td>47/75</td>
<td>72/75</td>
<td>75/75</td>
</tr>
<tr>
<td>Average # of trials till success</td>
<td>2.2</td>
<td>2.8</td>
<td>1.7</td>
</tr>
</tbody>
</table>

From the course, *principle of parallel programming*
Midterm Scoring

Floating Number Sorting

- Number of submission: 330
- Number of people: 75
- Average number of submissions: 4.4

Time(s)

<table>
<thead>
<tr>
<th>min</th>
<th>max</th>
<th>average</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.657065</td>
<td>4.767353</td>
<td>1.733752</td>
</tr>
</tbody>
</table>

Chromatic Coloring Counting

- Number of submission: 110
- Number of people: 75
- Average number of submissions: 1.47

Time(s)

<table>
<thead>
<tr>
<th>Min</th>
<th>max</th>
<th>average</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.264323</td>
<td>8.894231</td>
<td>1.436465</td>
</tr>
</tbody>
</table>
Term Projects

• Student propose the proposal for their term project
  – Application
  – Programming Model

• Selected Projects
  – From Neural Networks Implementations to Find out the Advantage of Parallel Programming
  – Automatic Panoramic Image Stitching using Invariant Features
  – Use Cuda to Implement a Path Finding Algorithm
  – Motion Estimation on CUDA
  – Use CUDA to simulate ocean, and render the scene with OpenGL
  – A Social Network Centrality Analyzer
  – Ray-Tracing Parallelization

• Course site
  – https://sites.google.com/site/ntucsiepp2011/home
Conclusion

- We developed a series of system software and application lab modules on multicore system.
- These lab modules involve various topics of multicore programming techniques and real world multicore applications.
- We trained students with capabilities to develop multicore applications.
- The parallel design patterns is introduced to lay the foundations for advanced multicore system research.
- MTL-based lab modules show good promises for further devising innovative system curriculum.