Programming and Performance Evaluation of the Cell Processor

Ryuji Sakai, Seiji Maeda, Christopher Crookes, Mitsuru Shimbayashi, Katsuhisa Yano, Tadashi Nakatani, Hirokuni Yano, Shigehiro Asano, Masaya Kato, Hiroshi Nozue, Tatsunori Kanai, Tomofumi Shimada and Koichi Awazu

Toshiba Corporation
Outline

- Cell Processor
- SPE Scheduling
  - Programming Model
  - Real-time Resource Scheduler
  - Memory Bandwidth Reservation
- SPE Programming (H.264 encoder as an example)
  - SPE Programming Steps
  - Parallel Execution Model for Sub Modules
  - Evaluation of the H.264 Encoder
- Conclusions
**Cell Processor**

- **Heterogeneous Multi-core processor**
  - 1 Power Processor Element (PPE)
  - 8 Synergistic Processor Elements (SPEs)
  - 2ch XDR DRAM Memory
Multi-layered Programming Model

**PPE/SPE Module Layer**
- Tuner
- Stream Demux
- Audio Decoder
- Video Decoder
- Video Proc.
- AV Renderer

**SPE Thread Layer**
- SPE 0
- SPE 1
- SPE 2

**SPE Overlay Layer**
- Local Storage
- Function
- Function A
- Function B
- Function C
- Function D

---

Aug. 15, 2005

Copyright © 2005 TOSHIBA CORPORATION, All Rights Reserved.
Multi-layered Programming Model

- **SPE module**
  - Designed for parts of stream processing model
    - ex. viewing digital TV, trans-coding digital contents, …
  - Multi SPE threads can be used to utilize multi SPEs

- **SPE thread**
  - An execution context running on an SPE
  - Context save and restore can be applicable
    - Context includes SPE registers, LS, outstanding DMA requests

- **SPE overlay**
  - Text and/or data on LS can be replaced without intervention of PPE
  - Symbol resolution is supported
Real-time Resource Scheduler

- Extends existing OS running on PPE
  - Existing OS only needs to implement glue codes to manage specific features of Cell Processor
- Schedules “Resources” with 2 sched. types
  - Resources: SPEs and Memory bandwidth
  - Sched. types: Time-shared and Dedicated
- Ensures real-time constraints of all SPE modules
  - Detects whether scheduling is feasible or not before accepting new SPE modules
Issue: Performance Stability

To stabilize performance, processing rate and data rate must be adjusted.

Processor performance increases continuously:
- 4-way SIMD processing requires 4 times wider bandwidth.
- Multi-core processor multiplies access rate by the number of cores.

Cell processor tackles this issue:
- SPE has 128 registers and high speed local storage.
- Bandwidth of 2-ch XDR DRAM is up to 25.6 GB/s.

Leveling memory access to XDR DRAM is a key for performance stability:
- Peak memory access initiated by 8 SPEs exceeds even wide bandwidth accomplished by XDR DRAM.
Ans: Memory Bandwidth Reservation

- Resource Scheduler treats memory bandwidth as a time-shared resource
  - Resources: SPEs and Memory bandwidth
  - Bandwidth is only reserved while SPE thread is running
- Memory bandwidth can be specified as one of scheduling parameters of SPE module
  - Scheduling parameters:
    - The number of SPE threads, Processing ratio of SPEs, Memory bandwidth, Precedence constraints between SPE modules
- Real-time Resource Scheduler ensures that total bandwidth never exceeds max capacity of XDR DRAM
  - Keeping other constraints such as precedence constraints
ex. Trans-coder from MPEG-2 to H.264

- 2 trans-coders are running on a Cell Processor
- Focus on H.264 encoder in following sheets

Trans-coder (MPEG-2 to H.264)

H.264 Encoder: 3 SPE threads
Scheduling Image

2 encoders start at the same time

All threads are scheduled properly

BW shortage declines performance

Performance is stabilized with reserved BW

a) without BW reservation

b) with BW reservation

H.264 enc 2 runs after H.264 enc 1

BW of H.264 enc 1

Interfere

Memory Bandwidth

Thread 1

Thread 2

Thread 3

Thread 1

Thread 2

Thread 3

H.264 enc 1

H.264 enc 1

H.264 enc 2

H.264 enc 2

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr

Thr
Evaluation on Cell Processor

- Performance monitoring features are implemented in Cell Processor
  - Hardware counters and event flags can be monitored and recorded sequentially
- Performance monitoring tool is already available
  - The tool is very flexible with configuration file
- SPE behaviors and utilization of memory bandwidth had been monitored
  - Utilization of memory bandwidth of H.264 encoders had been monitored
  - SPE behaviors are presented later
Utilization of Memory Bandwidth

without BW reservation

H.264 enc 2
H.264 enc 1

H.264 enc 2
H.264 enc 1

with BW reservation

Slower performance causes wider access
Performance of H.264 Encoders

With bandwidth reservation, performance scales in a linear fashion.

Without bandwidth reservation, performance declines by 50%.
Multi-layered Programming Model

PPE/SPE Module Layer

- HDD Reader
- Demux
- MPEG-2 Decoder
- AAC Decoder
- H.264 Encoder
- AC3 Encoder
- Mux
- HDD Writer

SPE Thread Layer

- SPE 0
- SPE 1
- SPE 2
- H.264 Encoder
- SPE Thread

SPE Overlay Layer

- Local Storage
- Function
- XDR
- DRAM
- Function A
- Function B
- Function C
- Function D

SPE Programming
Outline

- Cell Processor
- SPE Scheduling
  - Programming Model
  - Real-time Resource Scheduler
  - Memory Bandwidth Reservation
- SPE Programming
  - SPE Programming Steps
  - Parallel Execution Model for Sub Modules
  - Evaluation of the H.264 Encoder
- Conclusions
SPE Programming Steps

- Partition program code into sub modules
  - Not only for parallelization but also for overlay

- Implement sub modules by using the extended C
  - *No assembler programming needed*
  - Coding in high level language is important for performance tuning step
  - Since sophisticated algorithm easily applied, higher performance than assembler can be achieved

- Assign sub modules to the SPE threads
  - Also define program overlay order
  - Configure parallel execution scheme

- Performance tuning
  - Effectively repeats the cycle of evaluation, refinement and debugging for performance tuning
Tips for Implementing Sub Modules

- **DMA latency hiding**
  - Reorder the program segment execution
  - Double buffer or execution queue techniques are useful

- **Effective use of 128 registers**
  - Use local (auto) variables and inline functions
  - Loop unrolling in compiler optimization realize that local defined array can be allocated to register

- **Reduce the penalty of branch miss**
  - Eliminate conditional branch by using logical operation or select instruction
Parallel Execution Model for Sub Module

Program & Data

Data Parallel

SPE 0
SPE 1
SPE 2

Data Pipelining

SPE 0
SPE 1
SPE 2
Structure of H.264 Encoder

Input Video Data

- Motion Estimation
- Intra Prediction

Reference Frame

- Generate Reference
- Deblocking Filter

Output Encoded Bits

- Transform/Quantization
- Context-based Adaptive Binary Arithmetic Coding
- Inverse Transform/Inverse Quantization

Very fine grain data dependency, 100 – 1000 cycles

Intra Prediction predicts pixel data based on surroundings
Partition the H.264 Encoder Code

Motion Estimation is the hotspot of the H.264 encoder.

Coding module was composed from multiple encoder functions.

ME module was divided into 3 sub modules to give priority to the execution speed.

Input Video Data

Output Encoded Bits

Coding module was composed from multiple encoder functions.

ME module was divided into 3 sub modules to give priority to the execution speed.

Reference Frame

IntraPred

T/Q

IQ/IT

ME1 ME2 ME3 MC

GenRef Filter

Coding module was composed from multiple encoder functions.

Input Video Data

Motion Estimation is the hotspot of the H.264 encoder.

Coding module was composed from multiple encoder functions.

ME module was divided into 3 sub modules to give priority to the execution speed.

Reference Frame

IntraPred

T/Q

IQ/IT

ME1 ME2 ME3 MC

GenRef Filter

Coding module was composed from multiple encoder functions.

Input Video Data

Motion Estimation is the hotspot of the H.264 encoder.

Coding module was composed from multiple encoder functions.

ME module was divided into 3 sub modules to give priority to the execution speed.

Reference Frame

IntraPred

T/Q

IQ/IT

ME1 ME2 ME3 MC

GenRef Filter

Coding module was composed from multiple encoder functions.
Assign Sub Modules to the SPE Threads

Input Video Data

Thread 1

ME1 ME2 ME3

Thread 2

MC

Coding

CABAC

Output Encoded Bits

GenRef

Filter

Reference Frame

Thread 3
Overlay by Frame Partitioning

Video frames are partitioned into processing units for pipelining and program code overlay.

Video frames are partitioned into processing units for pipelining and program code overlay.

Thread 3 waits until the 2nd partition is available, because Thread 3 refers to the top of the next partition.
3 Thread Execution Image

1 Frame divided into 5 partitions
Reference frame dependency
Processing elements for 1 Frame
Evaluation of the H.264 Encoder
SPE Core Cycle Utilization

SPE Core Cycle Ratio [%]

branch miss penalty
DMA channel stall
single commit
dual commit

1 Frame

Thread 1

ME1
ME3
ME2
ME1
ME3
ME1
ME3
ME1
ME3
ME1
ME3
ME1
ME3
ME1
ME3
ME1
ME3
ME1
ME3
ME1
ME3
ME1
ME3
ME1
ME3
Evaluation of the H.264 Encoder Memory Bandwidth by SPE Read

![Diagram showing memory bandwidth and thread interactions.](image)

- Thread 1: ME1 → ME2 → ME3
- Thread 2: DMA read from Thread 1
- DMA read from Thread 2

Memory Bandwidth [GB/s]

- ME1
- ME2
- ME3

Time [ms]

- 1 Frame
- 1 Partition
Conclusions

- Real-time Resource Scheduler with Memory Bandwidth Reservation can level memory access and stabilize the performance of SPE modules.
- H.264 Encoder was successfully implemented on SPE thread and overlay programming environment, and evaluated.
- Effectiveness of SPE scheduling and SPE programming is evaluated on “real” Cell Processor with performance monitoring tool.
- Using this infrastructure, next generation’s multi-stream media applications will be accomplished on the Cell Processor.
- Have fun programming on the Cell!