Low Power Stream Processor for Multimedia Signal Processing

Shao-Yi Chien (簡韶逸)

Assistant Professor
Media IC & System Lab
Graduate Institute of Electronics Engineering
National Taiwan University
2008/05/30
Outline

- Introduction
- Stream processor
- Nowadays GPU
- Low power stream processor core
- Multi-core stream processor SoC
- Conclusion
Outline

- Introduction
- Stream processor
- Nowadays GPU
- Low power stream processor core
- Multi-core stream processor SoC
- Conclusion
Mobile Multimedia Platform

The mobile handset becomes the personal multimedia platform
Evolution on Multimedia Communication Platform

Growth of specification

Diverse applications

- Comm.
- Graphic
- Video
- Display

1G: 2D Graphic
2G: Vector Graphic
3G: OPENGL-ES 1.0
4G: OPENGL-ES 2.0

1G: MPEG-1/2
2G: MPEG-4
3G: H.264
4G: SVC

Q CIF
CIF
D1
HDTV
Ultra HDTV
### Specifications of Current Handset

- **Browser phone**
- **Multimedia enhanced feature phone**
- **Smart phone**

<table>
<thead>
<tr>
<th></th>
<th>Browser Phone</th>
<th>Feature Phone</th>
<th>Smart Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Processor Frequency (MHz)</strong></td>
<td>50--60</td>
<td>100--133</td>
<td>200--</td>
</tr>
<tr>
<td><strong>Memory Bus Width (bits)</strong></td>
<td>16</td>
<td>16--32</td>
<td>32</td>
</tr>
<tr>
<td><strong>Memory Bandwidth (MB/s)</strong></td>
<td>100</td>
<td>120--400</td>
<td>800--</td>
</tr>
<tr>
<td><strong>Power Supply (mAh)</strong></td>
<td>600</td>
<td>800--1200</td>
<td>1200--2000</td>
</tr>
<tr>
<td><strong>Price (USD)</strong></td>
<td>50</td>
<td>100--400</td>
<td>400--</td>
</tr>
</tbody>
</table>
Challenge of the 3Low

- Low power consumption
- Low memory bandwidth
- Low silicon cost

<table>
<thead>
<tr>
<th></th>
<th>Browser Phone</th>
<th>Feature Phone</th>
<th>Smart Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor Frequency (MHz)</td>
<td>50--60</td>
<td>100--133</td>
<td>200--</td>
</tr>
<tr>
<td>Memory Bus Width (bits)</td>
<td>16</td>
<td>16--32</td>
<td>32</td>
</tr>
<tr>
<td>Memory Bandwidth (MB/s)</td>
<td>100</td>
<td>120--400</td>
<td>800--</td>
</tr>
<tr>
<td>Power Supply (mAh)</td>
<td>600</td>
<td>800--1200</td>
<td>1200--2000</td>
</tr>
<tr>
<td>Price (USD)</td>
<td>50</td>
<td>100--400</td>
<td>400--</td>
</tr>
</tbody>
</table>
Challenge of the 3High

- High integration
- High resolution
- High performance

Growth of specification:
- 1G: Graphic
- 2G: Vector Graphic
- 3G: OPENGL-ES 1.0
- 4G: OPENGL-ES 2.0

Applications:
- QCIF
- CIF
- D1
- HDTV
- Ultra HDTV

Diverse applications:
- Video
- Graphic
- Display
Challenge of the 3High

2005 SH-4 90nm 181 Million transistors, 13.5 M-gates
- Is there any better solution?
- Unified architecture for high integration functions
- Scalable architecture for high performance and high resolution
- Stream processor is a possible solution!
Outline

- Introduction
- Stream processor
- Nowadays GPU
- Low power stream processor core
- Multi-core stream processor SoC
- Conclusion
What is a Stream Processor?

- A stream program is a computation organized as streams of records operated on by computation kernels.
- A stream processor is optimized to exploit the locality and concurrency of stream programs.

Source: William J. Dally, Lecture Notes of Stream Processor Architecture, 2002
What is a Stream Processor?

A stream is composed of a lot of stream elements in the same data structure
- Pixel for image processing
- Vertex/fragment for graphics rendering

Each stream element is processed with the same procedure, which is called as kernel

The key of stream processor is to separate the operations of
- Data accessing: stream loading and storing
- Processing: kernel
Example

Typical Stream Processor

Imagine

Snapshot of the Stream Processor at ISSCC2007

Outline

- Introduction
- Stream processor
- Nowadays GPU
- Low power stream processor core
- Multi-core stream processor SoC
- Conclusion
Evolution of GPU

- 1995--1998: Texture mapping and z-buffer
- 1998: Multitexturing
- 1999-2000: Transform and lighting
- 2001: Programmable vertex shader
- 2002--2003: Programmable pixel shader
- 2004--2006: Shader model 3.0 and 64-bit color support
- 2007: Stream computing
Half-Life 2
Half-Life 2
Half-Life 2
Performance Evolution

- Graphic Specific Computing
- Programability
- Stream Processing

- Memory Bandwidth (GB/s)
- Vertex Rate (M vertices/s)
- Pixel Rate (M pixels/s)
- GFLOPS
- Power (mw)
- Area (M Transistors)

- GL1.1
- DX6
- DX7
- GL1.4
- DX8
- GL1.5
- DX9
- GL2.0
- DX10

- 1994
- 2004
- 2007
Contemporary GPUs Evolution

Memory Bandwidth (GB/s)
Vertex Rate (M vertices/s)
Pixel Rate (M pixels/s)
GFLOPS (G flops)
Power (mw)

2004
GL1.4
DX8
DX7
DX6

1994

ASIC

10000
1000
100

100000
0.1

ASIC mixed With Programmability

Stream computing Is coming (NV9 ATI-R700)

Area (M Transistors)

Stream Processing

Programmability

Graphic Specific Computing
Before Year 2000, ASIC
After Year 2004, Programmable Shader

http://www.nvidia.com (gh07)
Graphics: Snapshot of NV8600

Massive threads parallel stream processors

Taking Advantage of GPU

Using *texture* to make use of the high bandwidth between GPU and video memory.

System Memory -> North Bridge (6.4 GB/s or more) -> South Bridge -> Other Peripherals

North Bridge (6.4 GB/s) <-> South Bridge

South Bridge -> Video Memory (up to 35 GB/s) -> GPU (up to 8 GB/s) -> Display
Accelerate Motion Estimation with GPU

Loop(rows of macro blocks (MBs))
\hspace{1cm} Loop(columns of MBs)
\hspace{1cm} Loop(rows of search range (SR))
\hspace{1cm} Loop(columns of SR)
\hspace{1cm} SAD computation;
\hspace{1cm} SAD comparison;

Proposed Algorithm

Pass 1:
Loop(candidates per processing element){
    Loop(processing elements){
        SAD computation;
        SAD comparison;
    }
}

Pass 2:
Loop(rows of a MB){
    Loop(columns of a MB){
        Loop(processing elements){
            SAD comparison;
        }
    }
}
Experiment Environment

CPU:
- Intel Pentium IV 3.00 GHz
- 1024 MB memory

GPU:
- nVidia GeForce 6800 GT
- 128 MB video memory

Video sequence:
- Stefan in CIF (352x288) format with 30 fps
## Comparison of Integer-Pel ME

<table>
<thead>
<tr>
<th>Search range</th>
<th>Frame rate (s) (CPU only)</th>
<th>Frame rate (s) (CPU +GPU)</th>
<th>Speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>16x16</td>
<td>6.451</td>
<td>11.57</td>
<td>1.794</td>
</tr>
<tr>
<td>32x32</td>
<td>1.694</td>
<td>3.709</td>
<td>2.189</td>
</tr>
<tr>
<td>48x48</td>
<td>0.795</td>
<td>1.492</td>
<td>1.877</td>
</tr>
<tr>
<td>64x64</td>
<td>0.462</td>
<td>0.673</td>
<td>1.457</td>
</tr>
<tr>
<td>80x80</td>
<td>0.308</td>
<td>0.400</td>
<td>1.299</td>
</tr>
<tr>
<td>96x96</td>
<td>0.220</td>
<td>0.240</td>
<td>1.091</td>
</tr>
</tbody>
</table>
## Comparison of Half-Pel ME

<table>
<thead>
<tr>
<th>Search range</th>
<th>Frame rate (s) (CPU only)</th>
<th>Frame rate (s) (CPU + GPU)</th>
<th>Speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>16x16</td>
<td>0.805</td>
<td>7.275</td>
<td>9.037</td>
</tr>
<tr>
<td>32x32</td>
<td>0.217</td>
<td>3.091</td>
<td>14.24</td>
</tr>
<tr>
<td>48x48</td>
<td>0.102</td>
<td>1.374</td>
<td>13.47</td>
</tr>
<tr>
<td>64x64</td>
<td>0.060</td>
<td>0.650</td>
<td>10.83</td>
</tr>
<tr>
<td>80x80</td>
<td>0.040</td>
<td>0.393</td>
<td>9.825</td>
</tr>
<tr>
<td>96x96</td>
<td>0.028</td>
<td>0.236</td>
<td>8.429</td>
</tr>
</tbody>
</table>
Outline

- Introduction
- Stream processor
- Nowadays GPU
- Low power stream processor core
- Multi-core stream processor SoC
- Conclusion

Source:
Proposed Techniques

- Adaptive multi-thread (AMT)
- Configurable memory array (CMA)
- Early reject after transform (ERAT)
- Low power core pipeline (LPCP)
- Video accelerating instruction sets (VAIS)
Conventional Multi-Thread

- Inst0,1,2 have data dependency

<table>
<thead>
<tr>
<th>Thread 0</th>
<th>Thread 1</th>
<th>Thread 2</th>
<th>Thread 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inst0</td>
<td>Inst0</td>
<td>Inst0</td>
<td>Inst0</td>
</tr>
<tr>
<td>Inst1</td>
<td>Inst1</td>
<td>Inst1</td>
<td>Inst1</td>
</tr>
<tr>
<td>Inst2</td>
<td>Inst2</td>
<td>Inst2</td>
<td>Inst2</td>
</tr>
</tbody>
</table>
Convectional Multi-Thread

- Inst0, 1, 2 have data dependency

```
Thread 0  Thread 1  Thread 2  Thread 3
    Inst0       Inst0       Inst0       Inst0
    Inst1       Inst1       Inst1       Inst1
    Inst2       Inst2       Inst2       Inst2
```
Convectional Multi-Thread

Inst0,1,2 have data dependency

Thread 0  Thread 1  Thread 2  Thread 3

Inst0  Inst0  Inst0  Inst0
Inst1  Inst1  Inst1  Inst1
Inst2  Inst2  Inst2  Inst2
Convectional Multi-Thread

- Inst0,1,2 have data dependency

<table>
<thead>
<tr>
<th>Thread 0</th>
<th>Thread 1</th>
<th>Thread 2</th>
<th>Thread 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inst0</td>
<td>Inst0</td>
<td>Inst0</td>
<td>Inst0</td>
</tr>
<tr>
<td>Inst1</td>
<td>Inst1</td>
<td>Inst1</td>
<td>Inst1</td>
</tr>
<tr>
<td>Inst2</td>
<td>Inst2</td>
<td>Inst2</td>
<td>Inst2</td>
</tr>
</tbody>
</table>
Convectional Multi-Thread

- Inst0, 1, 2 have data dependency

<table>
<thead>
<tr>
<th>Thread 0</th>
<th>Thread 1</th>
<th>Thread 2</th>
<th>Thread 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inst0</td>
<td>Inst0</td>
<td>Inst0</td>
<td>Inst0</td>
</tr>
<tr>
<td>Inst1</td>
<td>Inst1</td>
<td>Inst1</td>
<td>Inst1</td>
</tr>
<tr>
<td>Inst2</td>
<td>Inst2</td>
<td>Inst2</td>
<td>Inst2</td>
</tr>
</tbody>
</table>
Convectional Multi-Thread

- Inst0,1,2 have data dependency

We get 3 cycles to hidden Inst0 latency
Adaptive Multi-Thread (AMT)

- Inst0,1,2 have data dependency

### Diagram

<table>
<thead>
<tr>
<th>Thread 0</th>
<th>Thread 1</th>
<th>Thread 2</th>
<th>Thread 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inst0</td>
<td>Inst0</td>
<td>Inst0</td>
<td>Inst0</td>
</tr>
<tr>
<td>Inst1</td>
<td>Inst1</td>
<td>Inst1</td>
<td>Inst1</td>
</tr>
<tr>
<td>Inst2</td>
<td>Inst2</td>
<td>Inst2</td>
<td>Inst2</td>
</tr>
</tbody>
</table>
## Adaptive Multi-Thread (AMT)

Inst0,1,2 have data dependency

<table>
<thead>
<tr>
<th>Thread 0</th>
<th>Thread 1</th>
<th>Thread 2</th>
<th>Thread 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inst0</td>
<td>Inst0</td>
<td>Inst0</td>
<td>Inst0</td>
</tr>
<tr>
<td>Inst1</td>
<td>Inst1</td>
<td>Inst1</td>
<td>Inst1</td>
</tr>
<tr>
<td>Inst2</td>
<td>Inst2</td>
<td>Inst2</td>
<td>Inst2</td>
</tr>
</tbody>
</table>
Adaptive Multi-Thread (AMT)

- Inst0,1,2 have data dependency

Inst0 only take 1 cycle. Use data forward to pass data to Inst1
Adaptive Multi-Thread (AMT)

Inst0,1,2 have data dependency

Thread 0  Thread 1  Thread 2  Thread 3

Inst0     Inst0     Inst0     Inst0
Inst1     Inst1     Inst1     Inst1
Inst2     Inst2     Inst2     Inst2
Adaptive Multi-Thread (AMT)

- Inst0,1,2 have data dependency
Adaptive Multi-Thread (AMT)

- Inst0,1,2 have data dependency

<table>
<thead>
<tr>
<th>Thread 0</th>
<th>Thread 1</th>
<th>Thread 2</th>
<th>Thread 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inst0</td>
<td>Inst0</td>
<td>Inst0</td>
<td>Inst0</td>
</tr>
<tr>
<td>Inst1</td>
<td>Inst1</td>
<td>Inst1</td>
<td>Inst1</td>
</tr>
<tr>
<td>Inst2</td>
<td>Inst2</td>
<td>Inst2</td>
<td>Inst2</td>
</tr>
</tbody>
</table>
Adaptive Multi-Thread

Inst0,1,2 have data dependency

<table>
<thead>
<tr>
<th>Thread 0</th>
<th>Thread 1</th>
<th>Thread 2</th>
<th>Thread 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inst0</td>
<td>Inst0</td>
<td>Inst0</td>
<td>Inst0</td>
</tr>
<tr>
<td>Inst1</td>
<td>Inst1</td>
<td>Inst1</td>
<td>Inst1</td>
</tr>
<tr>
<td>Inst2</td>
<td>Inst2</td>
<td>Inst2</td>
<td>Inst2</td>
</tr>
</tbody>
</table>
Adaptive Multi-Thread (AMT)

Inst0,1,2 have data dependency

<table>
<thead>
<tr>
<th>Thread 0</th>
<th>Thread 1</th>
<th>Thread 2</th>
<th>Thread 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inst0</td>
<td>Inst0</td>
<td>Inst0</td>
<td>Inst0</td>
</tr>
<tr>
<td>Inst1</td>
<td>Inst1</td>
<td>Inst1</td>
<td>Inst1</td>
</tr>
<tr>
<td>Inst2</td>
<td>Inst2</td>
<td>Inst2</td>
<td>Inst2</td>
</tr>
</tbody>
</table>
Adaptive Multi-Thread (AMT)

Inst0,1,2 have data dependency

<table>
<thead>
<tr>
<th>Thread 0</th>
<th>Thread 1</th>
<th>Thread 2</th>
<th>Thread 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inst0</td>
<td>Inst0</td>
<td>Inst0</td>
<td>Inst0</td>
</tr>
<tr>
<td>Inst1</td>
<td>Inst1</td>
<td>Inst1</td>
<td>Inst1</td>
</tr>
<tr>
<td>Inst2</td>
<td>Inst2</td>
<td>Inst2</td>
<td>Inst2</td>
</tr>
<tr>
<td>Inst0</td>
<td>Inst1</td>
<td></td>
<td>Inst1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Inst2</td>
</tr>
</tbody>
</table>
Adaptive Multi-Thread (AMT)

Inst0,1,2 have data dependency

<table>
<thead>
<tr>
<th>Thread 0</th>
<th>Thread 1</th>
<th>Thread 2</th>
<th>Thread 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inst0</td>
<td>Inst0</td>
<td>Inst0</td>
<td>Inst0</td>
</tr>
<tr>
<td>Inst1</td>
<td>Inst1</td>
<td>Inst1</td>
<td>Inst1</td>
</tr>
<tr>
<td><strong>Inst2</strong></td>
<td><strong>Inst2</strong></td>
<td><strong>Inst2</strong></td>
<td><strong>Inst2</strong></td>
</tr>
</tbody>
</table>
Adaptive Multi-Thread (AMT)

Inst0,1,2 have data dependency

We get 6 cycles to hidden Inst1 latency
Comparison

- **Conventional Multi-Thread**
  - Thread numbers = Hidden cycles
  - More on chip threads = More silicon area

- **Adaptive Multi-Thread**
  - Hidden cycles = Short Inst. * Thread Numbers
  - 40% power reduction
  - 2 times performance improve
Stream Processing Model

Input Stream Data

Input Regs

Temp Regs

Stream Kernel

Const. Regs

Output Regs

Output Stream Data

Ref. Data
Stream Processing Model

CMA

Stream Data

Input Regs

Stream Kernel

Const. Regs

Output Regs

Output Stream Data

Ref. Data
Conventional Dedicated Buffer

- Worst case design
  - 16 attributes per threads

Whole Memory Bank

- Thread 0
- Thread 1
- Thread 2
- Thread 3

16 attr.
Conventional Dedicated Buffer

- When application takes only 8 attributes per thread
- 50% utilization rate

Whole Memory Bank

8 attr.

Thread 0
Unused
Thread 1
Unused
Thread 2
Unused
Thread 3
Unused
Configurable Memory Array (CMA)

- Worst case and best case design
- Reconfigure physical memory to logic register files

Whole Memory Bank

<table>
<thead>
<tr>
<th>Thread 0</th>
<th>Thread 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread 2</td>
<td>Thread 3</td>
</tr>
<tr>
<td>Thread 4</td>
<td>Thread 5</td>
</tr>
<tr>
<td>Thread 6</td>
<td>Thread 7</td>
</tr>
</tbody>
</table>

8 attr.
Stream Processing Model

CMA

Input Stream Data

Input Regs

Stream Kernel

output Regs

Output Stream Data
Stream Processing Model

Level C Data Reuse
86% bandwidth reduction

CMA
Early Reject After Transform (ERAT)

- Reject the redundant triangles as early as possible
  - Reduce power
  - Increase performance

- Content aware processing
Triangle rejection type

- Outside rejection triangle
- Back face cull triangle
- Zero area triangle
Execution Flow Chart

Fetch Thread

Transform Exec Stage

IF Transform All Threads

Yes

Fetch Thread

IF Reject

No

Light&Tex Exec Stage
Low Power Core Pipeline

- **2 ISSUE VLIW**
  - Scalar type for control instruction
  - SIMD type for parallel computing instruction

- **Max 800 MOPS**
  - 16 vectors 8 bit fix point operation

- **Max 400 MFOPS**
  - Each slot issue 4 vectors floating point operation
Low Power Core Pipeline (LPCP)

Instruction level clock gating
Low Power Core Pipeline (LPCP)

Software control data forward
Low Power Core Pipeline (LPCP)

Software control data write back
Video Accelerating Inst. Sets (VAIS)

128 Bits

VLIW Instruction

Slot0

Slot1

Instruction Decode

Execution Unit 0 (Adder)
Execution Unit 1 (Adder)

......

Execution Unit 7 (Multiplier)
Execution Unit 8 (4Adder)
Execution Unit 9 (SOP)

Data Dispatch

Register File Controller

V RF
O RF
R RF
C RF

Connect to DMA
Video Accelerating Inst. Sets (VAIS)

- **Instruction Decode**
  - Slot0: VLIW Instruction
  - Slot1: 128 Bits
  - SAD PE0, SAD PE1, SAD PE2, SAD PE3
  - SAD PE14, SAD PE15

- **Data Dispatch**
  - Execution Unit 8 (4Adder)
  - Adder Tree

- **Register File Controller**
  - Current Block
  - Search Range Buffer

- **Connect to DMA**
System Architecture

SoC for Mobile Multimedia Applications

- Host Processor
- Rendering Engine
- Memory Controller

Peripherals

System Bus

Stream Processor Core

- Host I/F
- Instruction Memory
- Configurable Memory Array (CMA)
- Cache Tag
- General Registers
- Kernel Execution Unit
- Temporal Registers
- Early Rejection After Transformation (ERAT) Module

LCD

Stream Data Buffer (Off-Chip Memory)
# System Specification

<table>
<thead>
<tr>
<th>Process Technology</th>
<th>TSMC 0.18um CMOS 1P6M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chip Size</td>
<td>8.91 mm²</td>
</tr>
<tr>
<td>Supply Voltage</td>
<td>1.8 V</td>
</tr>
<tr>
<td>Clock Frequency</td>
<td>50 MHz</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>8.6* mw</td>
</tr>
<tr>
<td>Features</td>
<td>OpenGL ES 2.0 Support</td>
</tr>
<tr>
<td></td>
<td>Shader Model 3.0</td>
</tr>
<tr>
<td></td>
<td>Video encoding IME capability</td>
</tr>
</tbody>
</table>

*Measure at fully activate specular lighting
Chip Photo
Performance Result

*Optima performance
**Performance increase depending on reject rate

Power (mw)  M vertices/s

LPPC + CMA + AMT + ERAT  LPPC + CMA + AMT + ERAT  Opt*

59  28.1  17  8.6  0.4  1.2  2.5  4.6  x2.0  x1.84  12.5

x3.04  x2.0  x1.84  **
## Comparison Table

<table>
<thead>
<tr>
<th></th>
<th>ISSCC`05</th>
<th>ISSCC`06</th>
<th>This Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process (um)</td>
<td>0.18</td>
<td>0.18</td>
<td>0.18</td>
</tr>
<tr>
<td>Clock (MHZ)</td>
<td>200</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>Polygon Rate (M vertices/s)</td>
<td>50</td>
<td>120</td>
<td>12.5</td>
</tr>
<tr>
<td>Power (mW)</td>
<td>155</td>
<td>157</td>
<td>8.6</td>
</tr>
<tr>
<td>Area (mm$^2$)</td>
<td>23.04</td>
<td>16</td>
<td>8.91</td>
</tr>
<tr>
<td>Features</td>
<td>Graphics</td>
<td>Graphics</td>
<td>Graphics, Video</td>
</tr>
</tbody>
</table>
Outline

- Introduction
- Stream processor
- Nowadays GPU
- Low power stream processor core
- Multi-core stream processor SoC
- Conclusion
Conclusion

Stream processor is an efficient processor architecture for multimedia signal processing
- Unified architecture
- Scalable architecture

A low power stream processor core is developed with five optimized design techniques

A multi-core stream processor with dynamic task scheduling is developed
Thank You!