Design, Implementation and Evaluation of Built-in Functions on Parallel Programming Model in SMYLE OpenCL

Noriko Etani, Takuji Hieda and Hiroyuki Tomiyama

Research Organization of Science and Technology
Ritsumeikan University, Japan
Outline

Background

- Many-core architecture SMYLE
- SMYLE OpenCL parallel programming model

Research Questions

1. How to implement the software processor cores on the FPGA.

2. Portability of the built-in functions in SMYLE OpenCL.

Background 1: Many-core architecture SMYLEref

- IL1: L1 Instruction cache
- DL1: L1 Data cache

A two-dimensional meshed network on chip (NoC)
Background 2: SMYLEref Evaluation Platform

For 128 cores by The Univ. of Electro-Communications, Japan

8 cores/FPGA
Virtex-6 FPGA ML605
Background 2: **SMYLEref Evaluation Platform**

For 4 cores by Ritsumeikan University, Japan

4 cores/FPGA
Virtex-6 FPGA ML605
Background 3: **SMYLE OpenCL Parallel Programming Model**

**Data Parallel Execution**
- Host
- Command Que
- Kernel 3, Kernel 2, Kernel 1, Kernel 0
- Device: Compute Unit
  - PE

**Task Parallel Execution**
- Host
- Command Que
- Kernel 3, Kernel 2, Kernel 1, Kernel 0
- Device: Compute Unit
  - Kernel 0, Kernel 1, Kernel 2, Kernel 3
  - PE

**PE**: Processing Element
Research Questions

1. **How to implement** the software processor cores on the FPGA.

2. **Portability** of the built-in functions in SMYLE OpenCL.

3. High-speed and low-power **performance**.
## Specification of ML605 and Virtex-6

<table>
<thead>
<tr>
<th></th>
<th>ML605</th>
<th>Virtex-6</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FPGA</strong></td>
<td>Virtex-6 XC6VLX240T-1FFG1156</td>
<td></td>
</tr>
<tr>
<td><strong>SDRAM</strong></td>
<td>DDR3 SODIMM (512MB)</td>
<td></td>
</tr>
<tr>
<td><strong>Input / Output Ports</strong></td>
<td>UART, USB, DVI output, CF, SMA</td>
<td></td>
</tr>
<tr>
<td><strong>Clocking</strong></td>
<td>200MHz oscillator</td>
<td>66MHz socketed oscillator</td>
</tr>
<tr>
<td><strong>CMOS</strong></td>
<td>40nm, 1.0V</td>
<td></td>
</tr>
<tr>
<td><strong>Logic Cells</strong></td>
<td>241,152</td>
<td></td>
</tr>
<tr>
<td><strong>CLB Slices</strong></td>
<td>37,680</td>
<td></td>
</tr>
<tr>
<td><strong>Block RAM</strong></td>
<td>14,976Kbit</td>
<td></td>
</tr>
<tr>
<td><strong>Max User I/O</strong></td>
<td>720</td>
<td></td>
</tr>
<tr>
<td><strong>Power Consumption</strong></td>
<td>Static Power: 3.6W</td>
<td>Total Power: 6.5W</td>
</tr>
</tbody>
</table>
Core Assignment in SMYLEref

SMYLEref Architecture

FPGA

Core Assignment: 10MHz/core

Master Core: Core 0
Slave Core: Core 1, Core 2, Core 3
## Software Development Environment

<table>
<thead>
<tr>
<th>PC</th>
<th>Sony VAIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel® Core™ i7-2640M Processor</td>
</tr>
<tr>
<td></td>
<td>Clock Speed: 2.80GHz</td>
</tr>
<tr>
<td></td>
<td>Power Consumption(Max TDP): 35W</td>
</tr>
<tr>
<td>OS</td>
<td>Windows 7 Professional Service Pack 1</td>
</tr>
<tr>
<td>VM</td>
<td>VMware Player 4.0.3</td>
</tr>
<tr>
<td>HOST</td>
<td>32-bit Fedora 16</td>
</tr>
</tbody>
</table>

- Mips-geyser-linux cross compilation environment
- Portable OpenCL Kernel
- BenchMark tests
- Communication Terminal with ML605
Software Architecture of Master Core

Host

Application

Built-in Functions

mips-geyser-linux

SMYLEref Architecture

FPGA

Device

Application

Built-in Functions

SMYLE-Ref Layer

IO Library (ml501io.c)

Soft Floating Point Emulation

SMYLEref Architecture

FPGA

To control device cores for executing the parallel application program.

To control FPGA on ML605.
Constraint Condition of SMYLEref

No SIMD
Vector data typed and vector calculation NOT AVAILABLE

No Floating point arithmetic device

For HOST
The MPFR library
A C library for multiple-precision floating-point computation with correct rounding
GMP
A free library for arbitrary precision arithmetic, operating on signed integers, rational numbers, and floating point numbers

For KERNEL
Routines for soft floating point emulation
A host core executes a host program on OS.
• The kernel is loaded on a master core.
• A master core starts three slave cores, and four cores run in parallel.
Parallel Programming Functions

Thread (→ Device core)
- Master core → master thread
- Slave core → slave thread

Exclusive control
- mutex, critical section
- Functions in SMYLE-Ref layer
  - sr_mutex_init to initialize mutex
  - sr_mutex_lock to lock mutex
  - sr_mutex_unlock to unlock mutex

Condition variable
- A function to control resource scheduling
  Until some event happens, a condition variable allows one device core to block.
Example of Data Parallel Program

```c
int main()
{
    ....
    printf("Hello, World! \n"); //Data Parallel
    ....
    exit(0);
}
```

- All device cores display the same message.
Example of Task Parallel Program

```c
int main()
{
    ...... 

    // to get core ID
    sr_core_id_t my_id = sr_get_core_id();
    // Task Parallel
    if ( my_id == 0 ) printf("core0\n");
    if ( my_id == 1 ) printf("core1\n");
    if ( my_id == 2 ) printf("core2\n");
    if ( my_id == 3 ) printf("core3\n");
    ...... 

    exit(0);
}
```

- Each device core displays each message.
extern sr_mutex_t mutex;
//shared resource
int __attribute__ ((section (".bss2"))) global_counter = 0;

int kernel_app()
{
    sr_mutex_lock( &mutex );
    {
        global_counter++;
    }
    sr_mutex_unlock( &mutex );
}
Example of mutex initialization

A master core initializes mutex.

```c
sr_mutex_t __attribute__((section(".bss2"))) mutex;
int main()
{
    sr_core_id_t my_id = sr_get_core_id();
    //to initialize mutex by master core
    if ( my_id == 0 ) {
        sr_mutex_init(&mutex);
        sleep(1);
    }
    else {
        sleep(1);
    }
    //to execute parallel application
    kernel_app();
    .....
    exit(0);
}
```
Example of conditional variable

```c
while (1) {
    // waiting loop
    while (remain == 0) {}  
    sr_mutex_lock( &mutex );
    {  // !!!CRITICAL SECTION!!!
        i = queue[rp];
        rp++;
        remain--;
        if(rp == MAX_QUEUE_NUM) rp = 0;
    }
    sr_mutex_unlock( &mutex );
    if(i == END_DATA) break;
}
```

To make the present device core waited for execution until a queue will receive a data
Performance Evaluation

System
- SMYLE OpenCL Kernel on 4 cores of SMYLEref
- Pthread Code with 4 threads of 32-bit Fedora 16 on PC

BenchMark
- **Addition 1**
  4 threads/device cores add 12 data each, and display the sum total.
- **Addition 2**
  12 data are split to 3 data for 4 threads/device cores, and displays the sum total of 12 data.
- **Producer/Consumer**
  2 threads/device cores of producers and 2 threads/device cores of consumers
Results of Benchmark Applications

Power Consumption:
- Intel Core i7: 35W
- SMYLE: 6.5W

By calculation:
- $6.5W \times 0.5 = 3.25W$
- $3.25W / 35W = 0.0928$
Research Questions

1. How to implement the software processor cores on the FPGA.
   - Master/Slave cores assignment
   - Host/Device operations on a framework of OpenCL
   - Double speed, one-tenth power saving

2. Portability of the built-in functions in SMYLE OpenCL.

Implementation of built-in functions

To extend the original OpenCL semantics giving our system’s original limitation and interpretation for SMYLEref

- Full specification 200 functions
- Specification to be supported 110 functions
- Specification not to be supported 90 functions
Implementation Issues

To develop **static library** and utilize uClibc standard library

To embed **routines of soft floating point emulation in kernel**
- Addition, subtraction, multiplication and division operation "+", "-", "*", "/"
- Comparison "+", "<=", ">
  But, "<" and ">" are available for positive number
- Conversion of the data type
  "Expansion of data width in the return value”,  
  "Reduction of data width in the return value”, and  
  "Conversion to integer typed”
- Handling of zero (-0)

To develop **barrier function** of synchronization function
Control of Barrier Function

- To watch synchronization of plural blocks of application programs between device cores.
- To make each processing waited until all application programs call barrier function.
Code of Barrier Function

1: int barrier(){
2:    // waiting for barrier
3:    while(global_escape != 0){
4:    }
5:    // increment of barrier counter
6:    sr_mutex_lock( &mutex );
7:    { // CRITICAL SECTION
8:         global_barrier +=1;
9:    }
10:    sr_mutex_unlock( &mutex );
11:    // waiting for escaping
12:    while(global_barrier < SR_NUM_OF_CORES){
13:    }
14:    // increment of escape counter
15:    sr_mutex_lock( &mutex );
16:    { // CRITICAL SECTION
17:        global_escape +=1;
18:        // to initialize escape counter and barrier counter
19:        if((global_escape == SR_NUM_OF_CORES) &&
20:            (global_barrier == SR_NUM_OF_CORES)) {
21:            global_escape = 0;
22:            global_barrier = 0;
23:        }
24:    }
25:    sr_mutex_unlock( &mutex );
26: }

- All device cores call barrier function.
- Then, all device cores can escape from waiting.
### Implementation Results

<table>
<thead>
<tr>
<th>Built-in functions</th>
<th>OpneCL 1.2</th>
<th>SMYLE OpenCL</th>
<th>Using uClibc Standard Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>Math</td>
<td>60 functions</td>
<td>Host: 3, Kernel: 3</td>
<td>Host: 26, Kernel: 0</td>
</tr>
<tr>
<td>Integer</td>
<td>17</td>
<td>Host: 14, Kernel: 14</td>
<td>Host: 1, Kernel: 1</td>
</tr>
<tr>
<td>Common</td>
<td>9</td>
<td>Host: 4, Kernel: 1</td>
<td>-</td>
</tr>
<tr>
<td>Geometric</td>
<td>7</td>
<td>Host: 4, Kernel: 0</td>
<td>-</td>
</tr>
<tr>
<td>Relational</td>
<td>16</td>
<td>Host: 10, Kernel: 10</td>
<td>-</td>
</tr>
<tr>
<td>Barrier</td>
<td>1</td>
<td>Host: -, Kernel: 1</td>
<td>-</td>
</tr>
</tbody>
</table>

Host: 35 functions by the static library, 26 functions by uClibc. Kernel: 28 functions by the static library, 1 functions by uClibc.
Built-in Functions using routines of soft floating point emulation

Math Functions (3)
  • fdim, mad, nextafter

Common Functions (1)
  • step

Relational Functions (8)
  • isequal, isnotequal, isordered
    Only the positive value is available as follows:
  • isgreater, isgreateerequal, isless, islessequal, islessgreater
Causes of non-workable functions

Not to be supported by uclibc standard library
  ▪ Host  38 functions
  ▪ Kernel 38 functions

No overloading functions for C++
  ▪ Host  30 functions
  ▪ Kernel 31 functions

Operation errors
  ▪ Host  6 functions
  ▪ Kernel 12 functions
## Built-in Functions to be supported

### Math Function

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Host</th>
<th>Kernel</th>
<th>Function Name</th>
<th>Host</th>
<th>Kernel</th>
<th>Function Name</th>
<th>Host</th>
<th>Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>acos</td>
<td>※</td>
<td></td>
<td>cosh</td>
<td>※</td>
<td></td>
<td>nextafter</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>acospi</td>
<td>※</td>
<td></td>
<td>cospi</td>
<td>※</td>
<td></td>
<td>pow</td>
<td>※</td>
<td>○</td>
</tr>
<tr>
<td>asin</td>
<td>※</td>
<td></td>
<td>expml</td>
<td>※</td>
<td></td>
<td>rsqrt</td>
<td>※</td>
<td>○</td>
</tr>
<tr>
<td>asinpi</td>
<td>※</td>
<td></td>
<td>fabs</td>
<td>※</td>
<td></td>
<td>sin</td>
<td>※</td>
<td>○</td>
</tr>
<tr>
<td>atan</td>
<td>※</td>
<td></td>
<td>fdim</td>
<td>○</td>
<td>○</td>
<td>sinh</td>
<td>※</td>
<td>○</td>
</tr>
<tr>
<td>atan2</td>
<td>※</td>
<td></td>
<td>floor</td>
<td>※</td>
<td></td>
<td>sinpi</td>
<td>※</td>
<td>○</td>
</tr>
<tr>
<td>atanpi</td>
<td>※</td>
<td></td>
<td>ldexp</td>
<td>※</td>
<td></td>
<td>sqrt</td>
<td>※</td>
<td>○</td>
</tr>
<tr>
<td>ceil</td>
<td>※</td>
<td></td>
<td>log</td>
<td>※</td>
<td></td>
<td>tan</td>
<td>※</td>
<td>○</td>
</tr>
<tr>
<td>copysign</td>
<td>※</td>
<td></td>
<td>log10</td>
<td>※</td>
<td></td>
<td>tanh</td>
<td>※</td>
<td>○</td>
</tr>
<tr>
<td>cos</td>
<td>※</td>
<td></td>
<td>mad</td>
<td>○</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Integer Function

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Host</th>
<th>Kernel</th>
<th>Function Name</th>
<th>Host</th>
<th>Kernel</th>
<th>Function Name</th>
<th>Host</th>
<th>Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs</td>
<td>※</td>
<td>※</td>
<td>rhadd</td>
<td>○</td>
<td>○</td>
<td>min</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>abs_diff</td>
<td>○</td>
<td>○</td>
<td>clamp</td>
<td>○</td>
<td>○</td>
<td>mul24</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>add_sat</td>
<td>○</td>
<td>○</td>
<td>mad24</td>
<td>○</td>
<td>○</td>
<td>rotate</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>clz</td>
<td>○</td>
<td>○</td>
<td>mad_sat</td>
<td>○</td>
<td>○</td>
<td>sub_sat</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>hadd</td>
<td>○</td>
<td>○</td>
<td>max</td>
<td>○</td>
<td>○</td>
<td>popcount</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

※ uClibs standard library  ○ static library
### Built-in Functions to be supported (cont'd)

#### Common Function

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Host</th>
<th>Kernel</th>
<th>Function Name</th>
<th>Host</th>
<th>Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>degrees</td>
<td>○</td>
<td></td>
<td>radians</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>step</td>
<td>○</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Geometric Function

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Host</th>
<th>Kernel</th>
<th>Function Name</th>
<th>Host</th>
<th>Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>dot</td>
<td>○</td>
<td></td>
<td>distance</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>normalize</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Relational Function

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Host</th>
<th>Kernel</th>
<th>Function Name</th>
<th>Host</th>
<th>Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>isequal</td>
<td>○</td>
<td>○</td>
<td>isless</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>isnotequal</td>
<td>○</td>
<td>○</td>
<td>islessequal</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>isgreater</td>
<td>○</td>
<td>○</td>
<td>islessgreater</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>isgreaterequal</td>
<td>○</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Barrier Function

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Host</th>
<th>Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>barrier</td>
<td>-</td>
<td>○</td>
</tr>
</tbody>
</table>

※ uClibs standard library  ○ static library
ResearchQuestions

1. How to implement the software processor cores on the FPGA.
   - Master/Slave cores assignment
   - Host/Device operations on a framework of OpenCL
   - Double speed, one-tenth power saving

2. Portability of the built-in functions in SMYLE OpenCL.
   - Host Portability
     35 functions in static library
     26 functions in uClibc standard library
   - Kernel Portability
     28 functions in static library
     1 functions in uClibc standard library
     Barrier function depending on SMYLEref architecture

Performance Evaluation

System
- SMYLE OpenCL Kernel on 4 cores of SMYLEref
- Portable OpenCL Kernel on 4 threads of 32-bit Fedora 16 on PC

BenchMark
- **Benchmark 1**
  Data Parallel Programming Model
  12 Integer functions with barrier calls
- **Benchmark 2**
  Task Parallel Programming Model
  Different order of 12 Integer functions with barrier calls
Results of Performance Evaluation

Clock Speed:
- Intel Core i7: 2.80GHz
- SMYLE: 10MHz

Power Consumption:
- Intel Core i7: 35W
- SMYLE: 6.5W

By calculation,
\[ 6.5W \times 0.5 = 3.25W \]
\[ \frac{3.25W}{35W} = 0.0928 \]
Research Questions

1. **How to implement** the software processor cores on the FPGA.
   - Master/Slave cores assignment
   - Host/Device operations on a framework of OpenCL
   - Double speed, one-tenth power saving

2. **Portability** of the built-in functions in SMYLE OpenCL.
   - Host Portability
     35 functions in static library
     26 functions in uClibc standard library
   - Kernel Portability
     28 functions in static library
     1 functions in uClibc standard library
   Barrier function depending on SMYLERef architecture

3. **High-speed and low-power performance.**
   - Double speed, one-tenth power saving in SMYLE OpenCL
Conclusion

1. Parallel programming model on a framework of OpenCL is implemented on the software processor cores of FPGA.

2. Build-in functions are portable except for one barrier function depending on the target many-core architecture.

3. The execution speed is twice as fast and the power consumption is reduced to $\frac{1}{10}$. 
Future Works

To enhance high-speed and low-power performance of SMYLE OpenCL

Conversion of the kernel functions into custom FPGA hardware accelerators

To enhance portability of SMYLE OpenCL

Workability of math functions on the kernel
Acknowledgment

New Energy and Industrial Technology Development Organization (NEDO), Japan
Conclusion

1. Parallel programming model on a framework of OpenCL is implemented on the software processor cores of FPGA.

2. Build-in functions are portable except for one barrier function depending on the target many-core architecture.

3. The execution speed is twice as fast and the power consumption is reduced to 1/10.
Mips-geyser-linux Cross Compilation Environment

Linux source archive
geyserlinux.20110331.2331

<table>
<thead>
<tr>
<th>Tool Package</th>
<th>version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binutils</td>
<td>2.20.1a</td>
</tr>
<tr>
<td>gcc</td>
<td>4.3.4</td>
</tr>
<tr>
<td>uClIBC</td>
<td>0.9.31</td>
</tr>
<tr>
<td>gdb</td>
<td>7.2a</td>
</tr>
<tr>
<td>lzo</td>
<td></td>
</tr>
<tr>
<td>mpfr</td>
<td>2.4.2</td>
</tr>
<tr>
<td>gmp</td>
<td>4.3.2</td>
</tr>
<tr>
<td>mpc</td>
<td>0.9</td>
</tr>
</tbody>
</table>