

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

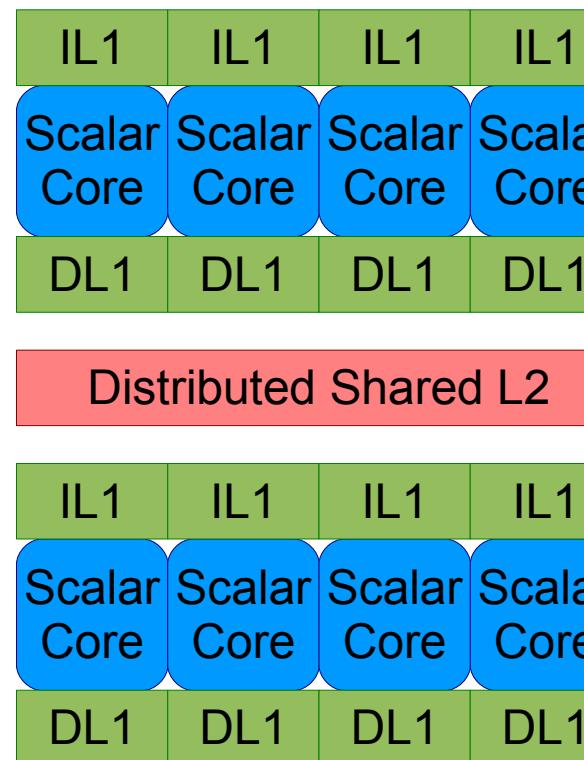
Background

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

Research Questions

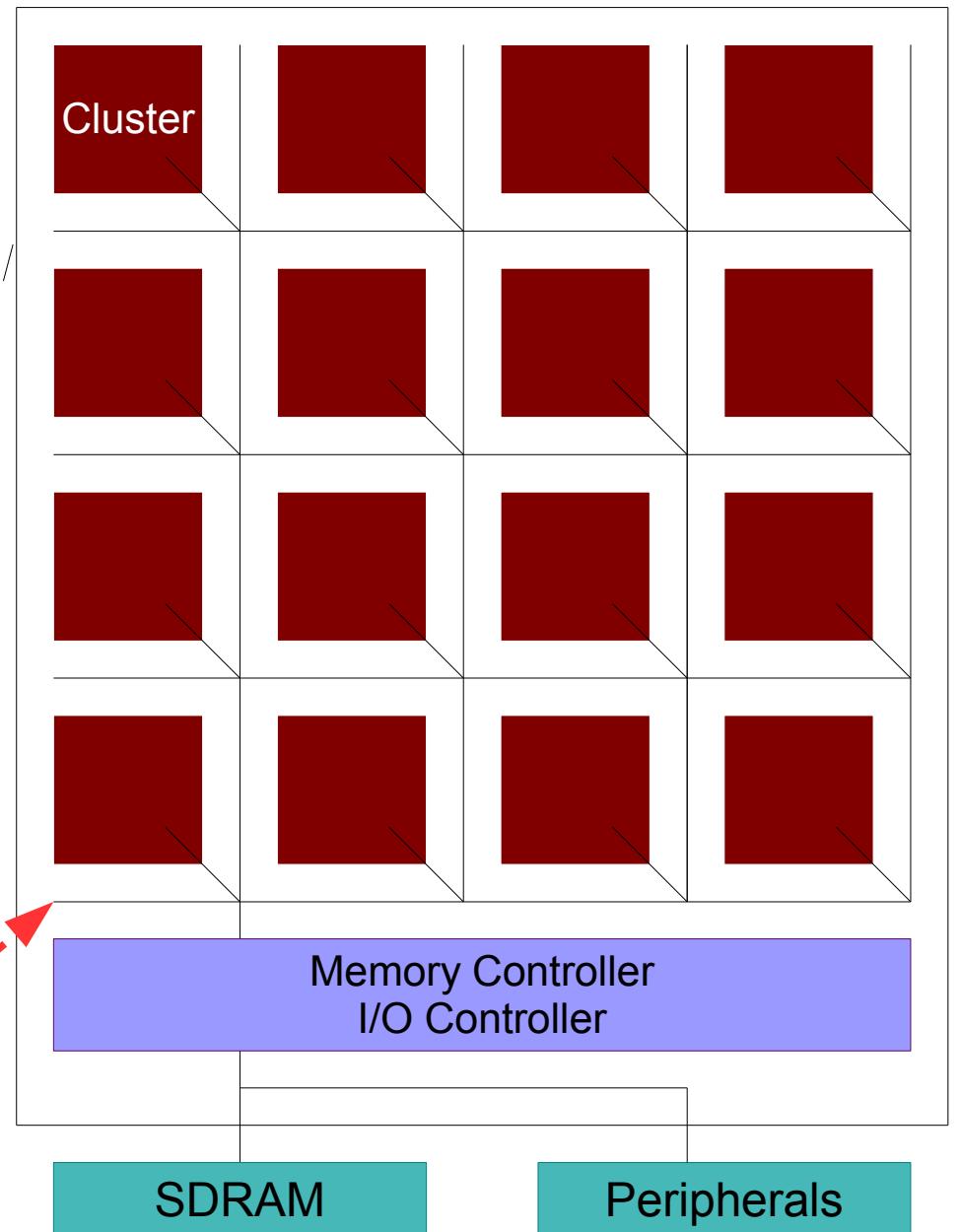
1. How to implement the software processor cores on the FPGA.
2. Portability of the built-in functions in SMYLE OpenCL.
3. High-speed and low-power performance.

Background 1: Many-core architecture SMYLEref



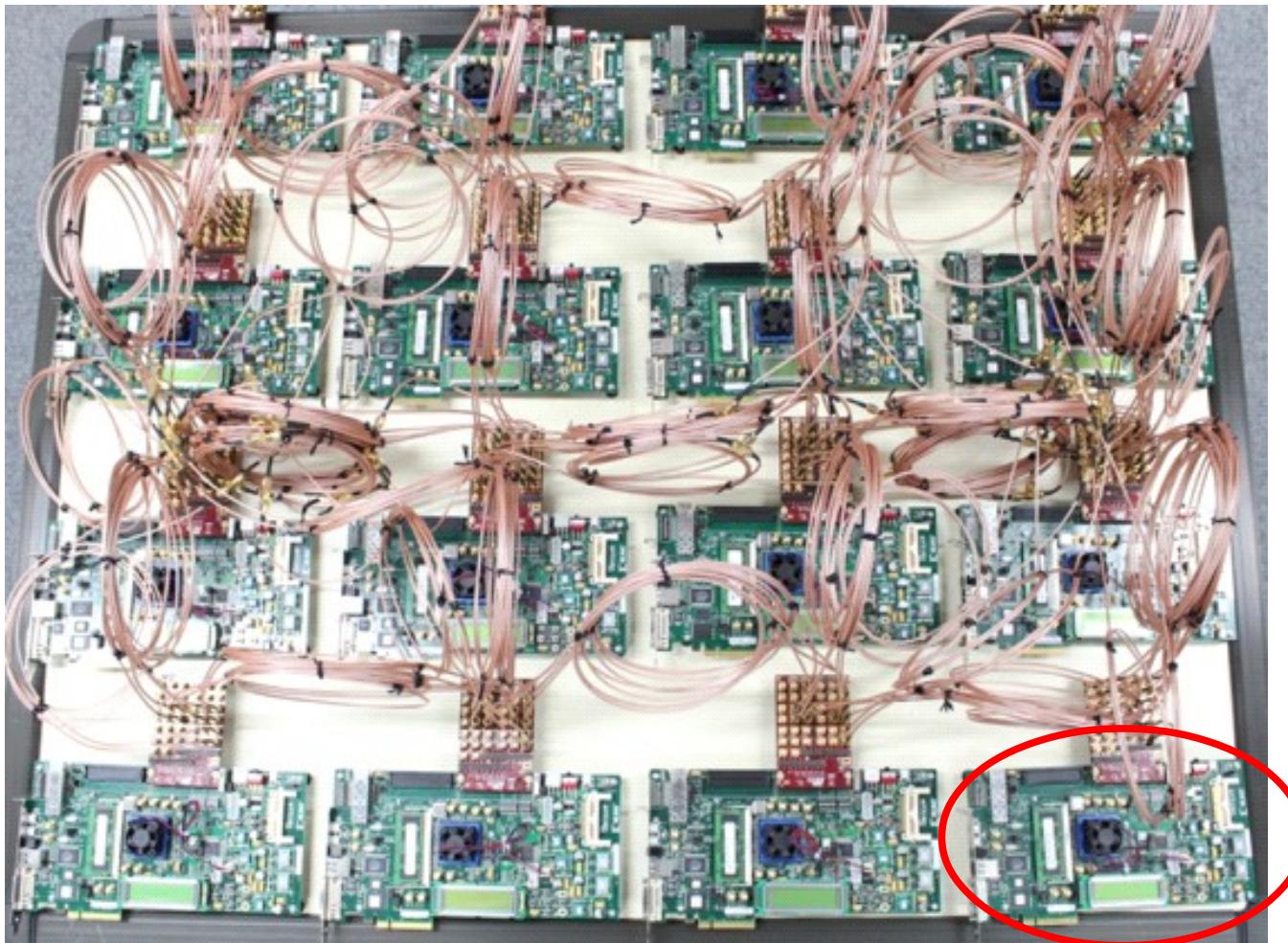
IL1 : L1 Instruction cache
DL1: L1 Data cache

A two-dimentional meshed network on chip(NoC)



Background 2: SMYLEref Evaluation Platform

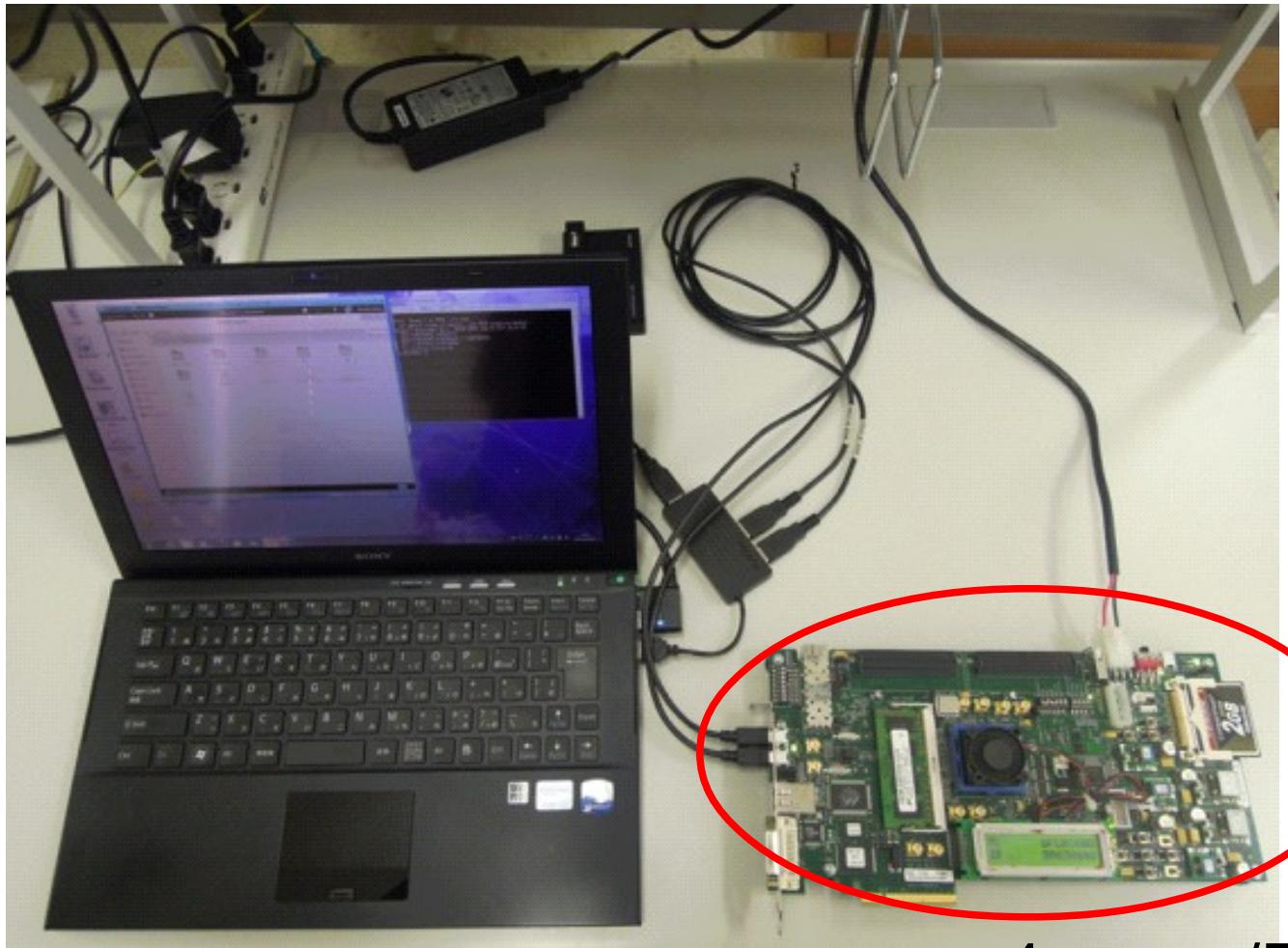
For 128 cores by The Univ. of Electro-Communications, Japan



8 cores/FPGA
Virtex-6 FPGA ML605

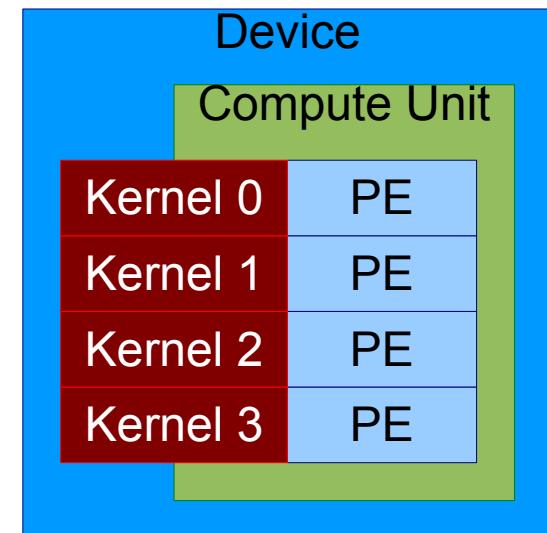
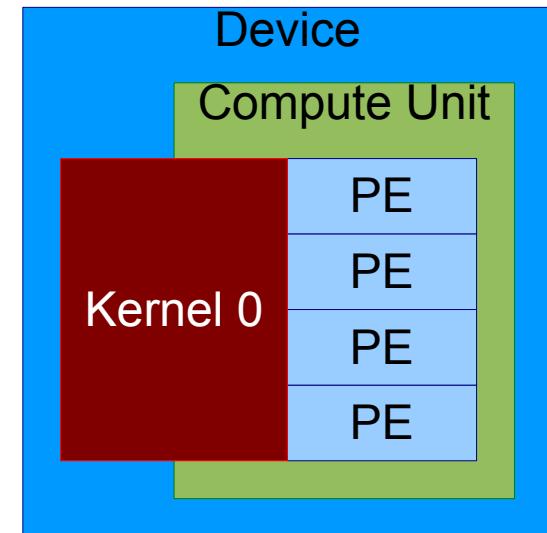
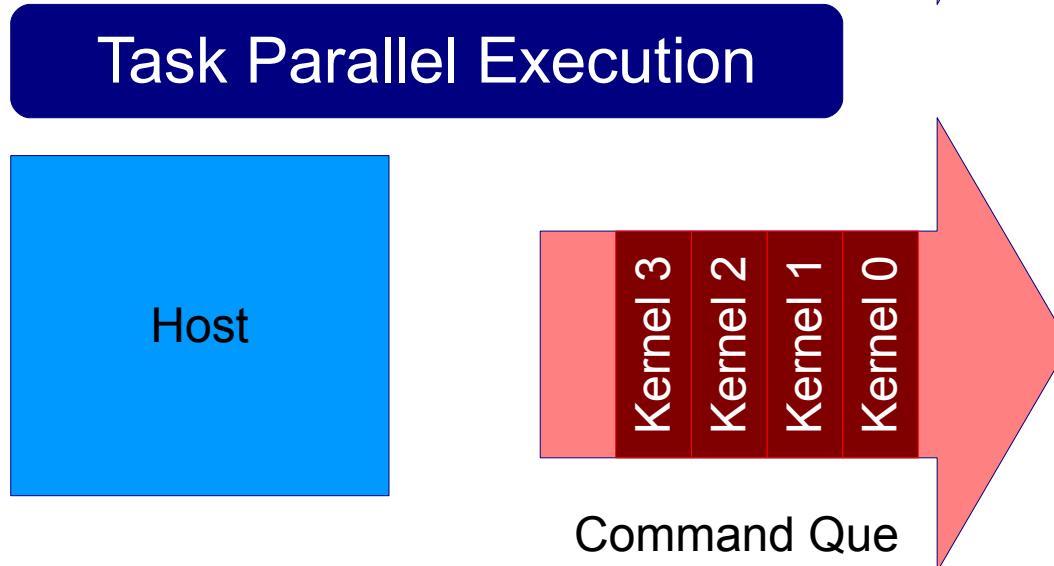
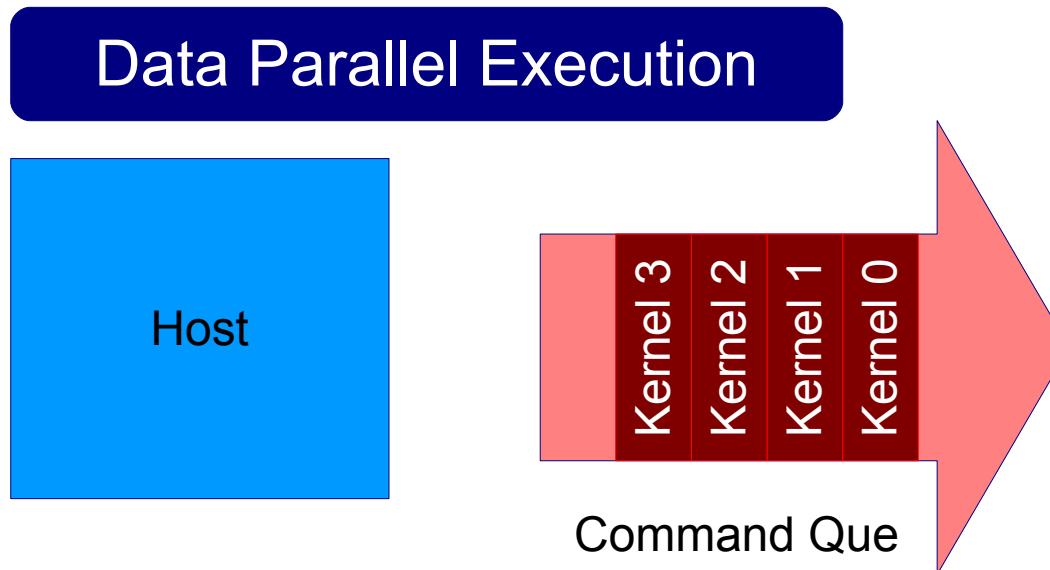
Background 2: SMYLEref Evaluation Platform

For 4 cores by Ritsumeikan University, Japan



4 cores/FPGA
Virtex-6 FPGA ML605

Background 3: SMYLE OpenCL Parallel Programming Model



PE: Processing Element

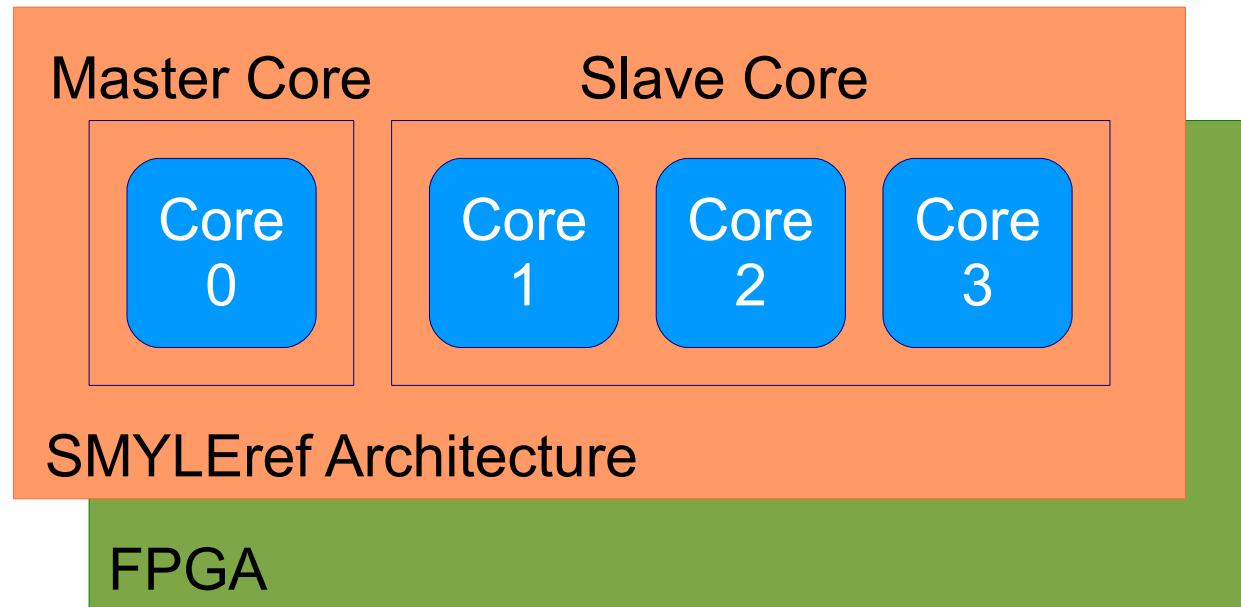
ResearchQuestions

- 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

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

Core Assignment in SMYLEref



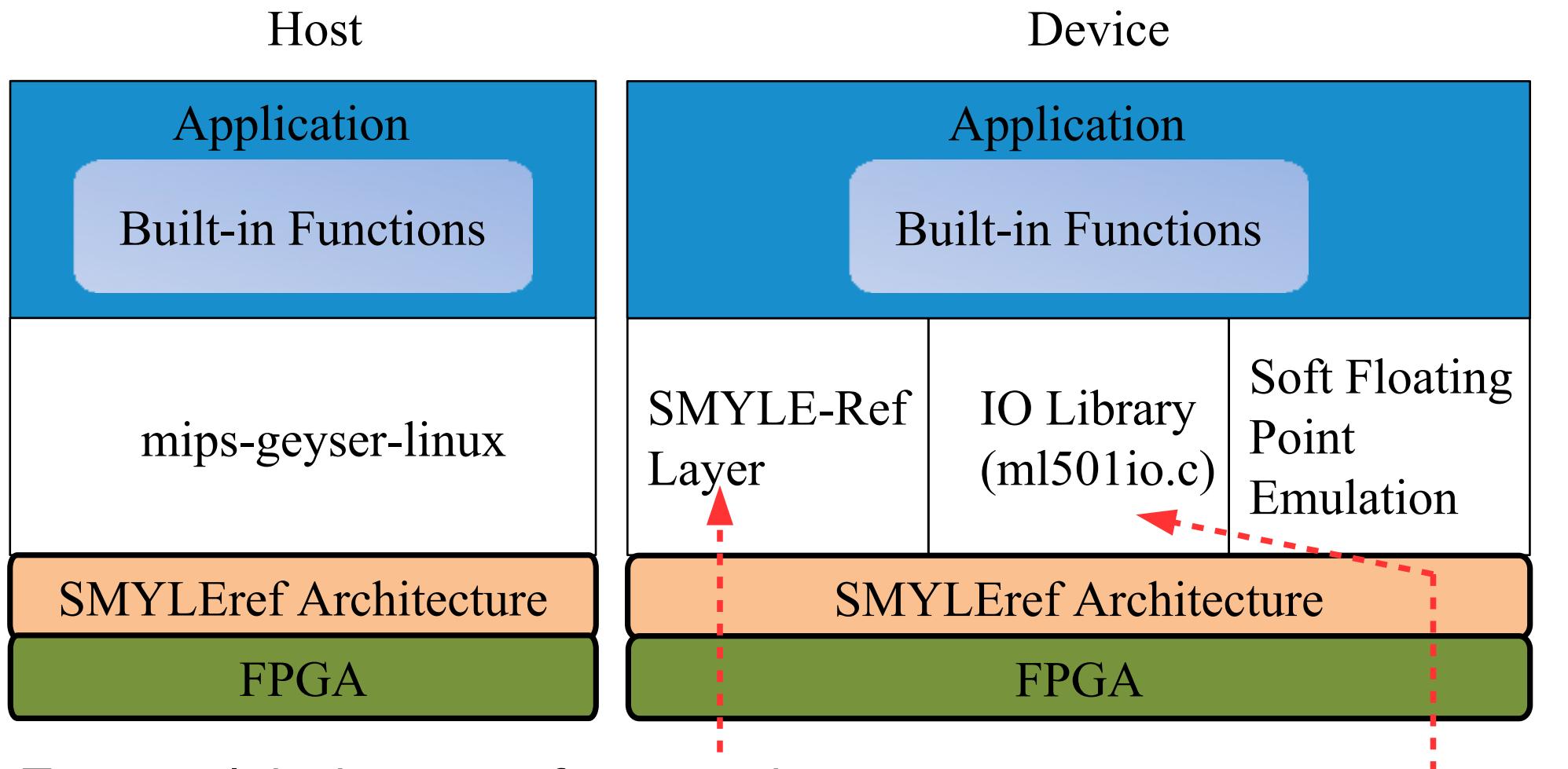
the frequency of supplying : 10MHz/core

Software Development Environment

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

- Mips-geyser-linux cross compilation environment
- Portable OpenCL Kernel
- BenchMark tests
- Communication Terminal with ML605

Software Architecture of Master Core



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

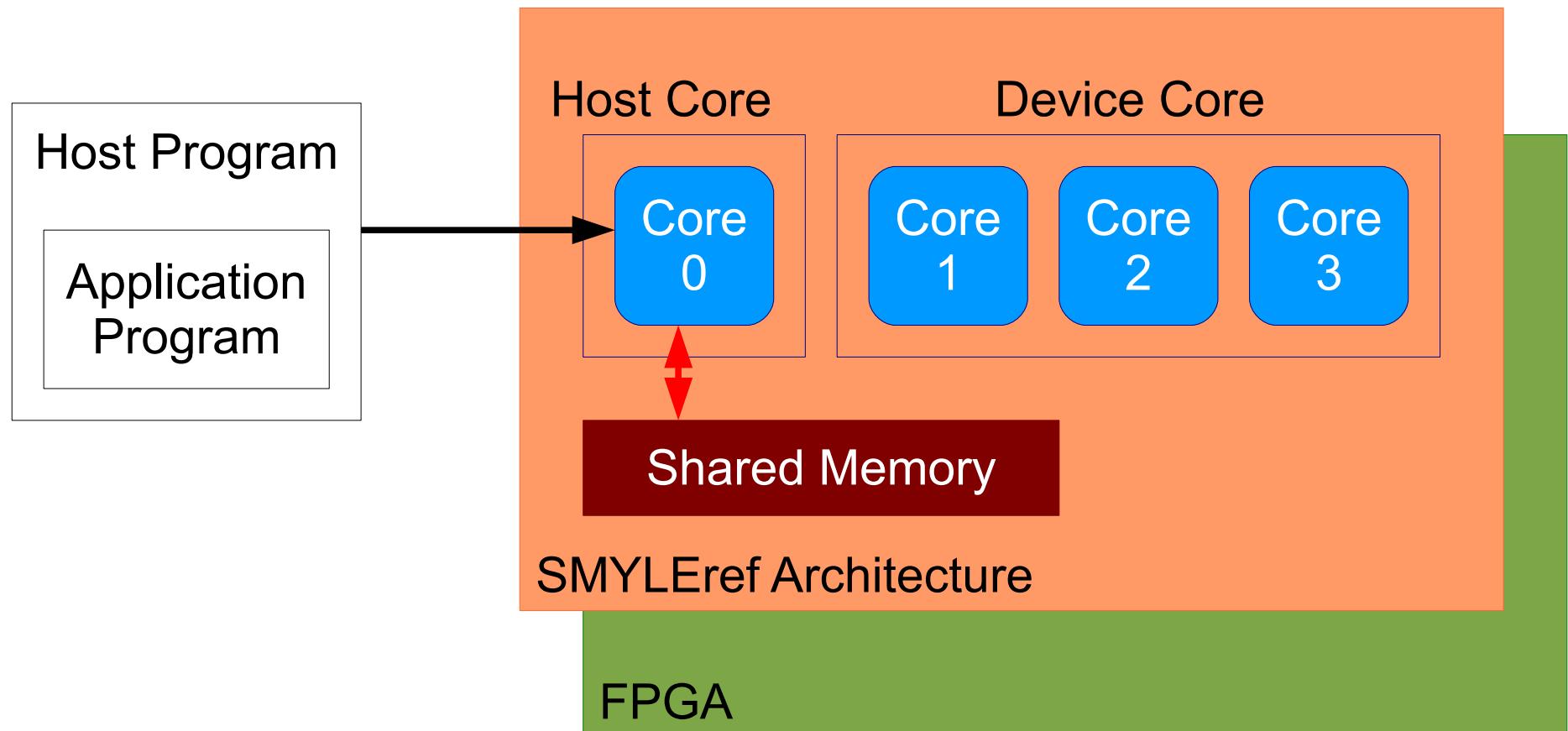
GMP

A free library for arbitrary precision arithmetic, operating on signed integers, rational numbers, and floating point numbers

For KERNEL

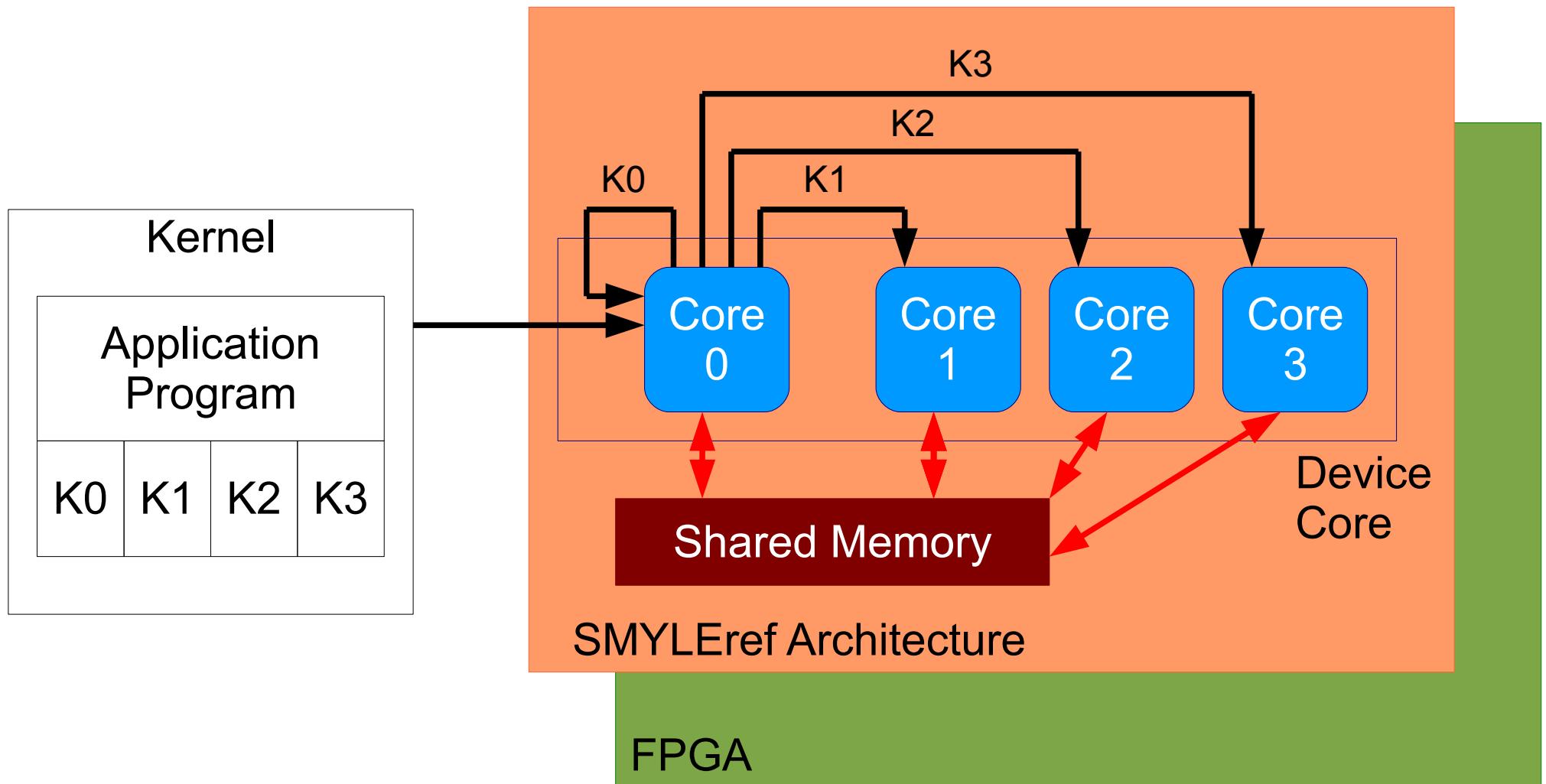
Routines for soft floating point emulation

Host Operation



- A host core executes a host program on OS.

Device Operation



- 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

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

- All device cores display the same message.

Example of Task Parallel Program

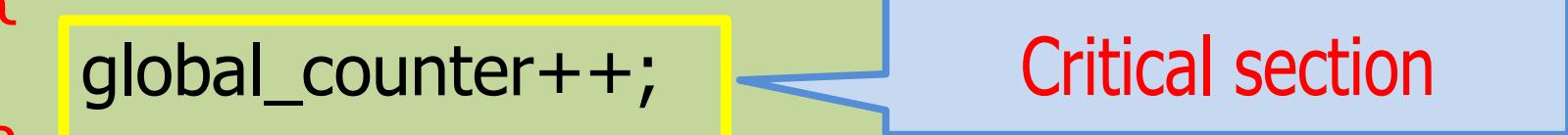
```
int main()
{
    .....
    //to get core ID
    sr_core_id_t my_id = sr_get_core_id();
    //Task Parallel
    if ( my_id == 0 ) printf("core0\n");
    if ( my_id == 1 ) printf("core1\n");
    if ( my_id == 2 ) printf("core2\n");
    if ( my_id == 3 ) printf("core3\n");
    .....
    exit(0);
}
```

- Each device core displays each message.

Example of mutex and critical section

```
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 );
}
```



The diagram illustrates a critical section. A yellow rectangular box highlights the line of code `global_counter++;`. A blue arrow points from this yellow box to a blue rectangular box containing the text **Critical section**.

Example of mutex initialization

```
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);  
}
```

- A master core initializes mutex.

Example of conditional variable

```
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

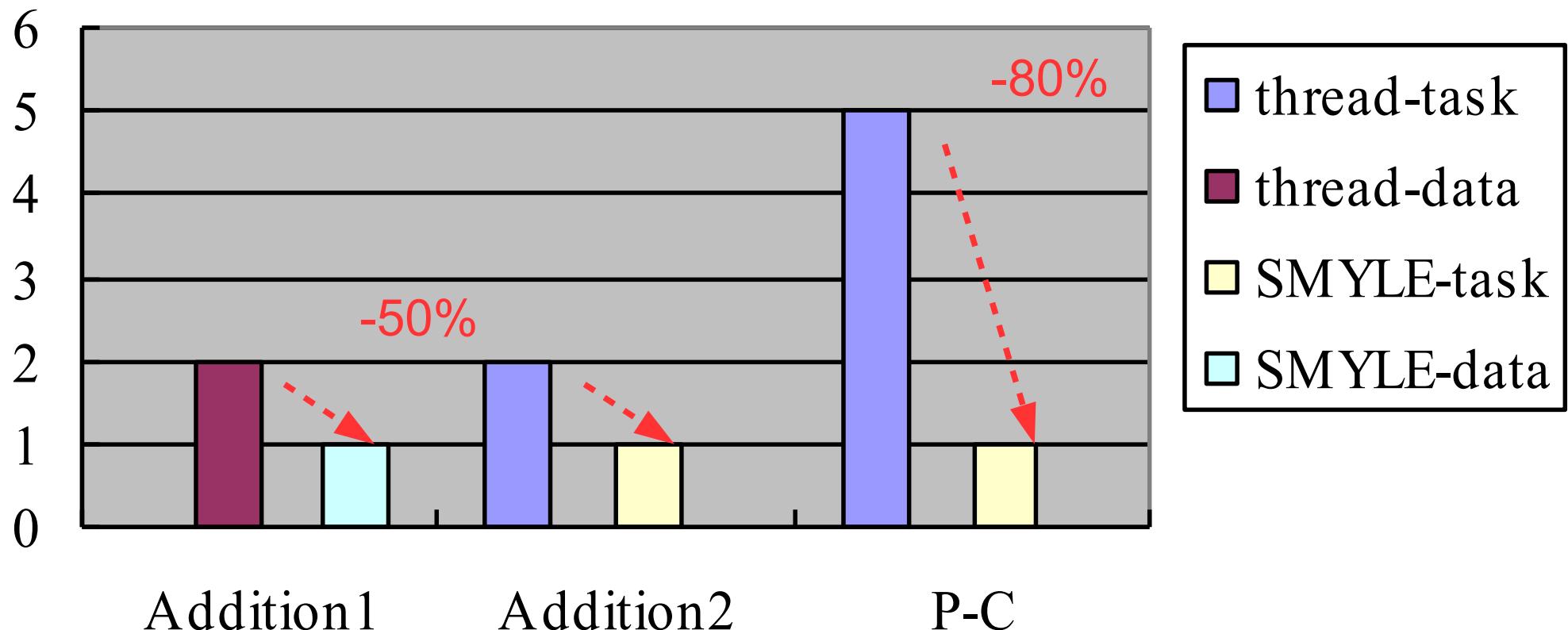
System

- 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,
 $6.5W * 0.5 = 3.25W$
 $3.25W / 35W = 0.0928$

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.
3. High-speed and low-power performance.

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

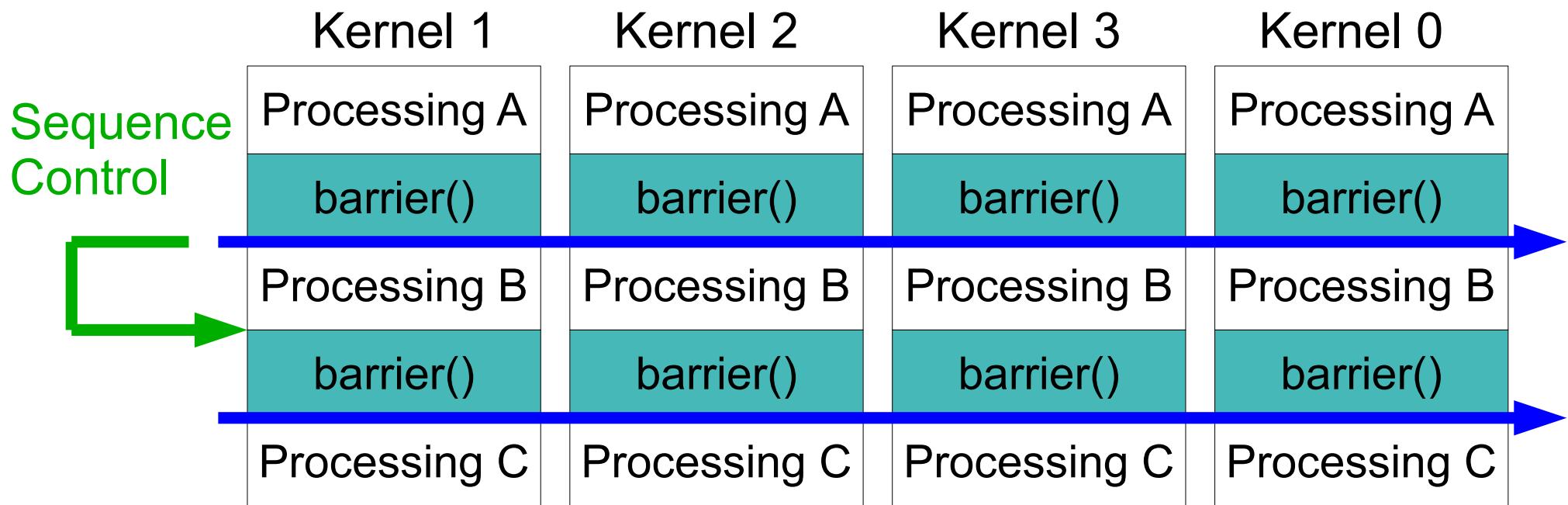
To develop **static library** and utilize uClibc standard library

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.

Synchronization Control

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

Built-in functions	OpenCL 1.2	SMYLE OpenCL	Using uClibc Standard Library
Math	60 functions	Host: 3, Kernel: 3	Host: 26, Kernel: 0
Integer	17	Host: 14, Kernel: 14	Host: 1, Kernel: 1
Common	9	Host: 4, Kernel: 1	-
Geometric	7	Host: 4, Kernel: 0	-
Relational	16	Host: 10, Kernel: 10	-
Barrier	1	Host: -, Kernel: 1	-

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`, `isgreaterequal`, `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

Function Name	Host	Kernel	Function Name	Host	Kernel	Function Name	Host	Kernel
acos	✗		cosh	✗		nextafter	○	○
acospi	✗		cospi	✗		pow	✗	
asin	✗		expml	✗		rsqrt	✗	
asinpi	✗		fabs	✗		sin	✗	
atan	✗		fdim	○	○	sinh	✗	
atan2	✗		floor	✗		sinpi	✗	
atanpi	✗		ldexp	✗		sqrt	✗	
ceil	✗		log	✗		tan	✗	
copysign	✗		log10	✗		tanh	✗	
cos	✗		mad	○	○			

Integer Function

Function Name	Host	Kernel	Function Name	Host	Kernel	Function Name	Host	Kernel
abs	✗	✗	rhadd	○	○	min	○	○
abs_diff	○	○	clamp	○	○	mul24	○	○
add_sat	○	○	mad24	○	○	rotate	○	○
clz	○	○	mad_sat	○	○	sub_sat	○	○
hadd	○	○	max	○	○	popcount	○	○

✗ uClibs standard library

○ static library

Built-in Functions to be supported (cont'd)

Common Function

Function Name	Host	Kernel	Function Name	Host	Kernel	Function Name	Host	Kernel
degrees	O		radians	O		mix	O	
step	O	O						

Geometric Function

Function Name	Host	Kernel	Function Name	Host	Kernel	Function Name	Host	Kernel
dot	O		distance	O		length	O	
normalize	O							

Relational Function

Function Name	Host	Kernel	Function Name	Host	Kernel	Function Name	Host	Kernel
isequal	O	O	isless	O	O	isordered	O	O
isnotequal	O	O	islessequal	O	O	bitselect	O	O
isgreater	O	O	islessgreater	O	O	select	O	O
isgreaterequal	O	O						

Barrier Function

Function Name	Host	Kernel
barrier	-	O

* uClibs standard library

○ static library

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.

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 fuctions with barrier calls

Results of Performance Evaluation

Clock Speed:

Intel Core i7 2.80GHz

SMYLE 10MHz

Power Consumption:

Intel Core i7 35W

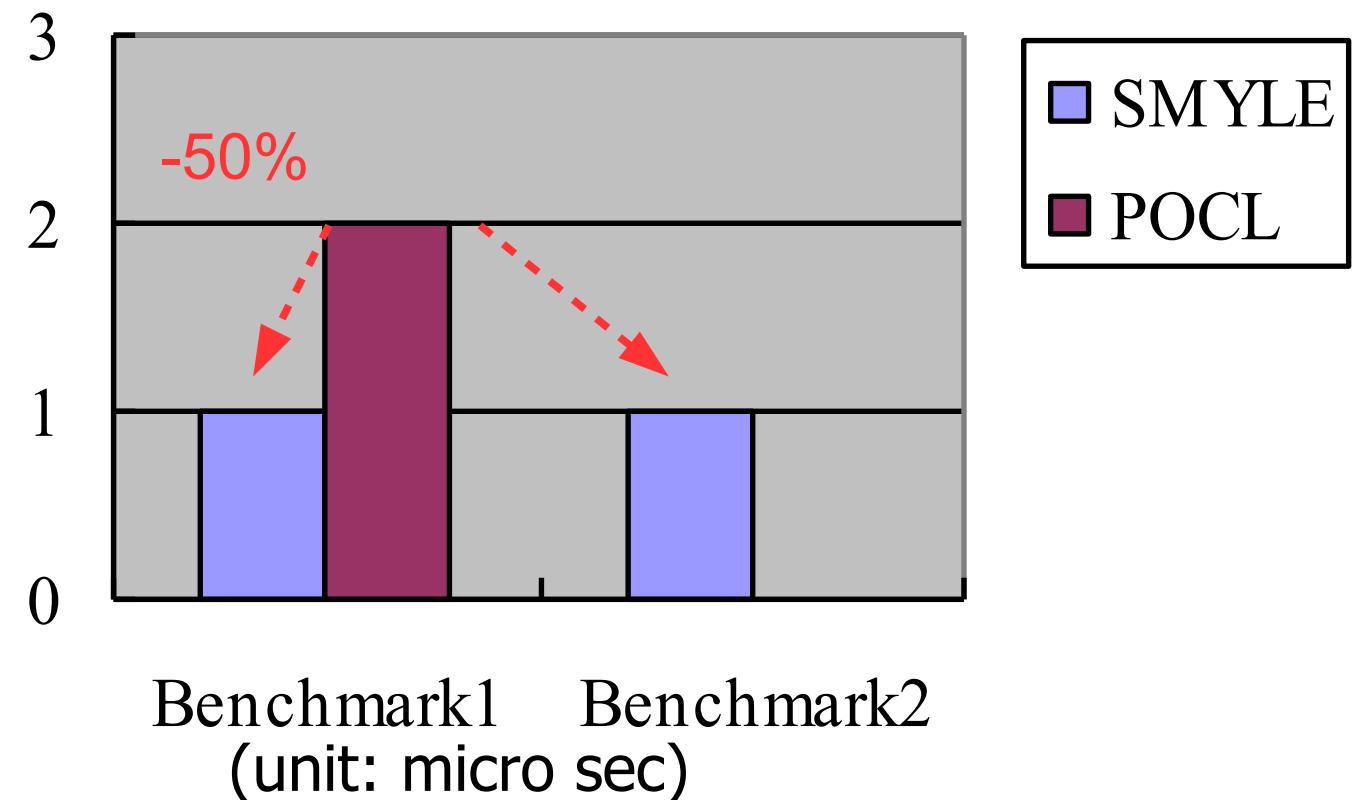
SMYLE 6.5W



By calculation,

$$6.5W * 0.5 = 3.25W$$

$$3.25W / 35W = 0.0928$$



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.

- 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 **1/10**.

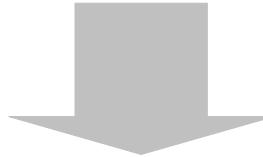
Future Works

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

<u>Tool Package</u>	<u>version</u>
Binutils	2.20.1a
gcc	4.3.4
uClibc	0.9.31
gdb	7.2a
lzo	
mpfr	2.4.2
gmp	4.3.2
mpc	0.9