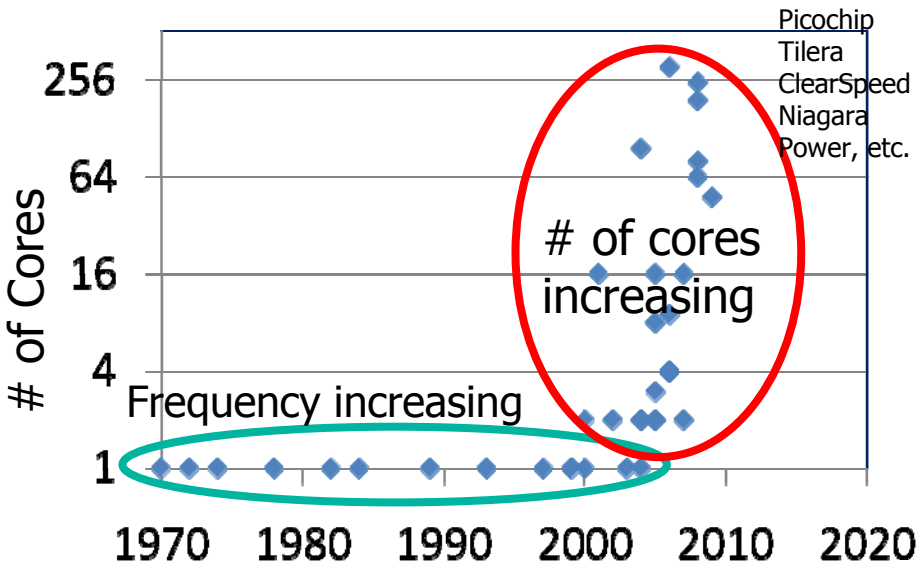
Empowered by Innovation NEC

A Software Development Toolset for Multi-Core Processors

Yuichi Nakamura
System IP Core Research Labs.
NEC Corp.

Motivations

- Embedded Systems: Performance enhancement by multi-core systems
- Multi-core Systems need the parallelization of software
- How to develop parallelized software on multi-core
  - Realistic parallelization
  - Keeping hard real time constraints
  - Detail verification and debug



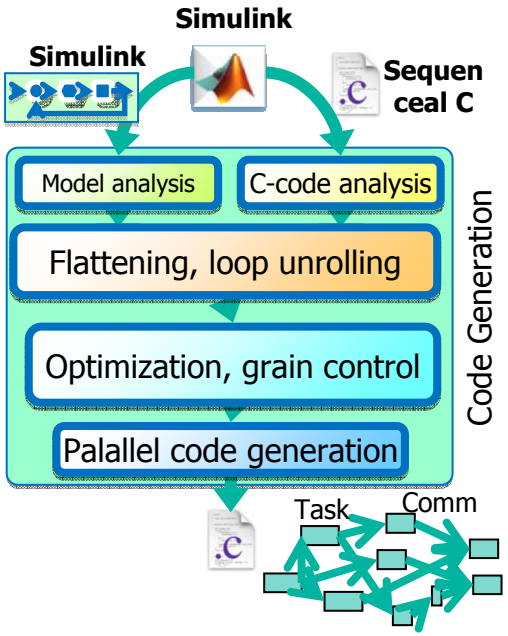
Challenges:
Enhancement of the productivity of software development for multi-core systems with high performance and high reliability.

Our Approaches

From an industry's view, 3 approaches/tools are proposed

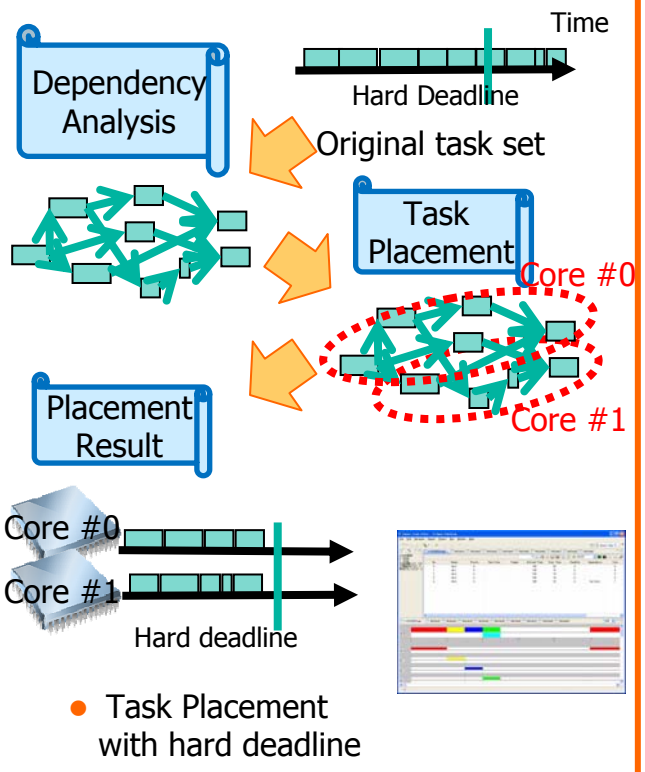
1.Parallelization from models

Support parallelization from Simulink models



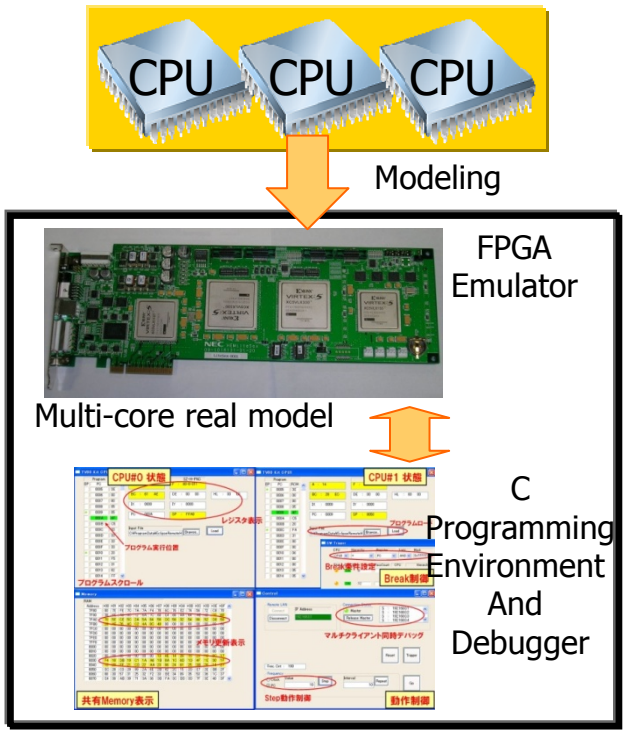
- Task Parallelization in short time

2.Task placement with constraints



- Task Placement with hard deadline

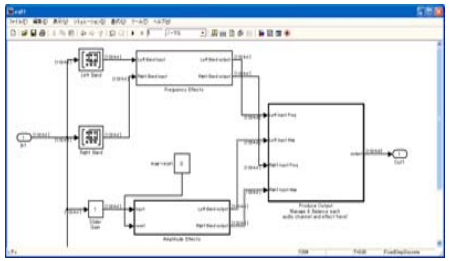
3.Debugging SW by using FPGA



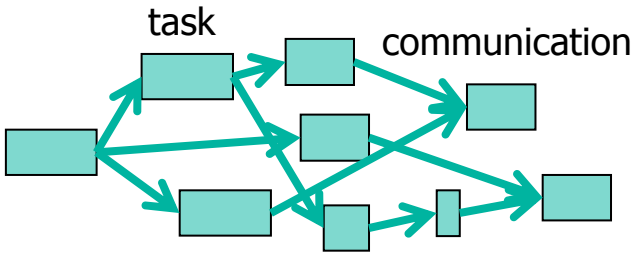
1-1.Parallel C Code Generation from Simulink Models

- A tool to generate parallel C code from Simulink models
- The tool enables users to develop parallel software without using parallel APIs explicitly.

Parallelization Method:
1. treat each of leaf blocks in models as a task.
2. signal completion of processing of tasks to descendant tasks with synchronized messages.



Simulink models

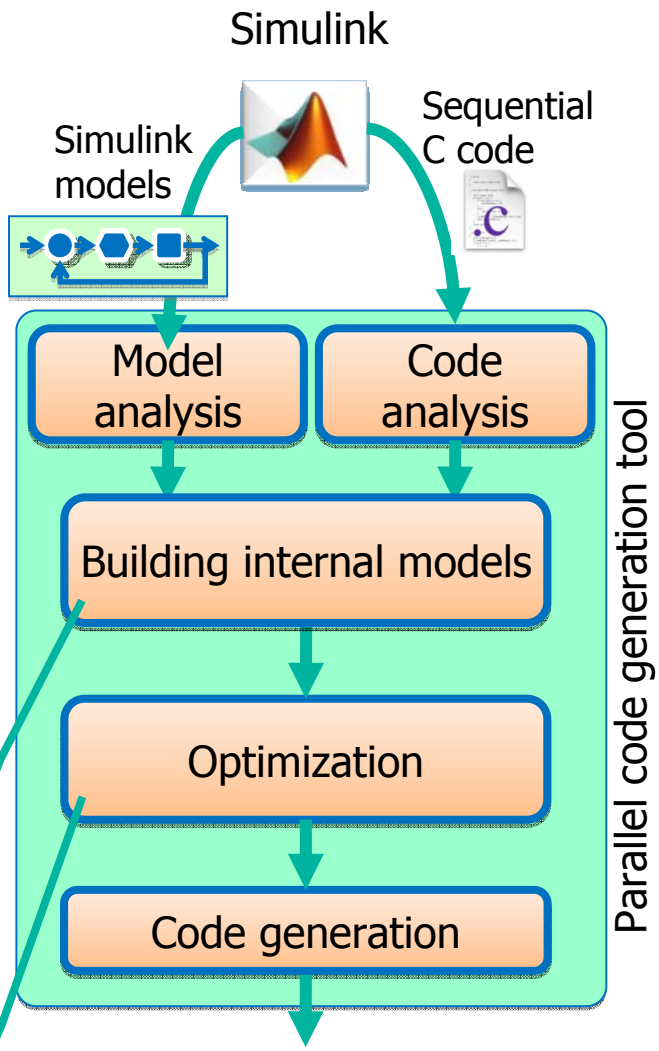


Preliminary experimental result (audio equalizer, 100% before parallelization)	
2-core PC	60%
4-core PC	38%

* Number of tasks in parallelized software: 57

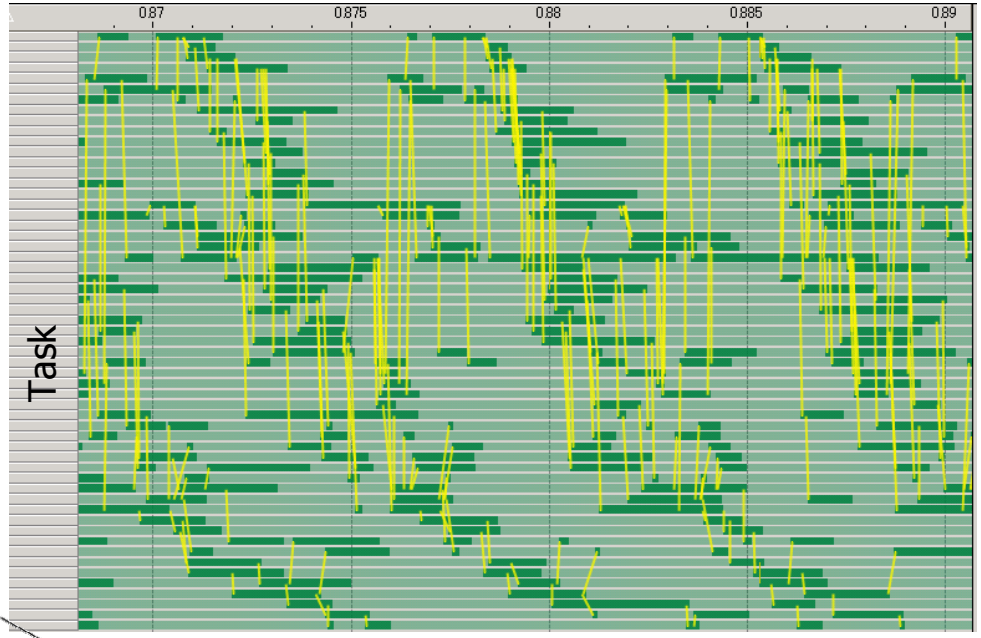
To extract structural parallelism expressed in models through removing block hierarchy and loop structures.

To optimize task granularity for small communication overhead.

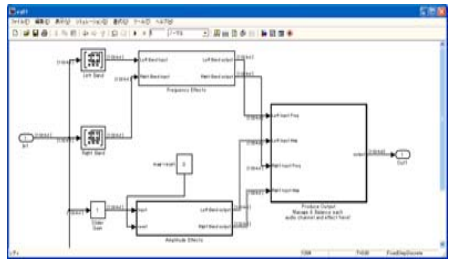


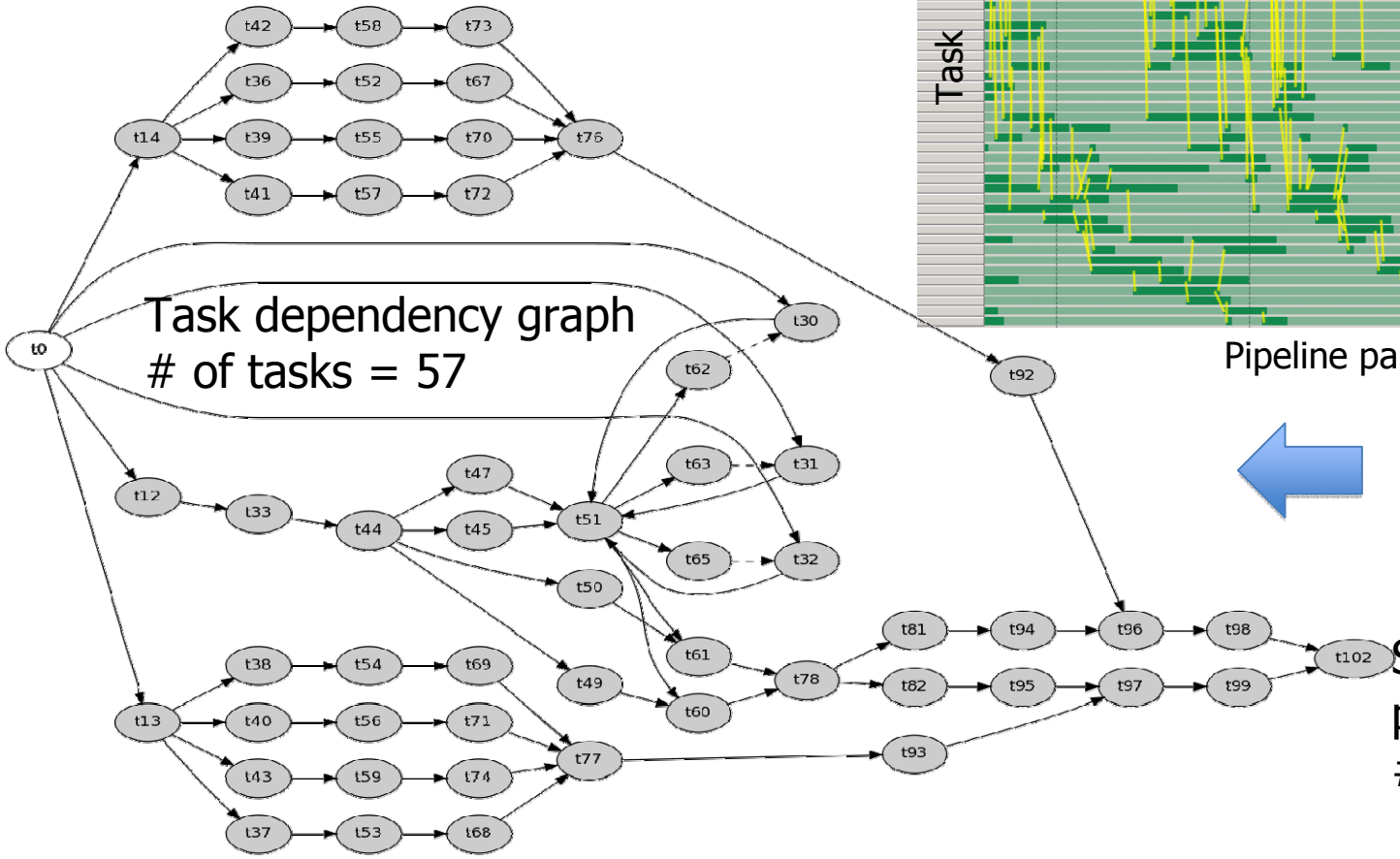
1-2. How to Introduce Parallelization from Simulink Models

Sound effect process:
Modification of frequency and amplitude、
1024sample/frame/channel



Pipeline parallelization

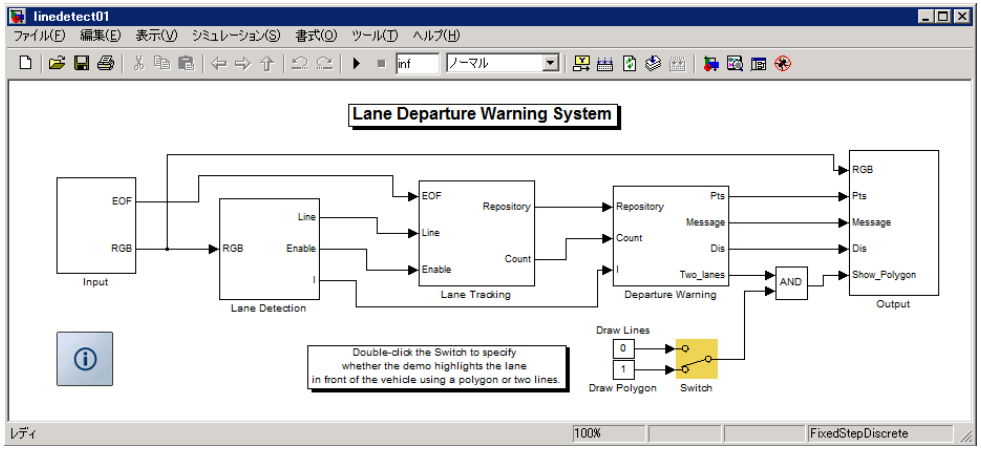




Sound effect processing model
# of blocks = 252

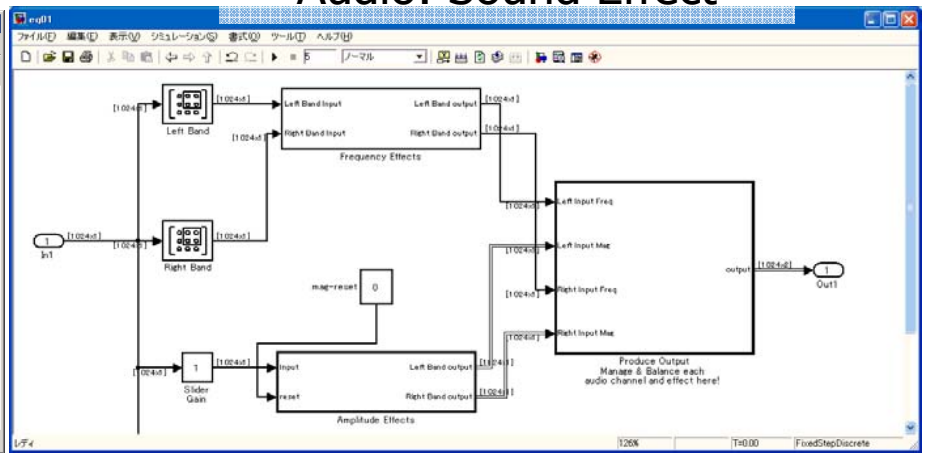
1-3. Case Studies Parallelization from Simulink Model

Lane departure warning



MATLAB :Video and Image processing toolbox : vipldw_all.mdl

Audio: Sound Effect



http://www.mathworks.com/matlabcentral/fileexchange/18317-professional-simulink-audio-equalizer

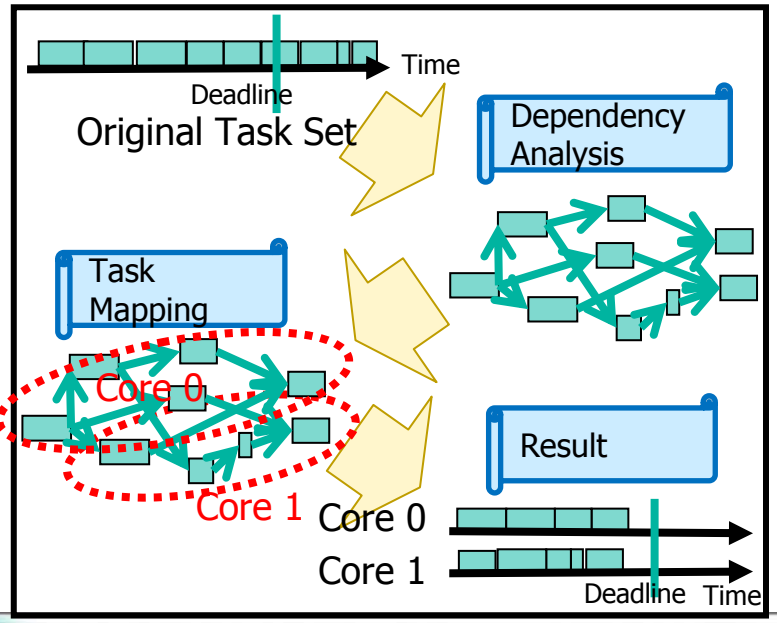
Model	# of blocks	# of tasks	Execution time compared with sequential implementation	
			Windows Xeon 4core @1.8GHz	eSOL eT-Kernel NaviEngine (4core @400MHz)
Audio	252	57	38%	26%
Lane	302	63	35%	39%

2-1.Multi-Core Task Mapping for Hard Real-Time Systems

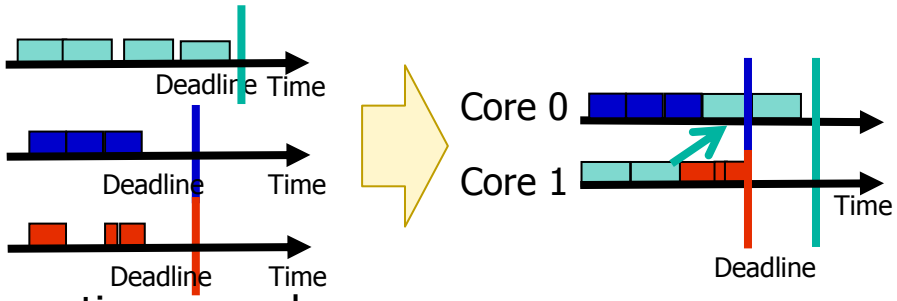
Multi-Core Task Mapping Tool: STF (Smart Task Fitter)
- ✓ Generates static mapping of embedded software on multi-core CPUs
  - ➢ Satisfying deadlines, execution order and other real-time restrictions
- ✓ Can generate mapping of hundreds of tasks within few seconds

UseCase1: Easy migration of multi task systems from single-core to multi-core

- STF generates task mapping automatically that satisfies execution order and real-time restrictions.



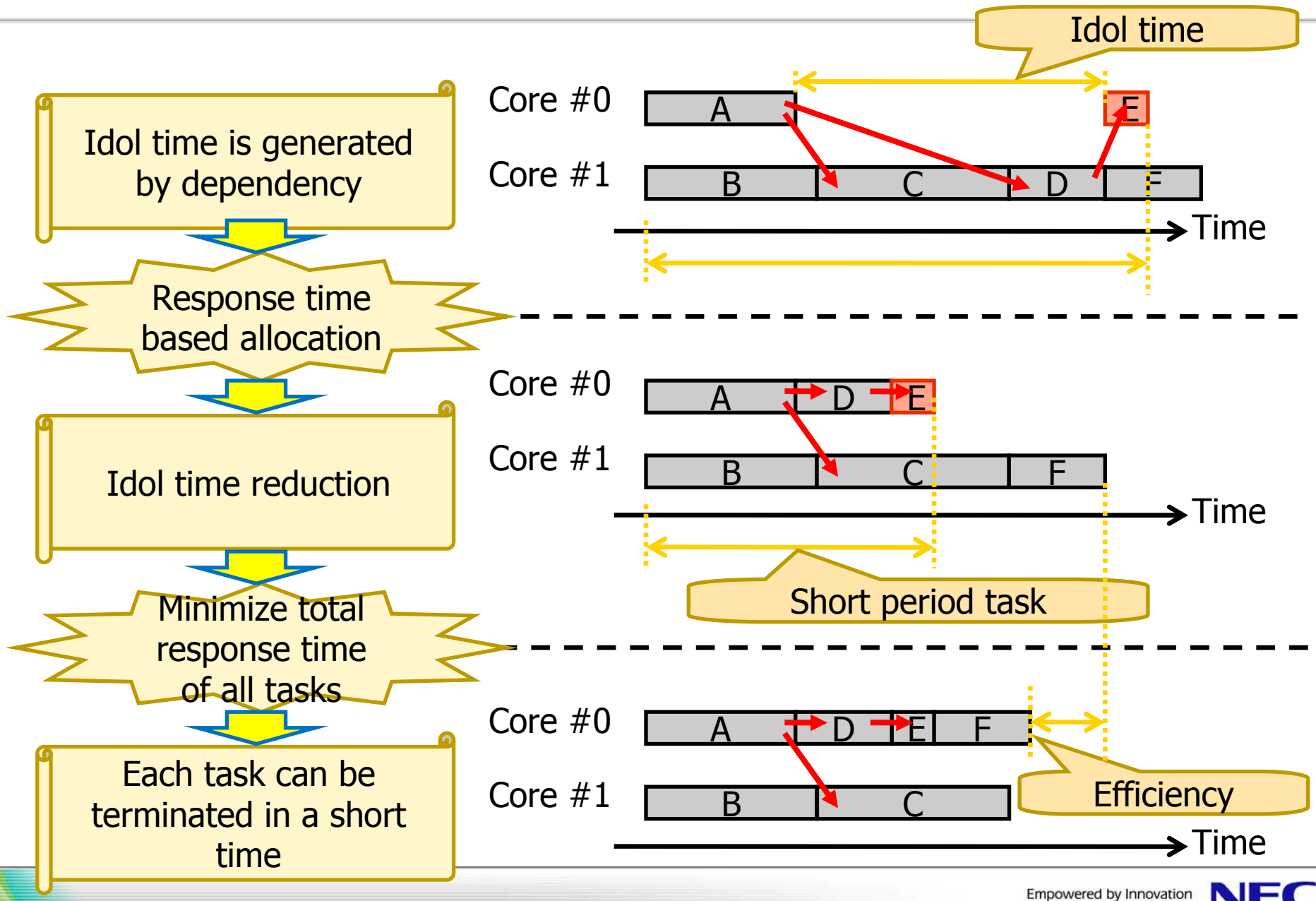
UseCase2: Integration of discrete systems onto multi-core



Integration example:
Three discrete real-time systems are integrated onto a dual-core CPU by using automatically mapping function of STF.
Each task keeps execution order and deadline.

Mapping Algorithm:
1) Task allocation with minimum response time
2) Reallocation by min-cut based placement

2-2. Flow of Task Allocation with Minimum Response Time



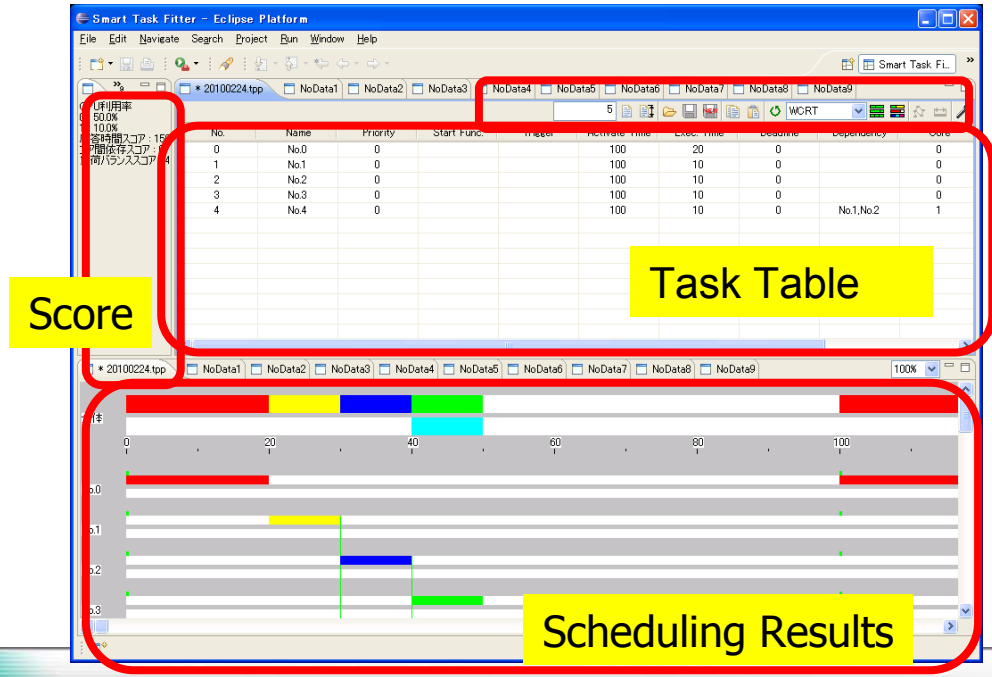
2-3. Min-cut based Re-allocation

Minimize total dependency

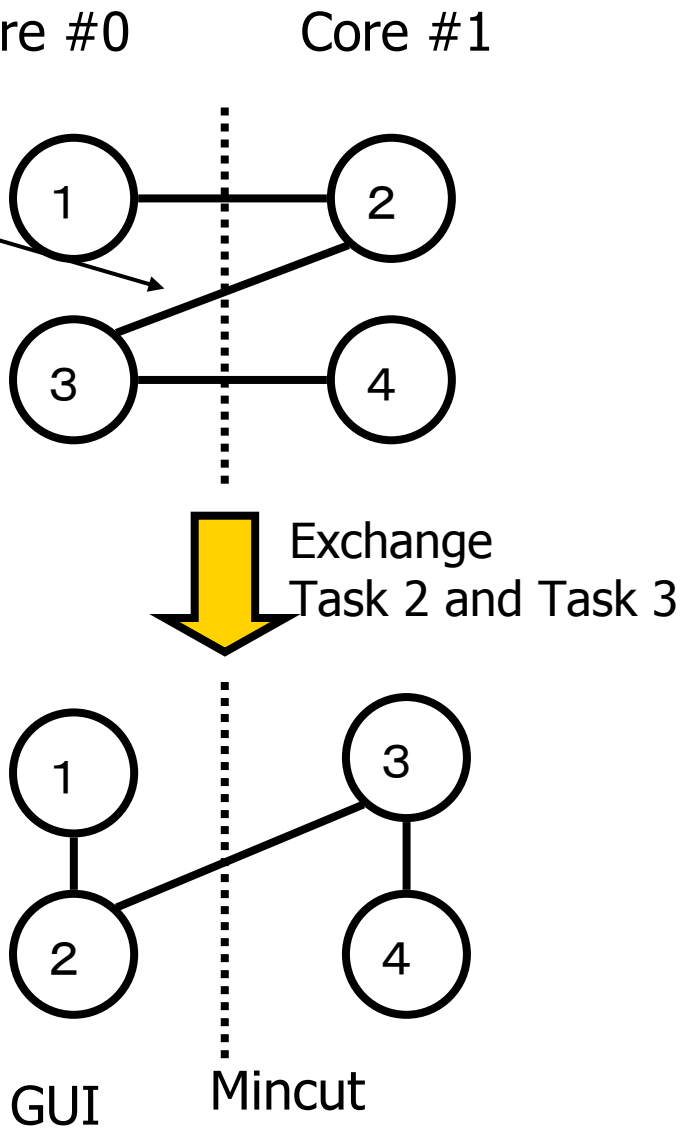
Cut = Dependency

Target: Minimize the dependency
= Minimum cut of graph

Introducing LSI placement method



Tool GUI



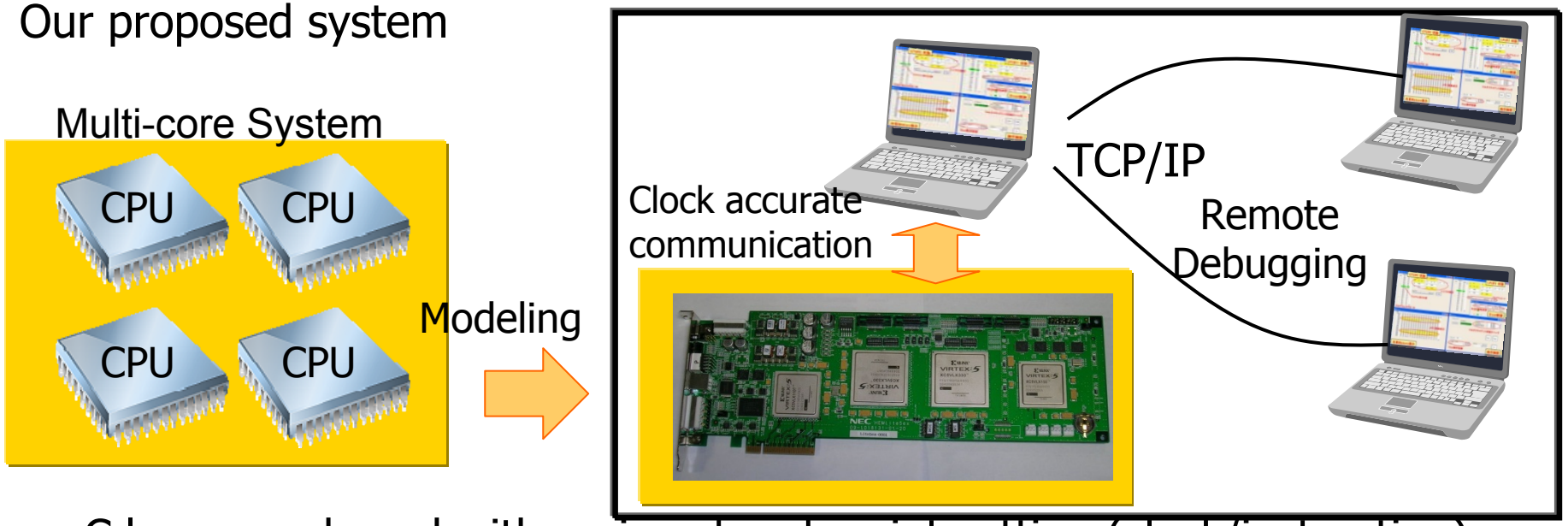
Mincut

3-1. Software debugging environment by FPGA emulator

Conventional software development environments

1. Instruction set simulator
   - Advantage: Rich observability and controllability
   - Disadvantage: Slow and less accuracy
2. Real Chip
   - Advantage: Fast and accurate
   - Disadvantage: Less observability and controllability

Our proposed system

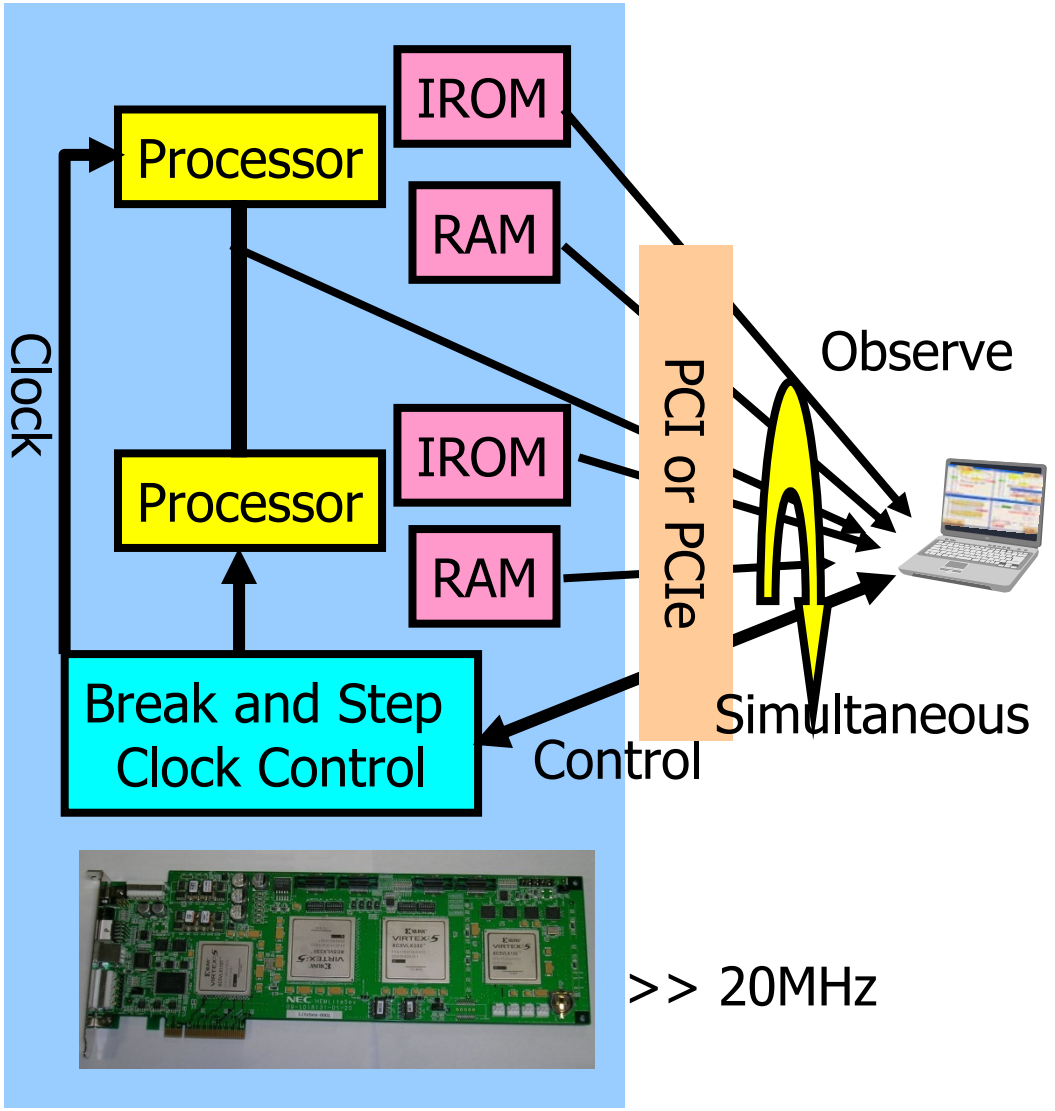


C language based with various break point setting (clock/instruction), rich observability and rich controllability

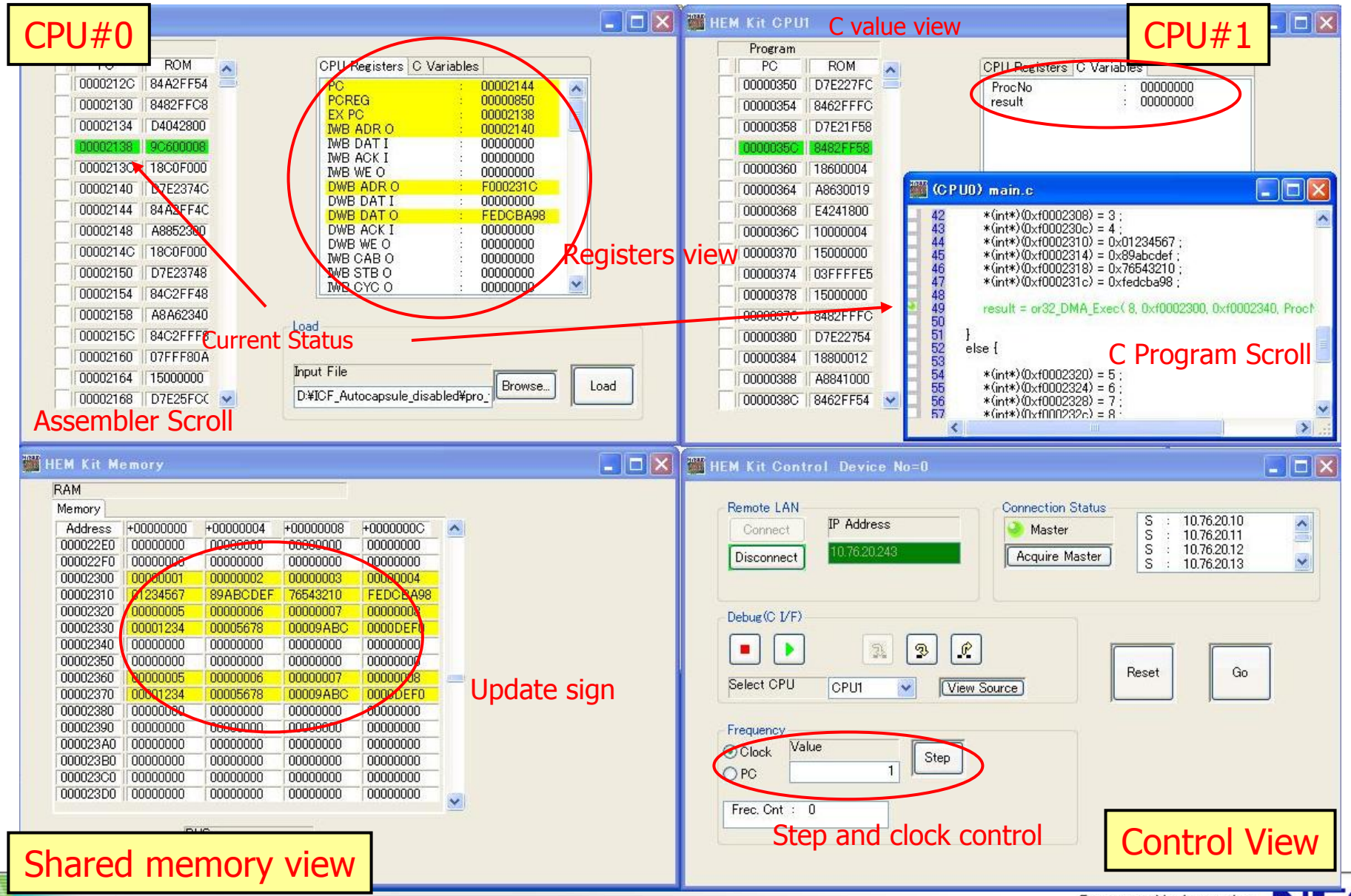
3-2. Clock Accurate FPGA Emulator Debugging System

- FPGA:
  Processors models,
  bus, IROM, RAM, break and step control
- PC:
  RAM view, ROM view, program scroll, bus monitor, and control terminal
- Debugging fabric:
  Instruction step, clock step, break point setting on register, memory view and C code

This system can handle clock and instruction level break setting.



3-3. Case Study: Environment for Multi-core System with Shared Memory



Demo Videos

Parallelization from Simulink Models

- Lane departure warning
- Compare with before parallelization and after parallelization

Software debugging environment by FPGA emulator

- Dual core(2 OpenRISC Processor) model
  - Each processor has local memory
  - Shared memory
- C language interface
- Break points and step running
- DMA from memory for CPU1 to memory for CPU2

Conclusion

- Complicated software development for embedded multi-core systems
  - Proposed 3 method
    1. Parallel C Code Generation from Simulink Models
    2. Multi-Core Task Mapping for Hard Real-Time Systems
    3. Software development environment by FPGA emulator
- Case Studies indicates the effectiveness of the proposed tools
  - They help efficient development for software design for multi-core systems
- Next target
  - Analysis the performance and quality of software on multi-core processor