

Porting Linux to the M32R Processor

Hirokazu Takata

Renesas Technology Corp., System Core Technology Div.

takata.hirokazu@renesas.com

Renesas = Hitachi + Mitsubishi

- Renesas Technology Corporation
 - ~ *Renaissance Semiconductor*
for Advanced Solutions ~
 - New joint company established by
Hitachi and *Mitsubishi* (April 2003)
 - World's Largest Microcontroller Company
 - 32-bit RISC Microcomputer
 - SuperH Family ... for processor application
 - M32R Family ... for controller application



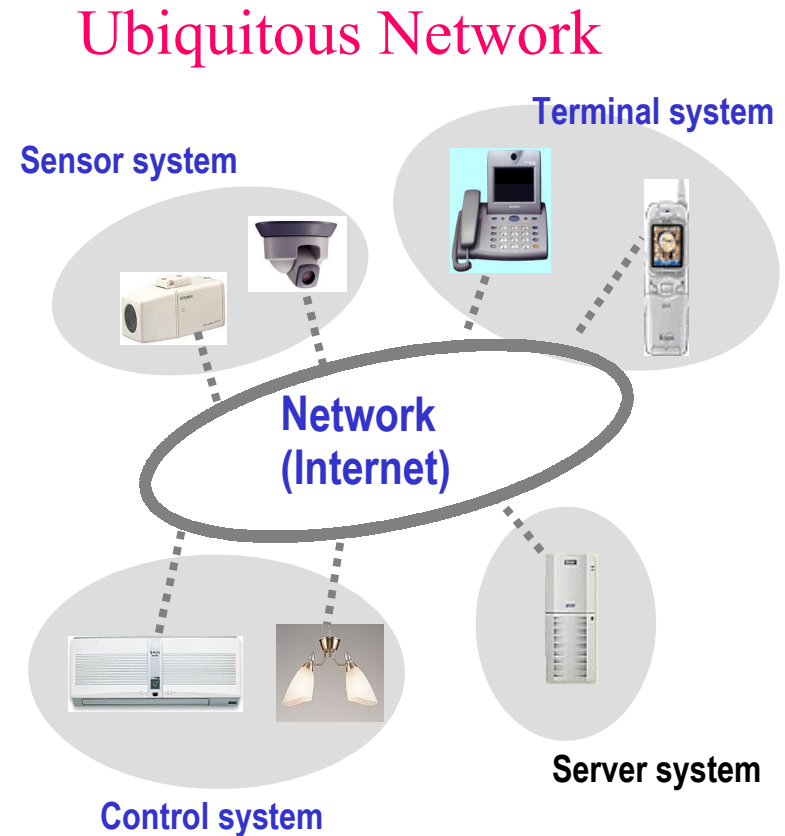
Outline

- Introduction
 - Why Linux on M32R?
 - Object: to provide “M32R Linux platform”
- Target hardware environment
 - M32R softmacro core, FPGA evaluation board
- Development of Linux/M32R
 - Porting of the Linux kernel
 - Development of GNU tools and libraries
 - Preparation of software packages (**deb** packages)
- Summary
- Demonstration



Introduction

- Background
 - Progress of system LSI technology
⇒ System-on-a-Chip (SoC)
 - Embedded systems will be connected each other.
 - Embedded systems will be more functional.
 - It is required to develop software efficiently on a de facto standard environment (Linux etc.).
- Objects
 - Establish GNU/Linux environment for the M32R
 - Prototyping of a new M32R processor
 - support SMP
 - with MMU



M32R Linux Platform

- M32R microprocessor
 - 32-bit RISC microprocessor for embedded systems (Renesas original architecture)
- Linux/M32R Project (2000~)
 - GNU/Linux Environment for M32R
 - Development of Linux/M32R (A new architecture port to the M32R)
 - Development of target hardware environment:
 - New M32R cores (with MMU) and evaluation boards
 - Porting Linux kernel
 - Development of GNU toolchains (GCC, Binutils)
 - Porting GNU C libraries
 - Preparation of self tools and root filesystem
 - :



Linux/M32R Current Status

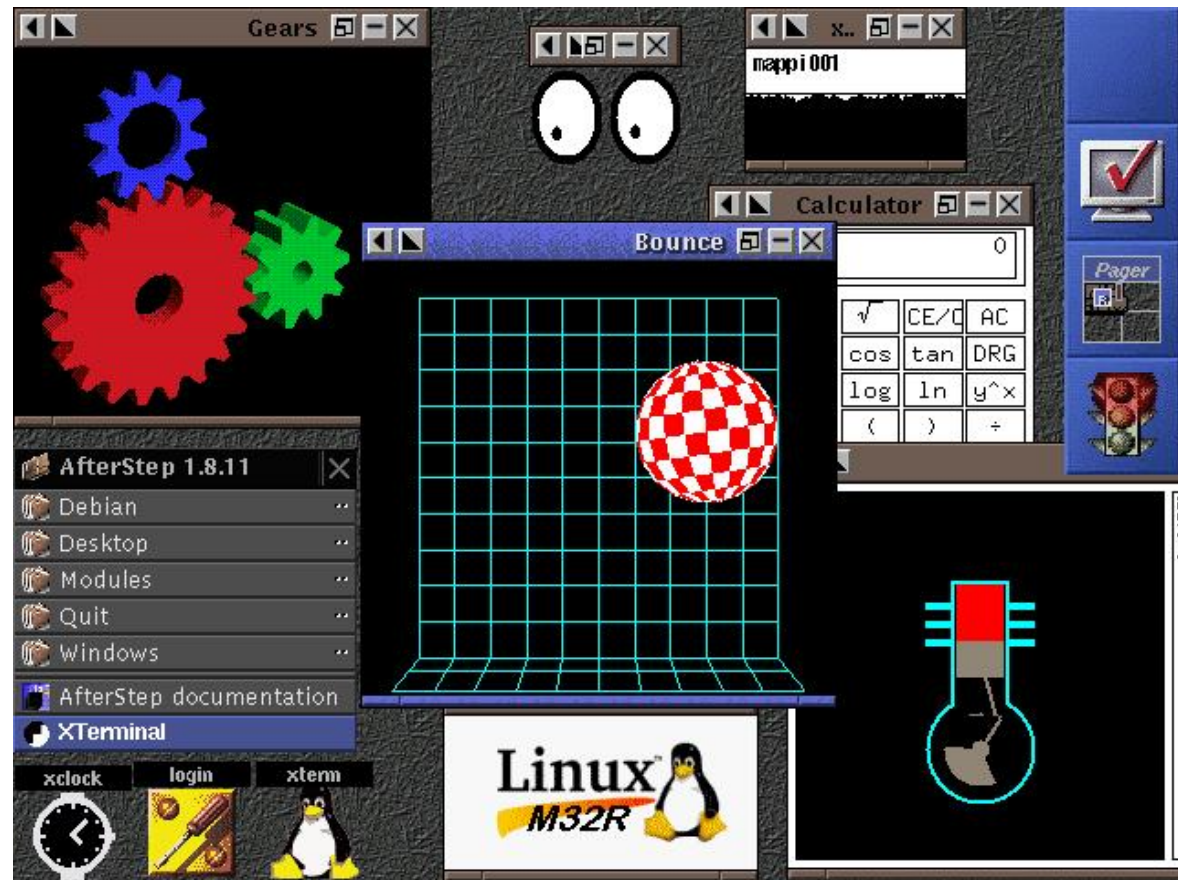
- Linux kernel
 - linux-2.4.19
 - with SMP support
- Device drivers
 - Serial driver
 - Ethernet LAN driver
 - Frame buffer device
 - PC/CF card
 - Wireless LAN,
Compact flash, etc.
- User land
 - Root filesystem: Based on the Debian GNU/Linux
 - Standalone/NFSRoot environment
 - Self tools (GCC, Binutils, etc.)
 - glibc-2.2.5
 - LinuxThreds library (Pthreads)



Linux/M32R Current Status

- GUI Environment
 - Window Systems
 - X
 - Qt-Embedded
 - MicroWindows

Snapshot of the X desktop image ⇒



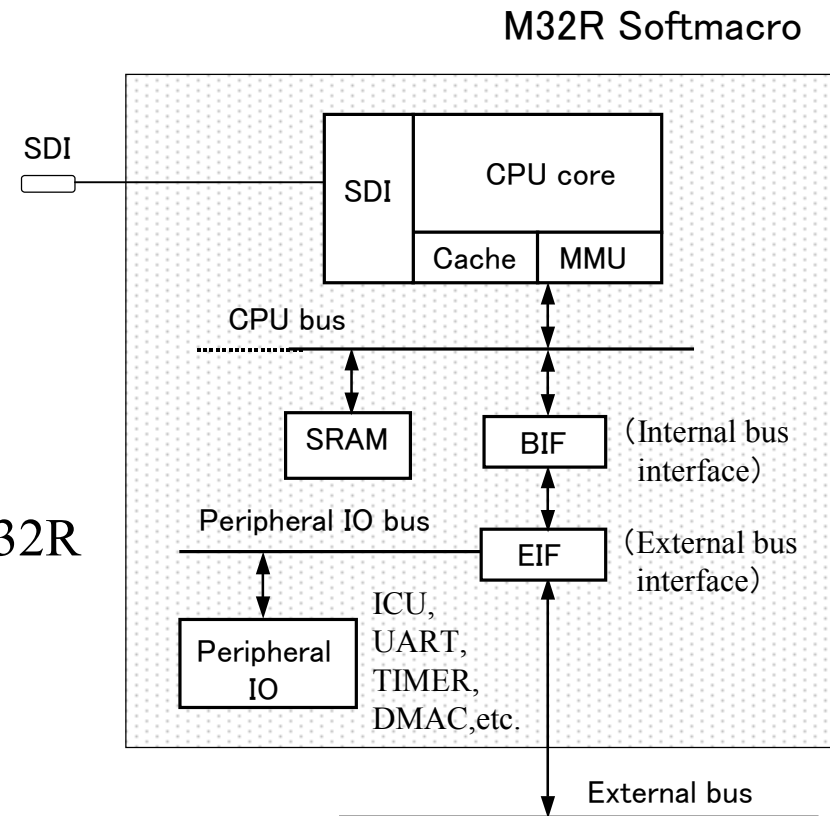
Target Hardware Environment

- Develop the target hardware environment of the M32R Linux Platform
 - M32R softmacro core
 - FPGA Evaluation board “Mappi”



M32R Softmacro Core

- Softmacro Core
(Full Synthesizable Core)
 - Not dependent on process technologies
 - Can be mapped to an FPGA
→ Easy revise and update
- M32R-II Core
 - Upper compatible ISA to the M32R
 - 5-stage pipeline, dual-issue
 - out-of-order completion
 - MMU support
 - Compact size



M32R Registers (M32R-II)

General Purpose Registers

0	31
R0	
R1	
R2	
R3	
R4	
R5	
R6	
R7	
R8	
R9	
R10	
R11	
R12	
R13	
R14 (Link Register; LR)	
R15 (Stack Pointer; SP)	

Control Registers

0	31
CR0 (PSW)	
CR1 (CBR)	
CR2 (SPI)	
CR3 (SPU)	
CR5 (EVB)	
CR6 (BPC)	

Processor Status Word
Condition Bit Register
Interrupt Stack Pointer
User Stack Pointer
EIT Vector Base Register
Backup PC

Accumulators

0	8	63
	A0	
	A1	

Program Counter

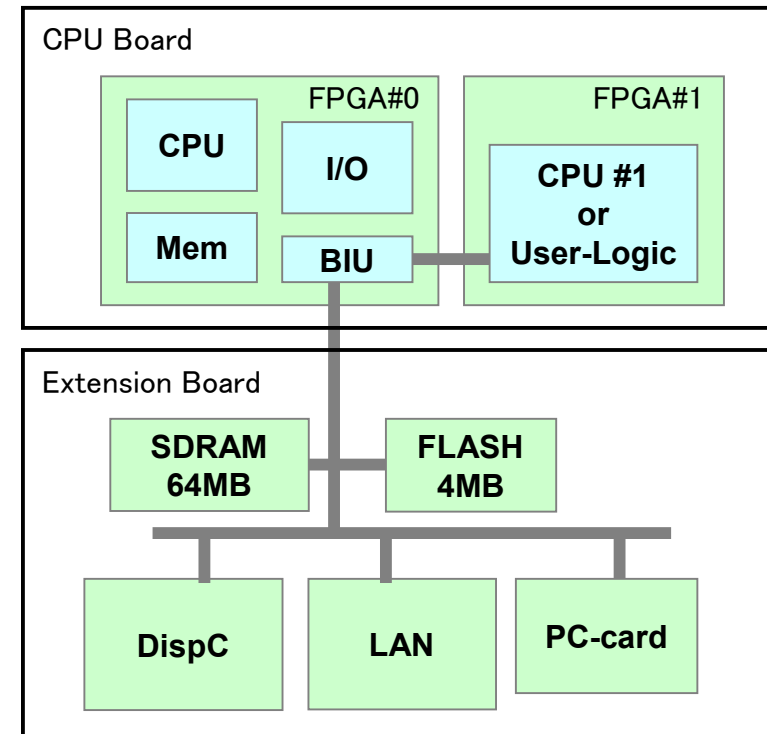
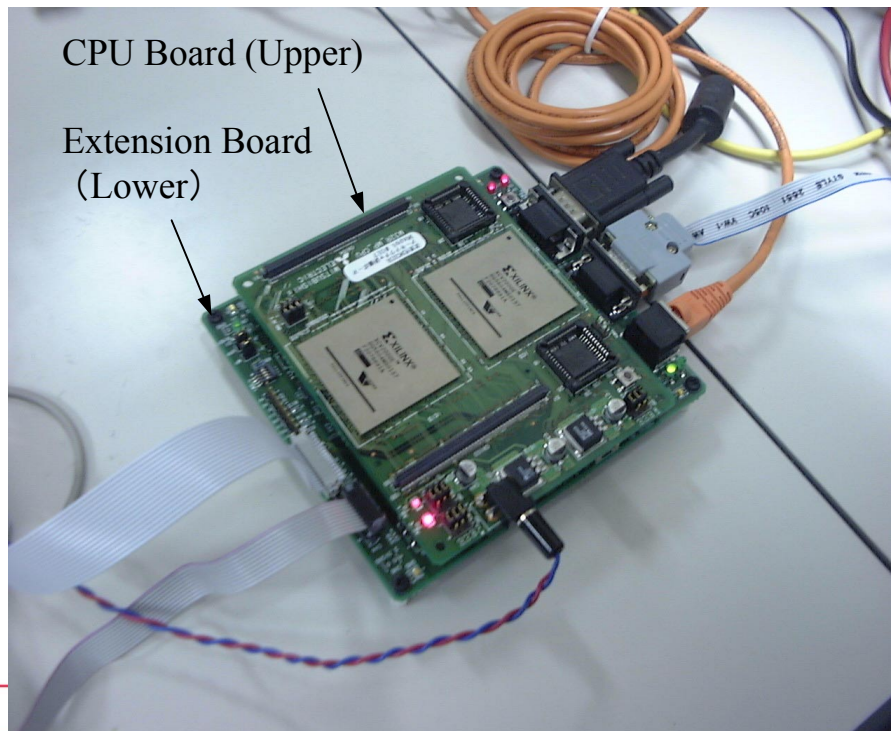
0	31
PC	



Evaluation Board “Mappi”

- Simple board ... minimum hardware for Linux
- 2 FPGAs on the CPU board;
 - An M32R softmacro core can be mapped into one FPGA.
 - Dual processor system can be achieved.

Block Diagram



Development of Linux/M32R

- **Port the Linux kernel**
 - **Porting Linux kernel to the M32R processor**
 - **Development of SMP kernel**
- Enhance GNU tools (GCC, Binutils)
- Port GNU libraries (glibc, etc.)
- Prepare debug environment
- Build software packages



Porting of the Linux Kernel

- Architecture dependent portions
 - include/asm-m32r/, arch/m32r/
- M32R specific implementations
 - Asm function routines
 - System call interface
 - Memory management routines
 - Based on the M32R's MMU/Cache specification
 - Split MMU exception handlers to lighten the TLB miss operation.



Porting the Linux Kernel (Cont.)

- Linux kernel for the M32R
 - Started to port linux-2.2 kernel (v2.2.16~)
 - After that, upgraded to linux-2.4 kernel (the latest ver. is v2.4.19)
- Development process
 - How we developed Linux kernel for the M32R ...
 - Started porting from the scheduler (by utilizing GNU simulator)
 - It is difficult to complete compilation, if header files are not complete.
⇒ Having made stub routines, we developed the kernel gradually.
 - What were problems?
 - In Linux kernel, GCC enhancement features are heavily used (inline functions, asm functions)
 - Maturity of cross tools (Develop and debug tools in parallel)



System Call Interface

- System call I/F

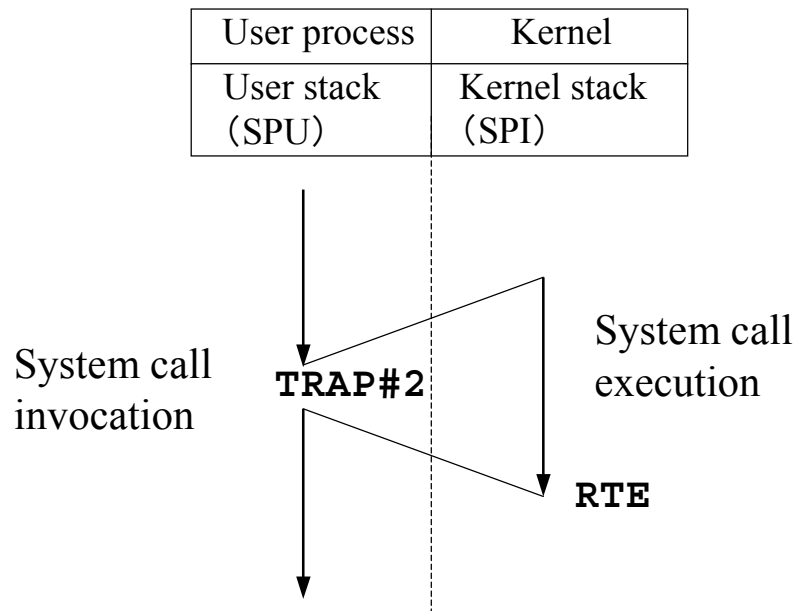
- System call : TRAP #2
 - R7: System call number
 - R0 ~ R6: arg 0 ~ arg 6 (max. 7 arguments)
- Pass pt_regs as an implicit stack parameter
- Stack is explicitly changed by CLRPSW instruction

Top of stack; SPI
(= pt_regs)

Lower address

+0x00	R4
+0x04	R5
+0x08	R6
+0x0c	*pt_regs
+0x10	R0
+0x14	R1
+0x18	R2
+0x1c	R3
+0x20	R7
+0x24	R8
+0x28	R9
+0x2c	R10
+0x30	R11
+0x34	R12
+0x38	syscall_nr
+0x3c	ACC0H
+0x40	ACC0L
+0x44	ACC1H
+0x48	ACC1L
+0x4c	PSW
+0x50	BPC
+0x54	SPU
+0x58	R13
+0x5c	LR
+0x60	SPI
+0x64	ORIG_R0

Upper address



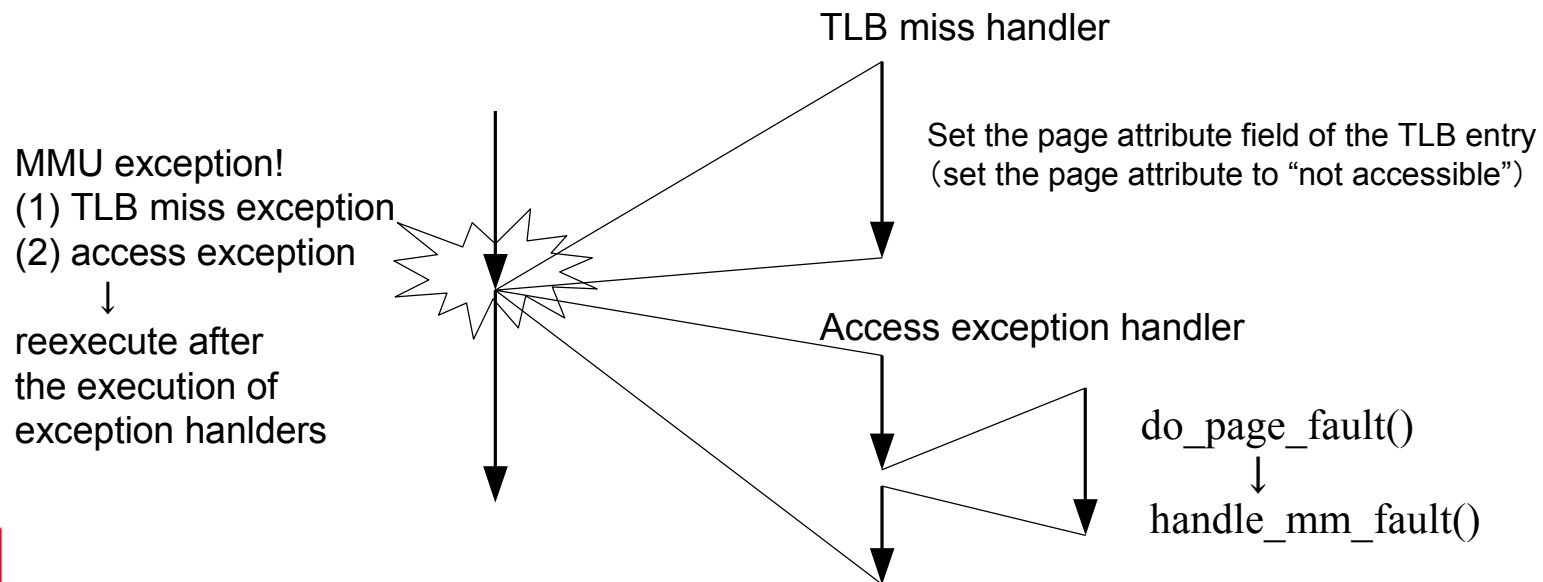
Memory Management

- Memory management of Linux (Paging)
 - Demand loading operation and Copy-On-Write operation can be executed by utilizing MMU exceptions.
⇒ MMU is necessary
- M32R-II Core
 - MMU
 - TLB entries are updated by software (cf. MIPS)
 - Number of TLB entries: Instruction/Data 16 entries (FPGA ver.)
 - 4KB/page (Linux/M32R), Large page (4MB)
 - Cache
 - Instruction/Data separated cache
 - Physically indexed physically tagged cache
⇒ Need no cache flushing operation



MMU Exception Handler

- Separate MMU exception operation into two exception handlers (because TLB misses happen more frequently than page faults)
 - TLB miss handler
 - Access exception handler
- In order to lighten the TLB miss handler ...
 - Simplify the TLB miss handler; it just sets a TLB entry
 - Write down in assembly code not to save full context

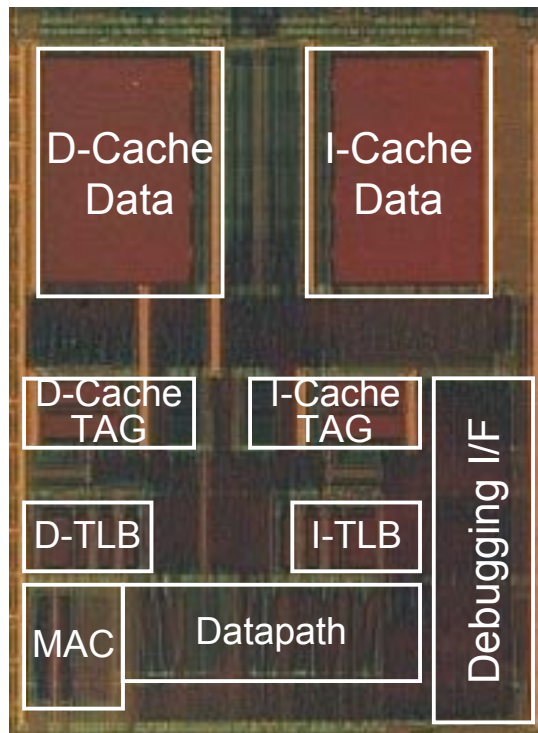


Porting the Linux kernel

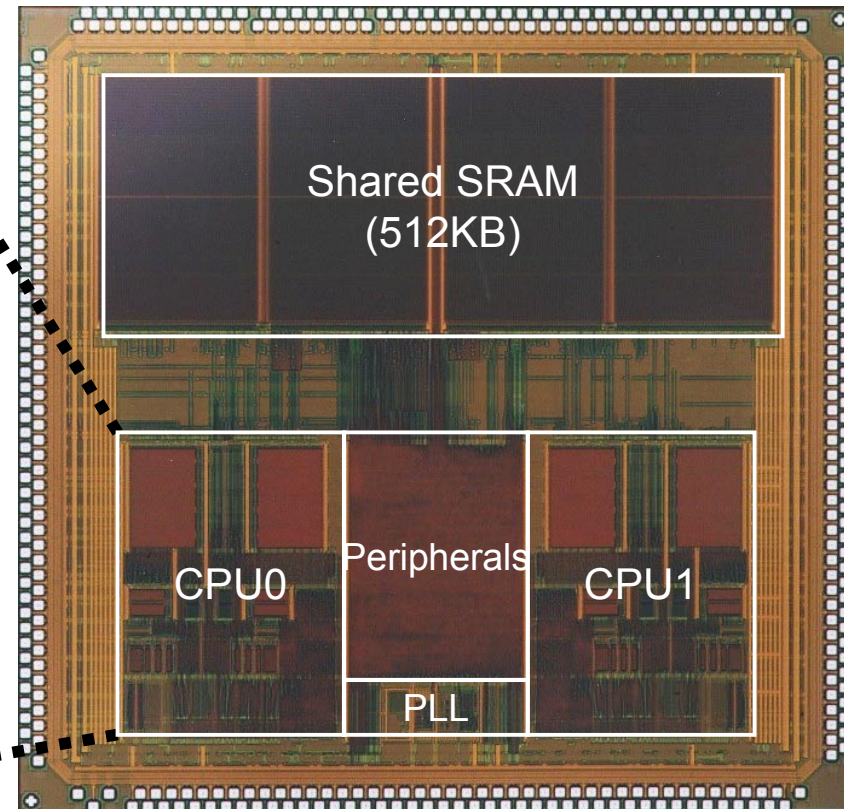
- Porting Linux kernel to the M32R
- **Development of SMP kernel**

M32R Evaluation Chip

- M32R On-Chip Multiprocessor (Ref.: Proc. of ISSCC 2003, 14.5)



M32R CPU Core



Chip Photomicrograph

Development of MP Linux System

- Development of the SMP kernel
 - Synchronization mechanism for SMP
 - Semaphore
 - Atomic access methods for variables
 - Spin lock ... LOCK/UNLOCK instructions
 - Inter-Processor Communication
 - Inter-Processor Interrupt (IPI)
 - Boot operation
- Enhance GNU C Library (for multithread programming)
 - LinuxThreads library (Pthreads; POSIX 1003.1c)
 - User-level mutual exclusion



Inter-Processor Interrupt (IPI)

- Inter-Processor Interrupt (IPI)
 - To avoid a dead lock due to IPI collisions, only one CPU can send IPI request in the M32R implementation.

- M32R's IPI spec.
 - IPI is non-maskable for the ICU
 - To mask IPI request, set IE bit (interrupt enable bit)
 - IPI requests are not queued
 - Sender CPU must confirm that the request have received by the receiver CPU.

Sender (CPU#0)


Get ipi_lock

Send IPI
IPICR0 bit#30 = 1

Confirmation
wait until
IPICR0 bit#0 = 0

Done

Receiver (CPU#1)

EIT happens

Receive IPI
read ICUISTS1

IPI operation
exec IPI#0 handler

Done

Inter-Processor Interrupt (IPI) (Cont.)

- IPI requests
 - For the Linux/M32R, the following 5 IPI factors (IPI0~IPI4) are used.

Factor	Operation
IPI0	Rescheduling request
IPI1	TLB flushing request
IPI2	Function execution request <ul style="list-style-type: none">▪ Flush whole TLB entries▪ CPU stop request▪ Request to free the slab cache
IPI3	Request for local timer operation
IPI4	CPU activation request

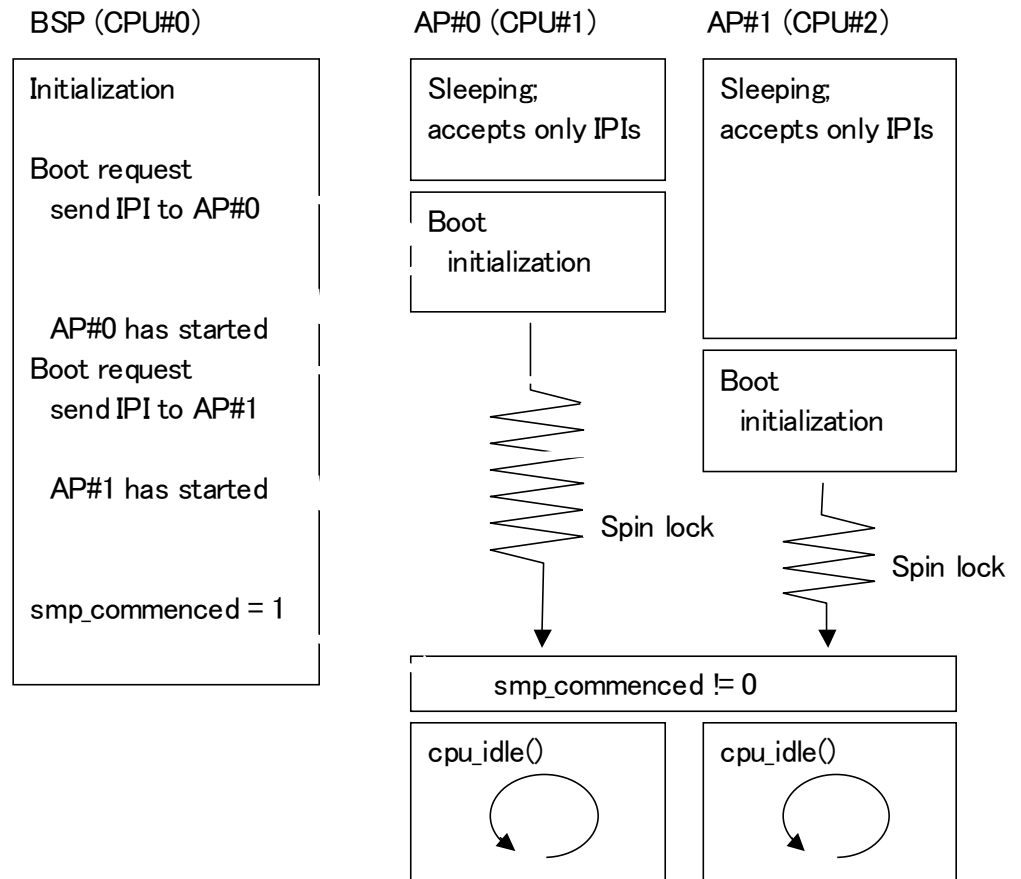
- IPI5~IPI7 are not used
- Local Timer

Currently, the local timer request is handled by IPI, because a broadcasted global timer request is not able to be accepted by all CPUs due to the ICU spec.



Boot Operation of SMP Kernel

- **BSP (Boot Strap Processor)**
 - CPU boots the system (only one)
 - BSP is selected by H/W
- **AP (Application Processor)**
 - CPUs except BSP
- **Boot sequence**
 - APs wait in sleeping at boot time (only IPIs can be accepted)
 - BSP initializes H/W and Linux, and finally boots all APs.
 - BSP sequentially activates APs, then set synchronization flags (`smp_commenced`) and makes APs into idle state (idle thread).



Development of Linux/M32R

- Port the Linux kernel
- **Enhance GNU tools (GCC, Binutils)**
 - **m32r-linux toolchain**
 - **Dynamic linking support for shared libraries**
- Port GNU libraries (glibc, etc.)
- Build software packages
- Prepare debug environment



Development of GNU Toolchain

- Enhancement of GNU tools (GCC, Binutils)
 - GCC (gcc-2.95 → gcc-3.2.3), Binutils (v2.11.92)
 - Based on the Cygnus GNUPro (m32r-elf toolchain)
 - Support ELF's dynamic linking function
 - PIC generation, shared library support
 - Enhancement of BFD library
 - No changes of the C-language's ABI (Application Binary Interface)
 - Endian support (little-endian is newly supported)
- Cross tools
 - Linux/x86 version cross tools (**m32r-linux** toolchain)
- Development of self tools
 - gcc, binutils, bash, sed, awk, perl, tcl



Dynamic Linking Support

- Dynamic Linking

- Dynamic linking/loading must be supported to utilize shared libraries.
- Programs's location (where to be loaded) is determined in runtime.



A program must be a relocatable and position independent binary.

※ PIC (Position Independent Code)

- PIC (Position Independent Code)

- Address of global symbols are dynamically stored into a GOT.
- GOT is used to resolve global symbol references.

※ GOT (Global Offset Table)

- Target of subroutine calls are also resolved by using a PLT.

※ PLT (Procedure Linkage Table)



Implement ELF Dynamic Linking

- GOT (Global Offset Table)
 - GOT is accessed via the R12 register
 - PC value is fetched by **BL**(branch&link) instruction

```
:  
; PROLOGUE  
push  r12  
push  lr  
bl     .+4  
ld24   r12, #_GLOBAL_OFFSET_TABLE_  
add     r12, lr  
:
```

- PLT (Procedure Linkage Table)
 - Symbol reference of a subroutine call is executed by indirect referencing of GOT.
(like IA-32 implementation)
- ⇒ Invalidation of instruction cache lines are not required, because the code fragment of the PLT entries are not changed.



Development of Linux/M32R

- Port the Linux kernel
- Enhance GNU tools (GCC, Binutils)
- **Port GNU libraries (glibc, etc.)**
 - **Dynamic linker for the dynamic linking**
 - **LinuxThreads library (Pthreads)**
- Prepare debug environment
- Build software packages



Porting Libraries

- Porting GNU C Library
 - glibc-2.2.3 \Rightarrow glibc-2.2.5
 - Dynamic Linker (ld-linux.so)
 - To use shared libraries (dynamically-linked libraries)
 - Implement LinuxThreads library (Pthreads)
- Development process
 1. Statically-linked “hello” binary
 - newlib version
 - glibc version
 2. Dynamically-linked binaries
 - hello.c, busybox, ...



LinuxThreads Library

- Enhance GNU C Library (to support multi-thread)
 - Multi-thread library: provides multi-thread programming env.
 - LinuxThreads library (Pthreads; POSIX 1003.1c)
 - User-level mutual exclusion has to be implemented to port
- Implement user-level mutual exclusion functions
 - On the user-level, interrupts cannot be disabled directly, and M32R's LOCK/UNLOCK instructions cannot be applied.
 - Compare with some kind of mutual exclusion support methods
 - system call implementation
 - mutual exclusion algorithm
 - Employ the Lamport's algorithm version



Development of Linux/M32R

- Port the Linux kernel
- Enhance GNU tools (GCC, Binutils)
- Port GNU libraries (glibc, etc.)
- **Prepare debug environment**
 - **Debug tools**
 - **Development environment**
- Build software packages



On-Chip Debugging Function

- SDI (Scalable Debug Interface)
 - SDI: debug interface specification commonly used by the M32R family
 - On-chip debugging function can be used through the JTAG port
 - Download target programs
 - Execute a monitor program
- Features
 - Monitor program and/or monitor ROM are not required on the target board.
 - High speed download

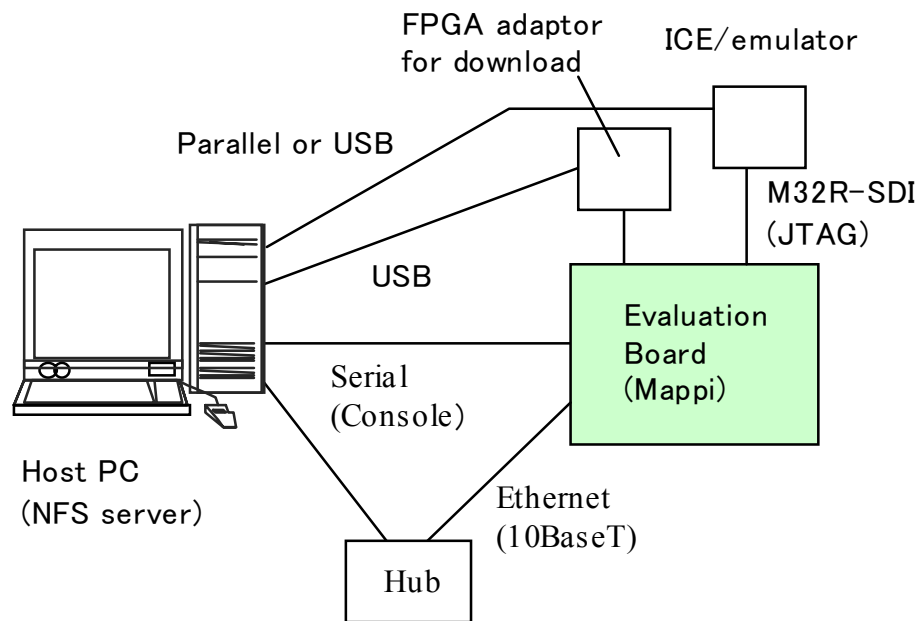


Debug Tools

- GDB with SDI support
 - Remote target: **m32rsdi**
 - Download, execution and debug by using SDI function
 - Virtual address is transformed by MMU
⇒ PC-break function is necessary.
- For the kernel debug
 - GDB (with SDI support)
 - Others: KGDB, GNU simulator (not support MMU)
- For the application debug
 - strace (trace system call invocations)
 - gdbserver (remote debugging via ethernet connection)



Development Environment



Development of Linux/M32R

- Port the Linux kernel
- Enhance GNU tools (GCC, Binutils)
- Port GNU libraries (glibc, etc.)
- Prepare debug environment
- **Build software packages**
 - Self packages for the target
 - Cross packages for the host machine

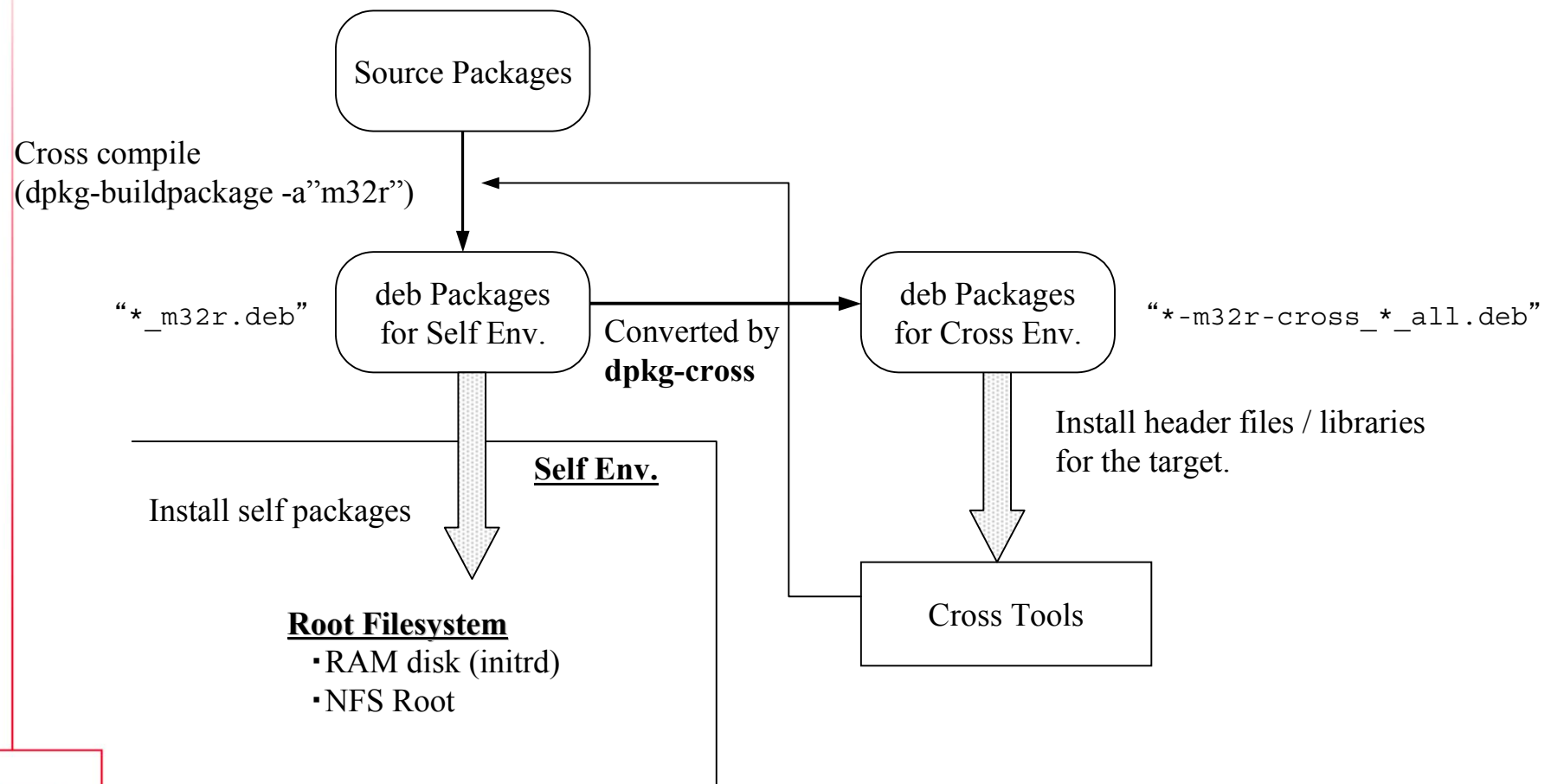


Building Software Packages

- Employ the Debian GNU/Linux as a base distribution
 - Sophisticated Package Management (→ efficient for developing)
 - With cross development support
 - **dpkg-cross**
 - **dpkg-buildpackage -a m32r -t m32r-linux**
 - .deb packages for M32R:
bash, libc6, perl, etc. ... more than 300 packages
 - Problems under cross-development
 - Header/library path is different from native environment.
 - Cannot configure/make correctly
(Perl, X server/clients, etc.)
 - Management of header files and shared libraries of target
- ⇒ Utilize both self and cross development environment



Building Packages for Cross Dev.



Evaluation

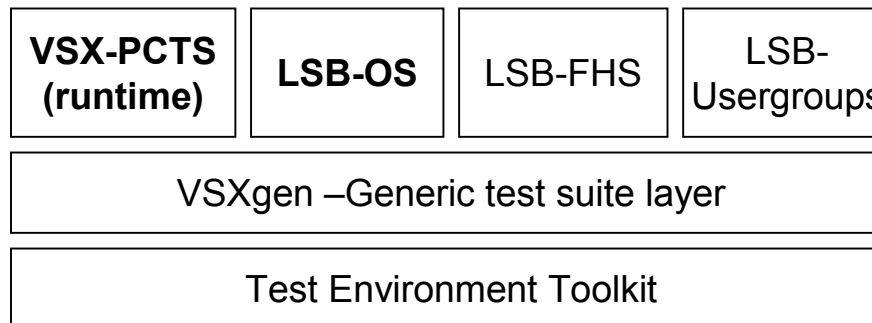
- Validation of Linux/M32R
 - LSB test suites v1.2.0



LSB Test Suites

- LSB (Linux Standard Base) Test Suite
 - Validation Test Suites for Linux
 - <http://www.linuxbase.org/test/>
- LSB Test Suite v1.2 ... LSB Specification 1.2
 - Functional validation test suites: VSX-PCTS, LSB-OS
 - Runtime Environment test suite
 - Validation of the system call and standard library APIs

LSB 1.2



LSB Test Results

- Validation Result : Good
- The result of Linux/M32R is comparable to RedHat7.3.

VSX-PCTS								LSB-OS			
Section		ANSIhdr	ANSIos		POSIXhdr	POSIXos		LSB.os		Total	RedHat7.3 Total
			F	M		F	M	F	M		
Total	Expect	386	1244	1244	394	1600	1600	908	908	8284	8284
	Actual	386	1244	1244	394	1600	1600	908	908	8284	8284
Succeed		176	1112	86	207	1333	0	695	0	3609	3583
Failed		4	0	0	5	2	0	49	0	60	45
Warnings		0	12	0	0	5	0	2	0	19	18
FIP		2	0	0	2	2	0	1	0	7	7
Unresolved		0	0	0	0	0	0	5	0	5	4
Uninitiated		0	0	0	0	0	0	0	0	0	0
Unsupported		203	0	0	179	72	0	59	0	513	513
Untested		0	4	0	0	7	0	59	0	50	43
NotInUse		1	116	1158	1	179	1600	58	908	4021	4021

Key: F: Function, M: Macro; FIP: Further Information Provided



Future Work

- Linux/M32R Platform
 - Performance evaluation, Tuning, and Stabilization
 - Continue to develop and enhance
 - Prepare development environment for middlewares, and application programs
 - Upgrade kernel version (2.5 kernel)
 - MP performance, O(1) scheduler, Preemptive kernel, ...
 - Feedback to the processor core design
- M32R GNU/Linux development environment
 - Publish the M32R GNU/Linux development environment
 - We'd like to merge source code to main stream if possible.



Summary

- Linux/M32R
 - The GNU/Linux environment for the M32R architecture
 - Linux system (UP / MP version) operates on both the M32R softmacro cores mapped on FPGAs, and an M32R single-chip multiprocessor evaluation chip.
 - Hardware/software co-design approach is employed
 - FPGA and M32R softmacro are useful for co-development or co-design of software programs and hardware IPs.
- Linux for embedded systems
 - The Open Source will provide a large impact on developing and designing of embedded systems.
 - Linux will play a great role in the field of embedded systems.



Linux/M32R Demonstration

Hirokazu Takata

Renesas Technology Corp., System Core Technology Div.

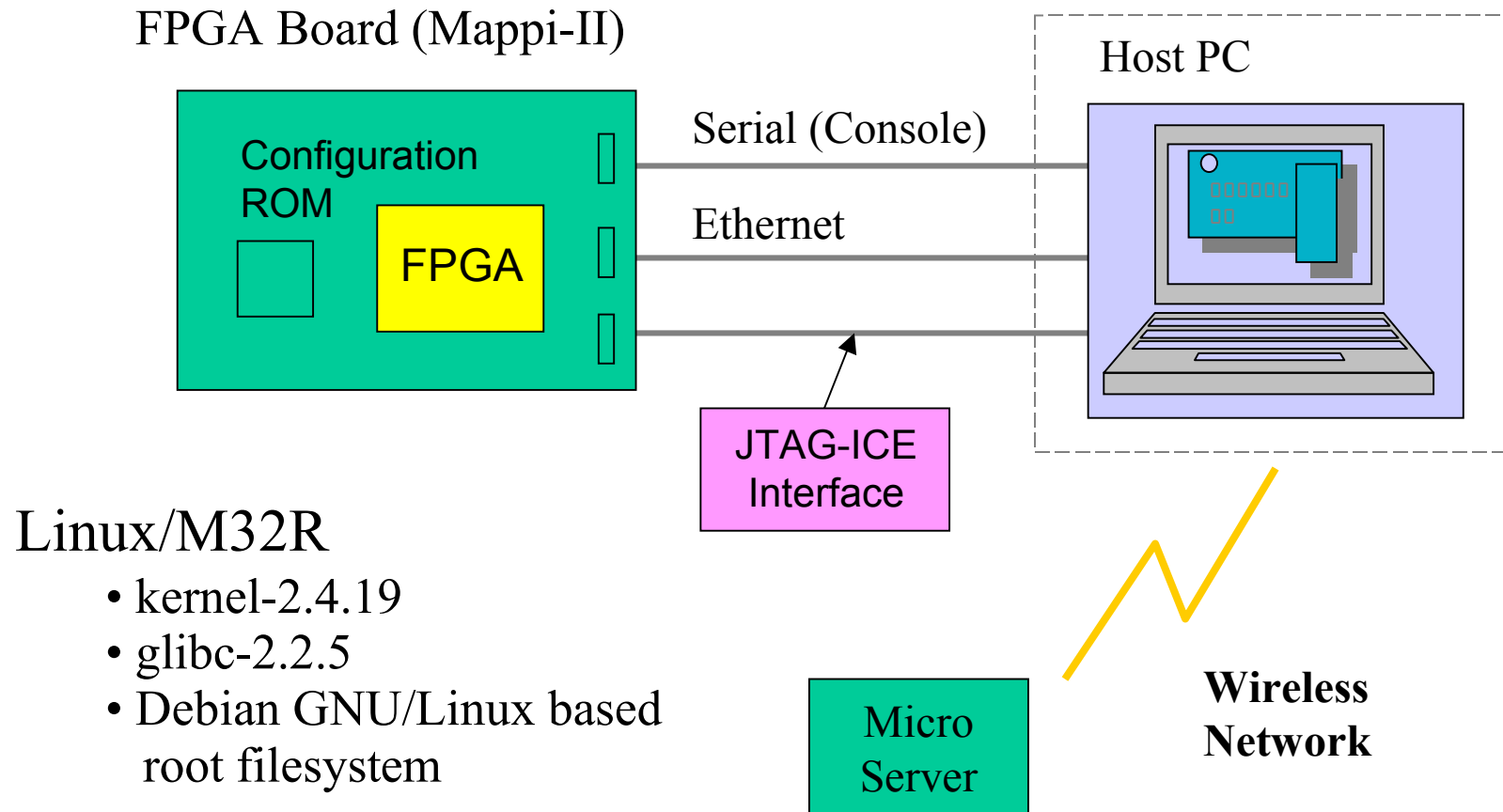
takata.hirokazu@renesas.com

Demonstration

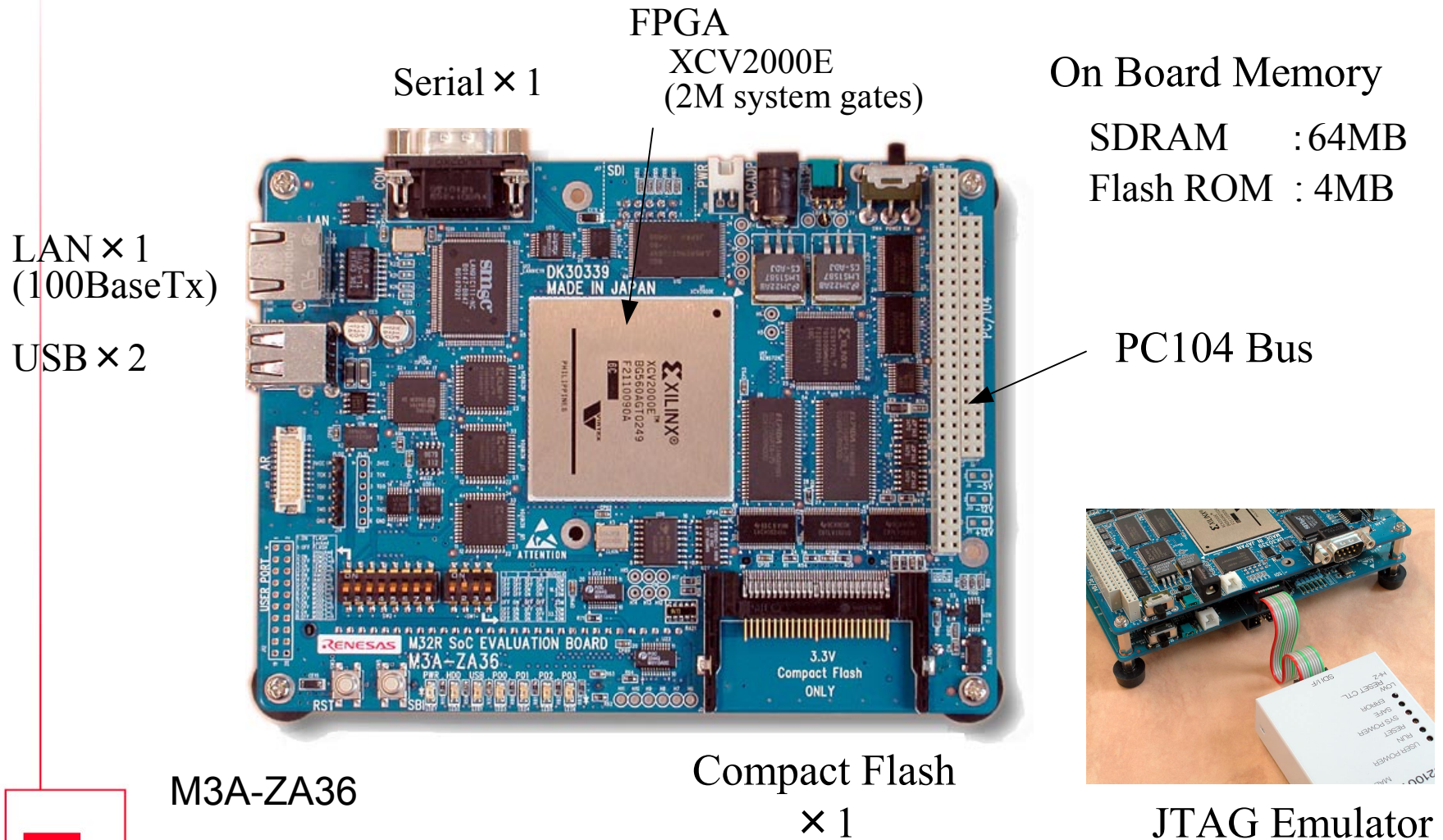
- Linux/M32R Demo. on an FPGA board “**Mappi-II**”
 - The M32R softmacro runs on an FPGA
 - NFSroot mount with LAN connection
 - GDB (GNU debugger) with JTAG connection support
 - Linux/M32R SMP Operation Demonstration
 - A tiny evaluation board “**MicroServer**” which has an M32R evaluation chip.
 - SMP kernel on the on-chip M32R multiprocessor
- ※ **MicroServer** : Developed by Mitsubishi Electric Corp.



Demonstration Environment



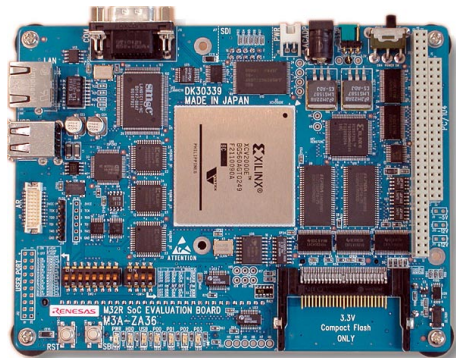
Evaluation Board “Mappi-II”



Extension Boards for “Mappi-II”

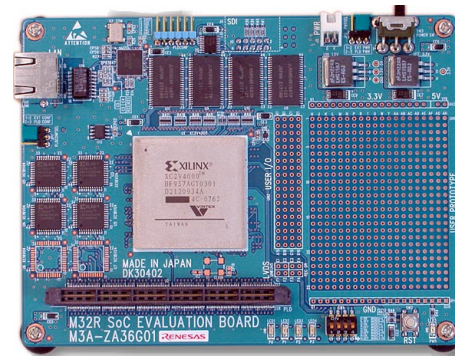
Main Board

M3A-ZA36
XCV2000E
(2M system gates)

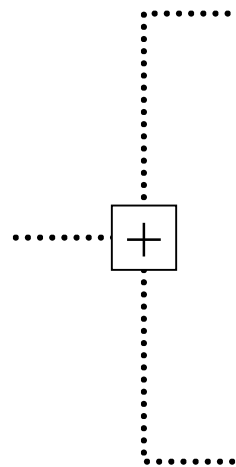
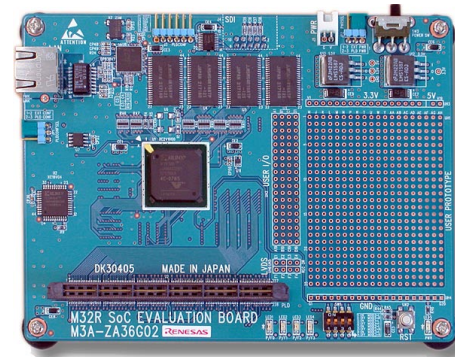


Extension FPGA Board

M3A-ZA36G01
XC2V4000
(4M system gates)

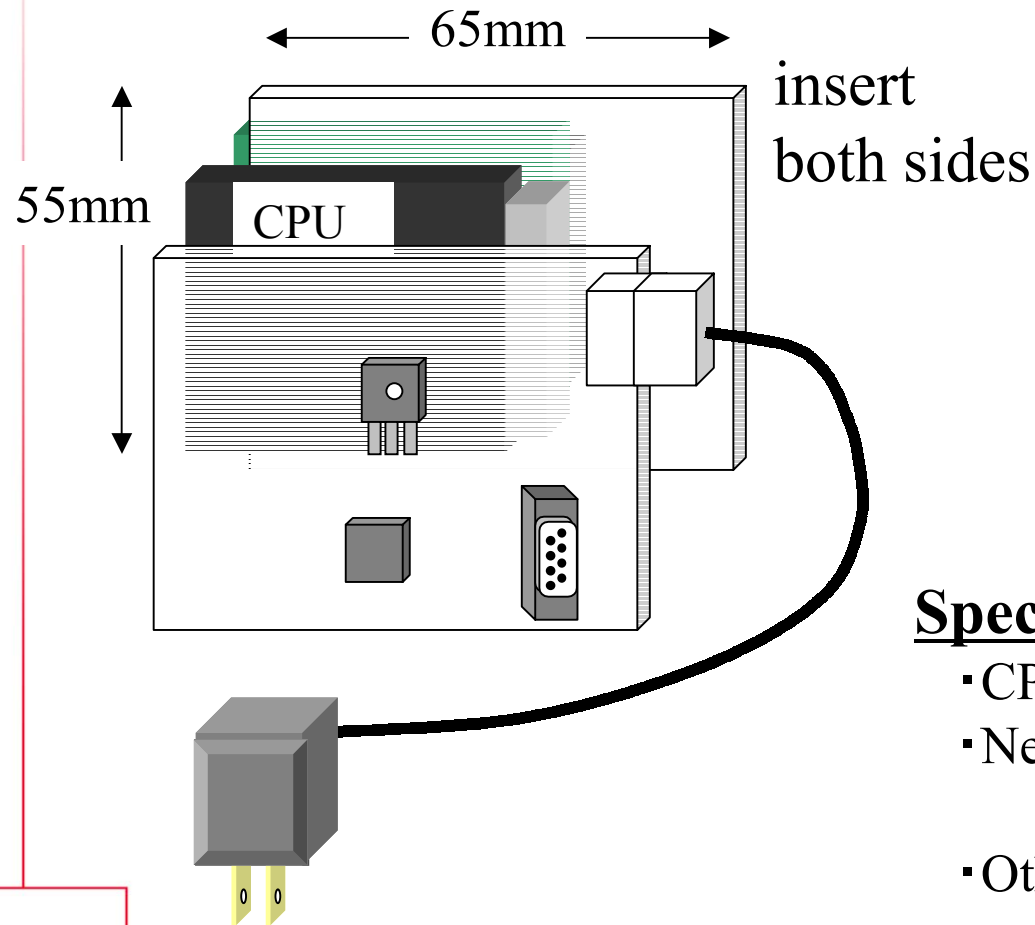


M3A-ZA36G02
XC2V1000
(1M system gates)

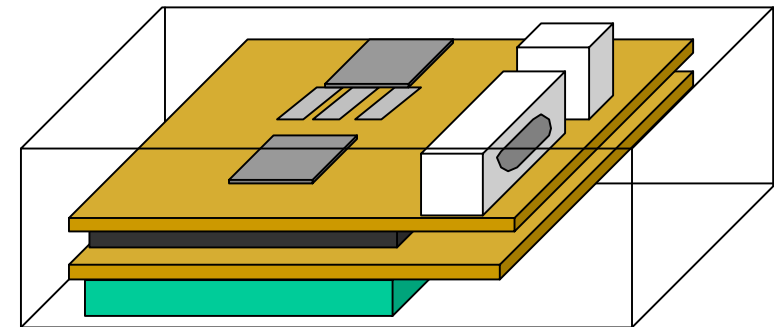


Example of Embedded Micro Server

System Image



Name card box size



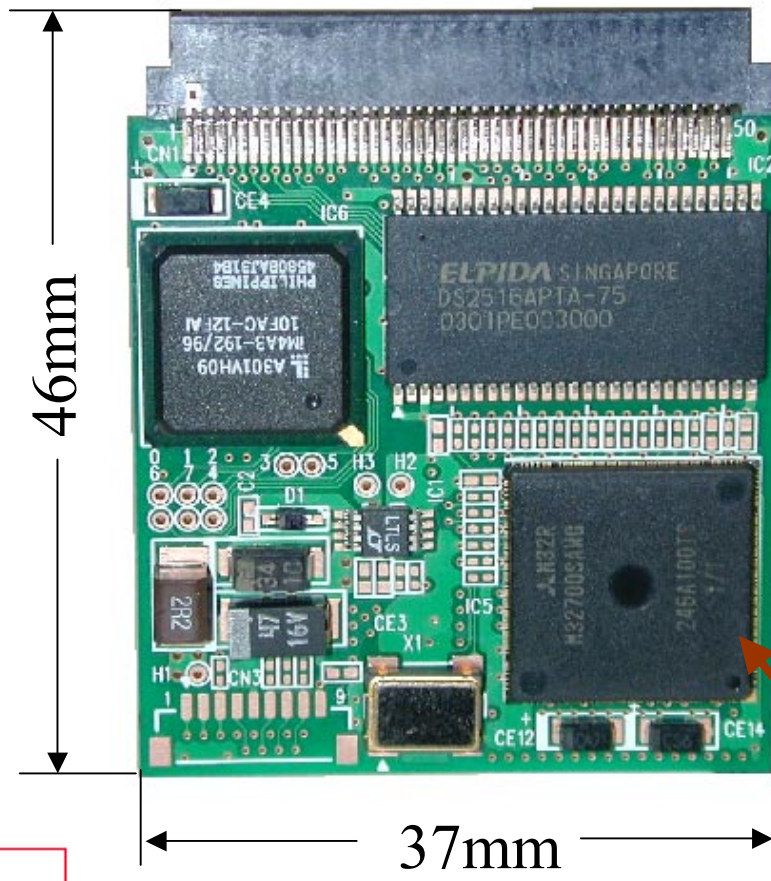
Specification

- CPU : CF size CPU module
- Network: Wired/Wireless LAN card of CF size
- Others : RS232C x1



Micro Server Module

“CF Card Size” CPU Module



- Features

CPU : M32R (Dual CPU)
OS : Linux
MW : WebServer (Boa)
SDRAM: 32MB
Flash : 8MB
I/F Con. : System, Debug,
Power Supply

- System Components

I/O : Compact Flash Card (*)
System Board, Power Supply

(*) LAN, PHS, MicroDrive, etc.
Lightweight wireless network

M32R (Evaluation Chip)



Renesas Technology Corp.