Fundamental Technology on Dependable SoC and SiP for Embedded Real-Time Systems

Nobuyuki Yamasaki (Keio Univ.)
Kikuo Wada (NECAT)
Masayuki Inaba (Tokyo Univ.)
Applications: High-end Embedded Real-Time Systems

- **Robot** (Humanoid, etc.)
- Spacecraft
- Factory automation
- Intelligent rooms/buildings
- Amusement system
- Car
- **VoD** (Video on Demand)
- Ubiquitous Computing

- Large systems in which sensors/actuators are spatially distributed
- Large scale systems that can not be controlled by a single CPU
- Fault tolerant systems, etc.
Distributed Control on Humanoid Robots

- Kojiro (Tokyo Univ.)
- Massively distributed control
  - 100-freedom, 60-controller
- HRP3L-JSK (Tokyo Univ.)
- Very high power leg
  - 48V: cont. 50[A], 15sec 100[A]

It is hard to realize a robot system by using a general purpose CPU (x86) and a general purpose OS (Windows/Linux).
Requirements in the Field of High-end Robots

Requirements for a high power motor driver

- Real-time processing under high speed communication
  - Motor temperature estimation processing for very high power motor driving such as 20 times overdrive rated at 200W
    - Current control cycle: 10msec → 10μsec

- Reliability, availability, and safety on communication and control under high-stress environment
  - Huge current noise, unusual situation such as cable disconnection, etc
  ⇒ Prevention of fatal accidents

Requirements for a large scale distributed motor driver

- Microminiaturization of the controller (size: 36x46x7mm)
  - Area constraint of the digital control part: 20mm square
  - Real-time communication and control under the size constraint
    - Poor processing power of current MPU (H8S/2215 16MHz)
      - Limit of control cycle: 1msec → 10μsec
      - External computation servers (Xeon 3.4GHz x 2) are required
    - High communication traffic 7.2MB/sec
      - Limit of Inter-device synchronization cycle: 8msec (USB) → 100μsec (Responsive Link)

- Reliability of communication under the size limitation
  - Severe noises under the logic servo systems

- Power saving scheme under the large scale distributed control
  - Static power of a whole logic part: 80W@idle → 1W
# Dependability Evaluation Indicators

<table>
<thead>
<tr>
<th>Classification</th>
<th>Evaluation items</th>
<th>Evaluation indicators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability</td>
<td>Real-time</td>
<td>Hard/soft time constraint (T/F)</td>
</tr>
<tr>
<td></td>
<td>Power</td>
<td>Time quantum (sec)</td>
</tr>
<tr>
<td></td>
<td>Noise tolerance</td>
<td>Jitter (sec)</td>
</tr>
<tr>
<td></td>
<td>Heat radiation</td>
<td>Dynamic power (W)</td>
</tr>
<tr>
<td></td>
<td>Footprint</td>
<td>Static power (W)</td>
</tr>
<tr>
<td></td>
<td>Plug-and-Play</td>
<td>S/N ratio (%)</td>
</tr>
<tr>
<td>Availability</td>
<td></td>
<td>Thermal resistance (deg C/W)</td>
</tr>
<tr>
<td></td>
<td>Network failure</td>
<td>Integration to robots (T/F)</td>
</tr>
<tr>
<td></td>
<td>Heat control</td>
<td>Correct operating (T/F)</td>
</tr>
<tr>
<td></td>
<td>Parts replacement</td>
<td>Network reconfigure time (sec)</td>
</tr>
<tr>
<td></td>
<td>Board replacement</td>
<td>Self-monitoring (T/F)</td>
</tr>
<tr>
<td></td>
<td>Failure analysis</td>
<td>Repair and replacement (sec)</td>
</tr>
<tr>
<td>Safety</td>
<td></td>
<td>Repair and replacement (sec)</td>
</tr>
<tr>
<td>Maintainability</td>
<td></td>
<td>Failure analysis (sec)</td>
</tr>
</tbody>
</table>

- **Real-time** indicators:
  - Hard/soft time constraint (T/F)
  - Time quantum (sec)
  - Jitter (sec)
  - Dynamic power (W)
  - Static power (W)
  - S/N ratio (%)
Real-Time Scheduling

- Real-time processing/communications are basically controlled by real-time schedulers.
  - EDF (Earliest Deadline First) [1]
  - RM (Rate Monotonic) [1], …

- EDF
  - Deadlines are translated to priority levels.
    - Earlier the deadline, higher the priority.
  - Priority is changed dynamically.
  - Optimal scheduling method
    - Maximum processor utilization: \( U = 1 \)

EDF Sample Schedule

Job

- J1
- J2
- J3
- J4
- J5

↑ Release time
↓ Deadline

Preemption

Earlier the deadline, higher the priority.
Distributed Real-Time Systems

- Time constraints (deadline, cycle, etc.)
- Each controller
  - Its own actuators and sensors
  - Connected via real-time network *Responsive Link*
- Almost all real-time scheduling algorithms assume:
  - Preemption
    - Context switching in case of processing
    - Packet overtaking in case of communication
  - Worst case latency
    - WCET (Worst Case Execution Time) in case of processing
    - WCRT (Worst Case Response Time) in case of communication
Multi-Level Dependability Support

- **SoC level**
  - Real-time processing/communication
  - Processing cores are connected via RT-NoC
  - Redundant processing cores and network links

- **SiP level**
  - D-RMTP I SoC and DRAM modules are integrated by FFCSP
  - Real-time DVFS w/ self-monitoring
  - Thermal control w/ self-monitoring

- **Robot level**
  - D-RMTP I SiPs are connected via *Responsive Link*
  - Adaptive ECC for *Responsive Link*
  - Network reconfiguration to avoid faulty links
  - Task migration from faulty SiPs
Real-time processing unit: **RMT PU**
- Real-time execution mechanism (RMT execution)
  - A context switch is converted to the prioritized SMT execution.
  - 8-thread simultaneous execution in order of priority
  - Thread control bases on priority (256-level)
  - Thread wake-up by an interrupt
  - IPC control (processing speed control of real-time threads): Control of WCET
- Multimedia processing units (Vector + SIMD)
  - Flexible 2D vector processing units (Integer, FP)
  - Shared vector registers by multiple threads
- Context cache (32 threads): 4-clock context switch
- Trace buffer

Real-time communication: **Responsive Link**
- Preemptive communication:
  - Packet overtaking by priority
  - Packet acceleration/deceleration: Packet priority can be replaced with new priority at each node.
- ISO/IEC 24740

Computer I/O peripherals
- PCI-X, IEEE-1394, Ethernet, etc.

Control I/O peripherals
- SpaceWire (3-ch switch)
- PWM Generators, Pulse Counters, etc.

ISO/IEC 24740

Computer I/O peripherals
- PCI-X, IEEE-1394, Ethernet, etc.

Control I/O peripherals
- SpaceWire (3-ch switch)
- PWM Generators, Pulse Counters, etc.

ISO/IEC 24740

Computer I/O peripherals
- PCI-X, IEEE-1394, Ethernet, etc.

Control I/O peripherals
- SpaceWire (3-ch switch)
- PWM Generators, Pulse Counters, etc.
Real-Time Scheduling

- A time constraint including deadline and cycle is converted to priority.
- Prioritized threads are scheduled and executed in priority order.

Thousands of clock cycles (yellow parts) are needed to switch contexts.

Elimination of the overhead the context switches

- A set of prioritized contexts is treated as a task cue of an RT-OS.
- The contexts are executed in priority order by hardware.
Multiple threads are executed simultaneously in priority order.

Implicit context switching: A context switch is converted to RMT execution (prioritized SMT execution).

Basically no software context switch exists.

High throughput real-time processing
Requirements for Real-Time Communication

- Preemption
  - Achieved by packet overtaking
  - Higher priority packets can overtake lower priority packets at each node.

- WCRT (Worst Case Response Time)
  - Network latency depends on the size of a packet and its blocked time
  - Packet level overtaking
    - Blocking time of a packet becomes constant.
Essential Requirement for Real-Time Communication

Preemption capability is required to apply real-time scheduling algorithms to communications.
A Real-Time Packet Scheduling Algorithm

Virtual Deadline Monotonic [2]

<table>
<thead>
<tr>
<th></th>
<th>Period (=Deadline)</th>
<th>Transfer time</th>
</tr>
</thead>
<tbody>
<tr>
<td>connection1</td>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>connection2, connection3</td>
<td>12</td>
<td>6</td>
</tr>
</tbody>
</table>

Real-Time Communication

Responsive Link

- Split transmission and independent routing of data and events
- Full-duplex and differential I/F
- Virtual cut-through switch with packet overtaking (preemption) function: The packet with higher priority can overtake other packets at each node.
- Priority replacement: Packet priority can be replaced with new priority at each node to accelerate/decelerate the packet.
- When the network address is same but the priority is different, the different route can be set to realize an exclusive line or a detour.
- Fixed packet size for WCRT estimation
  - Data link (64-byte), Event link (16-byte)
- Powerful adaptive error correction
  - {RS, None}, {BCH, Hamming, None}, {BS+NRZI, 8b10b, 4b10b}
- Flexible link speed (12.5 to 800 Mbps/link)
- Point-to-point link for any topology
- Standardization: ISO/IEC 24740
Split Links for Event and Data

Shared traffic
indefinite latency and throughput

Event link
Low Latency
Data link
High Throughput
Dual Physical Communication Links

Data Link: like a vessel network
Soft real-time communication for bulky data
- Multimedia data transmission (images, voice, etc.)
- Relatively large fixed packet size (64B)
- Total throughput is more important.

Event Link: like a nerve network
Hard real-time communication for control
- Control commands, Inter-processor interrupt, synchronization, etc.
- Relatively small fixed packet size (16B)
- Low latency is more important.
### Packet Format

#### Data Packet Format (64B)

<table>
<thead>
<tr>
<th>Source Addr.</th>
<th>Destination Addr.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Payload**

<table>
<thead>
<tr>
<th>Control &amp; Status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

#### Event Packet Format (16B)

<table>
<thead>
<tr>
<th>Source Addr.</th>
<th>Destination Addr.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Payload**

<table>
<thead>
<tr>
<th>Control &amp; Status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

#### Control & Status Format (32bits)

<table>
<thead>
<tr>
<th>Bit</th>
<th>UD</th>
<th>Full</th>
<th>Data Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Dirty0</td>
<td>Dirty1</td>
<td>Dirty2</td>
</tr>
<tr>
<td>2</td>
<td>Dirty8</td>
<td>Dirty9</td>
<td>Dirty10</td>
</tr>
<tr>
<td>3</td>
<td>Start</td>
<td>End</td>
<td>Int.</td>
</tr>
</tbody>
</table>

#### Frame Format (12bits)

<table>
<thead>
<tr>
<th>Data bits</th>
<th>Redundancy bits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Virtual Cut-through Switch with Packet Overtaking Function
It is possible to set a different route when the priority is different even if the network address is same. (Default route is priority 0.)

It is possible to replace a packet priority with a new priority at each node.
Routing based on Priority (1/2)

- Source
- Data (Priority0)
- Data (Priority1)
- Event (Priority2)
- Event (Priority0)
- Destination
Routing Based on Priority (2/2)
Adaptive Codecs

Error correction code

- Byte (block) error correction
  - RS (1byte error correction) (4B, 6B)
  - None
- Bit error correction
  - Hamming (1bit error correction) (8b, 12b)
  - BCH (2bit error correction) (8b, 16b)
  - None

Line code

- Bit staffing + NRZI (dynamic, clock embedded, DC balancing)
- 8b/10b (static, clock embedded, DC balancing)
- 4b/10b (static, clock embedded, DC balancing, 1bit error correction)
Measurement of Real Communication Noise

**Responsive Link + ECC**

- **Transmitter Pulse**
- **Receiver Pulse**

**D-RMTP** I 30mm sq Eva Kit

Noise generator (Amplification of real motor noise)
Performance of Noise Tolerance

**Combination of Codecs**

- **Error correction:** \{RS, None\}, \{HAM, BCH, None\}
- **Line code:** BitStaffing+NRZI, 8b10b, 4b10b

<table>
<thead>
<tr>
<th>Block level ECC</th>
<th>Bit level ECC</th>
<th>Line code</th>
<th>Codec rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BS+NRZI (9, 8)</td>
<td>BS+NRZI (9, 8)</td>
<td>29.6%</td>
</tr>
<tr>
<td>BCH (16, 8)</td>
<td>8b10b (10, 8)</td>
<td>8b10b (10, 8)</td>
<td>26.7%</td>
</tr>
<tr>
<td></td>
<td>4b10b (10, 4)</td>
<td>4b10b (10, 4)</td>
<td>13.3%</td>
</tr>
<tr>
<td>w/ Reed Solomon</td>
<td>Hamming (12, 8)</td>
<td>8b10b (10, 8)</td>
<td>35.6%</td>
</tr>
<tr>
<td>(48, 32)</td>
<td></td>
<td>4b10b (10, 4)</td>
<td>17.8%</td>
</tr>
<tr>
<td></td>
<td>ECC None</td>
<td>8b10b (10, 8)</td>
<td>53.3%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4b10b (10, 4)</td>
<td>26.7%</td>
</tr>
<tr>
<td>w/o Reed Solomon</td>
<td>BS+NRZI (9, 8)</td>
<td>BS+NRZI (9, 8)</td>
<td>44.4%</td>
</tr>
<tr>
<td>BCH (16, 8)</td>
<td>8b10b (10, 8)</td>
<td>8b10b (10, 8)</td>
<td>40.0%</td>
</tr>
<tr>
<td></td>
<td>4b10b (10, 4)</td>
<td>4b10b (10, 4)</td>
<td>20.0%</td>
</tr>
<tr>
<td>Hamming (12, 8)</td>
<td>8b10b (10, 8)</td>
<td>8b10b (10, 8)</td>
<td>53.3%</td>
</tr>
<tr>
<td></td>
<td>4b10b (10, 4)</td>
<td>4b10b (10, 4)</td>
<td>26.7%</td>
</tr>
<tr>
<td>ECC None</td>
<td>8b10b (10, 8)</td>
<td>8b10b (10, 8)</td>
<td>80.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4b10b (10, 4)</td>
<td>40.0%</td>
</tr>
</tbody>
</table>

**Effect of line code: dominant**

- **BS+NRZI**
  - BER10⁻³%: 50% error
  - BER10⁻²%: 100% error

- **8b10b**
  - BER10⁻³%: 20% error
  - BER10⁻²%: 90% error

- **4b10b**
  - BER10⁻²%: 0% error

@1-bit noise length
SiP Level Dependability

- **D-RMTP I** SoC and DRAM modules are integrated on a SiP Interposer by FFCSP
- Real-time DVFS (*D-RMTP I*)
  - Low-power while guaranteeing deadline
  - Safety voltage control w/ self-monitoring
- Prevent *D-RMTP I* & DRAM from Overheating
  - Thermal control w/ self-monitoring

**Voltage & thermal control (**\(D-RMTP I\)**)

**Vertical chip stacking (**\(D-RMTP II\)**)

Undershoot: 1.10V
Voltage transition (0.8 \(\rightarrow\) 1.1V)

0.80V
20\(\mu\)sec

Redundant vertical links
(LSI Internal Logic Level)
SoC/SiP Co-design for Improvement of Dependability

1. Optimization of IP and I/O pin arrangements for SiP design

2. Optimization of bump arrangement for SiP wiring

3. Inside-SoC adjustment scheme of wiring jitters that cannot be adjusted by SiP

Conventional design scheme

Jitter adjustment by SiP wiring pattern

Wide wiring area
Low noise tolerance

I/O buffers of SoC

Our scheme: Codesign of SoC and SiP

Jitter adjustment mechanism inside SoC

Stabilization of analog characteristic of SiP wiring pattern

Narrow wiring area
High noise tolerance

Improvement of dependability
30mm square \( D{-}\text{RMTP} \) SiP

RT-DVFS Function

- ADC for Thermal Sensor (DRAM)
- DC/DC Converter for RT-DVFS
- Flash
- D-RMTP
- FPGA
- ADC for Supply Voltage
- Potentiometer for RT-DVFS

Thermal Sensor for DRAM

Almost all functions of PC + Embedded Microcontroller + Real-Time Processing Core + Real-Time Communication

Almost all functions of PC + Embedded Microcontroller + Real-Time Processing Core + Real-Time Communication
Robot Level Dependability

- **D-RMTP I** SiPs are connected via Responsive Link
- Permanent faults (links & boards)
  - Network reconfiguration to avoid faulty links
  - Task migration from faulty **D-RMTP I** SiPs
- Transient faults (links)
  - Adaptive ECC & line codes for Responsive Link

### ECC code & Line code

<table>
<thead>
<tr>
<th>ECC code (4Byte)</th>
<th>ECC code (1Byte)</th>
<th>Line code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reed-Solomon</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(48, 32) BCH</td>
<td></td>
<td>BS+NRZI (9, 8)</td>
</tr>
<tr>
<td>(16, 8) 8b10b</td>
<td></td>
<td>(10, 8)</td>
</tr>
<tr>
<td>(10, 4) 4b10b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(12, 8) Hamming</td>
<td></td>
<td>BS+NRZI (9, 8)</td>
</tr>
<tr>
<td>(8b10b (10, 8))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(4b10b (10, 4))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(None)</td>
<td></td>
<td>8b10b (10, 8)</td>
</tr>
<tr>
<td>(4b10b (10, 4))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(None)</td>
<td></td>
<td>8b10b (10, 8)</td>
</tr>
<tr>
<td>(4b10b (10, 4))</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(1) Permanent faults by link disconnection
(2) Transient faults by motor noise
**D-RMTP On-board Motor Driver**

- **D-RMTP SiP on board**
- **Size:** 85x60x36 mm
- **Specification of motor driver**
  - Voltage: 80V
  - Current: cont. 50A, max. 200A
  - Vector control
  - Water-cooling

[Diagram of D-RMTP SiP on board]

[Image of D-RMTP SiP on board]

[Image of D-RMTP SiP on board with Water-cooling]
D-RMTP I Evaluation Kit

For more information, please contact:
Yamasaki lab., Keio Univ.
http://www.ny.ics.keio.ac.jp/
Ultra Small Evaluation Kit for Distributed Control via **Responsive Link**

**I/O Core SiP**

- **UART Connector**
- **FPGA**
- **RS485**
- **I2C Connector**
- **SPI Connector**
- **Acceleration Sensor**
- **Thermal Sensor**
- **A/D Converter**

**I/O Core SiP (Bottom)**

- **DRAM**
- **Flash ROM**
- **RS232C**

**Analog I/F Board (Center)**

- **USB Peripheral Connector**
- **JTAG Connector**
- **Flash**
- **PCI**

**USB Board (Top)**

- **USB Host Connector**
- **FPGA**
- **PCI**
Programming

**ISA of *RMT Processor***:
- MIPS upper compatible ISA + thread control instructions + vector instructions
- MIPS instruction
  - C and C++ available
- *RMT Processor* own instructions (multithread instructions, vector operations)
  - Assemble language
  - Libraries (boost+, tvnet)

**OS**
- Linux
- iTRON
- favor, Tflight (our original RT-OS)

**Cross development tools**
Simulators

- Anywhere the *D-RMT Processor* can be developed.
  - Host OS: Linux, Solaris, Cygwin
  - No real hardware is needed.
  - A laptop PC is available to develop a program.
- High speed and low functionality: ISS (Instruction Set Simulator)
  - Processor model
  - Major I/O models (Serial, *Responsive Link*)
  - No timing check
- Low speed and high functionality: RTLS (RTL Simulator)
  - All functional models of the *D-RMTP SiP*
    - *RMT Processor*
    - Responsive Link
    - PCI
    - PWM, etc.
Conclusion

- RT-Processor
- RT-Network

Hard/software co-design
Dependable SoC
Dependable SiP

SoC/SiP co-design

Real-time
Distributed control
RT-DVFS

QoS
RT-OS

High power motor driver
Noise tolerance
Heat radiation

Control board

High power leg
3-D mounted SiP
Humanoid robot