Towards a high performance parallel platform for dependable embedded systems

Mitsuhisa Sato

University of Tsukuba

JST-CREST “Dependable Operating Systems for Embedded Systems” Project

Outline

- Background
  - Trends of Microprocessors & embedded applications
- About our project
  - Concept of our project on high performance parallel platform of multi-core and multiprocessors systems for near-future dependable embedded systems
- OpenMP for Parallel embedded Systems
- Research topics in our project
  - Power-aware runtime management for OpenMP
  - Reliable DSM and check pointing
  - Reliable and high-performance communication layer using multiple link
  - High-speed and low-power interconnect by PCI-Express Gen2
- Summary

Background: Trends of Microprocessors & embedded applications

- Needs of high performance in embedded systems
  - Networking appliance, etc…
  - RMS (Recognition, Mining, Synthesize) (by P. Gelsinger@Intel)
  - High-performance and real-time processing
    - Car navigation system
    - High-level GUI in embedded system, such as 3D volume rendering
    - 3D recognition by collecting/synthesizing info from multi-cameras.
    - ….
- Multi-core, Multi-processors
  - Parallel embedded system for high performance
  - Allows flexible power and performance management by activating/inactivating each core (or DVFS)
  - Good for both high-performance and low-power!!!
  - Redundancy by multi-processors for fault-tolerance.

Power consumption of multi-core/multi-processors

$$P = N \times \alpha \times C \times V^2 \times f$$

#Core

Active Rate

Capacitance of Circuit

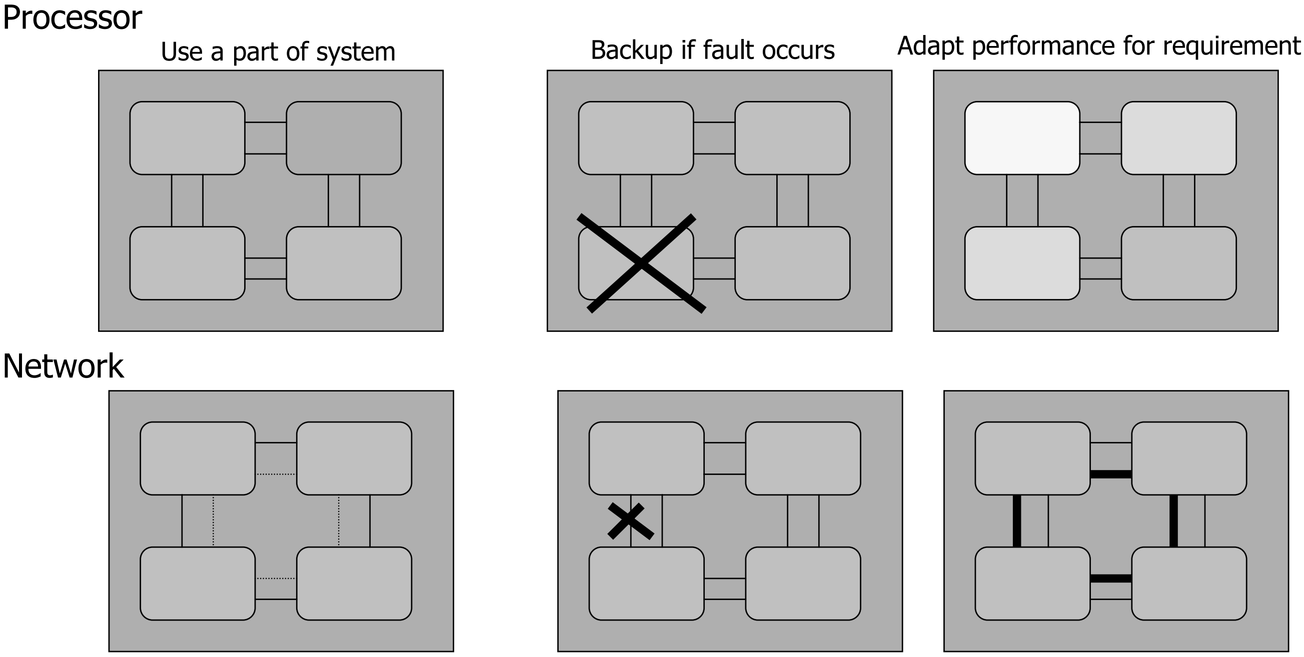
Voltage

Clock rate

Concept of our project on high performance dependable parallel embedded systems

Parallel embedded systems → Quality（real-time・high performance・low power）

Parallel embedded systems → Redundancy → Reliability (dependability)



Objective of our project

- **"Low-power and Highly Dependable Parallel Computer Platform for Embedded Systems"** **(U. of Tsukuba and Renesas)**
  - Under JST-CREST program, research area **"Dependable Operating Systems for Embedded Systems Aiming at Practical Applications"**
    - Research Supervisor: **Dr. Mario Tokoro** (SVP, Corporate Executive, Sony Corporation)
  - Project period: From Oct. 2006 to Nov. 2011 (5 years)

- Investigate dependable technologies for a high-performance parallel embedded computer platform with multi-core/multiprocessor systems.
  - Develop a programming tools and environment for embedded parallel programs, and run-time mechanism for dependability.
    - OpenMP and Reliable Software DSM & Checkpoint/Restart
  - Develop a power management run-time system to optimize performance and power consumption under real-time constraints
    - OpenMP power-aware runtime system
  - Develop communication facility and multiple network link hardware to provide fault-tolerance and power management in the communication layer of embedded parallel systems.
    - Multi-link comm. software and PCIe Gen2 network communicator

What's OpenMP

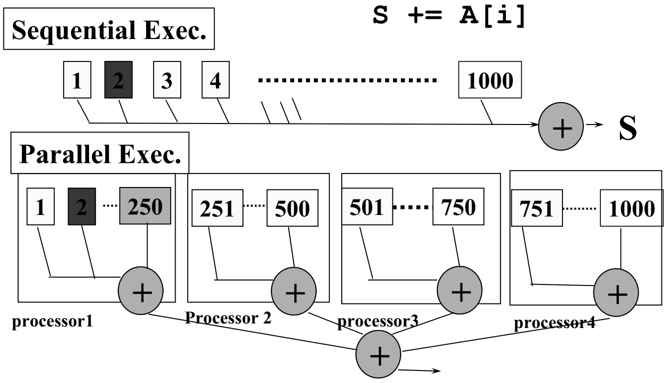
- Standard parallel programming model and API for shared memory multiprocessors
  - Extend the base language (Fortran/C/C++) with directives or pragma
  - Incremental parallel programming
  - keep sequential semantics with ignoring directives
  - allows range of programming styles
  - For scientific applications. Support for loop-based parallelism and task-parallelism
  - Target: small-scale(～16processors)to medium-scale (～64processors)
  - The last version 3.0 spec focuses on task-parallelism.

- OpenMP ARB
  - http://www.openmp.org/

- Example
  - Loop parallelized by OpenMP directive

```
#pragma omp parallel for reduction(+:s)
   for(i=0; i<1000;i++) s+= a[i];
```

```
for(i=0;i<1000; i++)
      S += A[i]
```

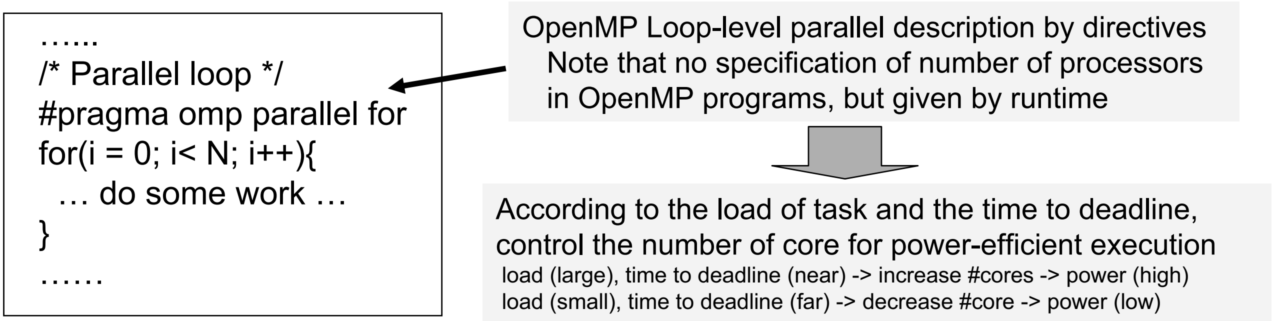


OpenMP and multi-core/multi-processors

- Multiprocessors with “simple” cores
  - Exploit thread-level coarse-grain parallelism
  - It may provide better performance than “complex” superscalar does in the same die
    - Good for applications with large amount of parallelism
    - Simpler and low-power architecture and implementation
    - low-latency communications between cores
  - OpenMP can be used for a “simple” and “easy-to-use” parallel programming environment
    - Most of Multi-core is naturally “shared memory” multiprocessors
    - Exploit thread-parallelism by programmers
- Research issues in OpenMP for multi-core
  - Current most multi-core embedded processors are not used for “parallel programming”
    - Lack of parallel programming environment!!
  - How to express the parallelism of embedded applications
    - The current OpenMP supports loop-level parallelism in scientific applications
    - Needs more task-level parallelism with constraints such as real-time task.
  - Thread scheduling for efficient execution of multi-threaded programs.
    - Co-scheduling, gang-scheduling with real-time constraints
  - Embedded multicore may not support “true” shared memory.
    - Cell BE@IBM, …

Power-aware runtime system for OpenMP

- In a parallel program, Open is usually used to exploit parallelism for high performance.
- We propose OpenMP run-time scheduling for a tradeoff between performance and power in real-time embedded applications for power-aware computing.
  - Typical requirements in real-time applications is to execute a reserved job within a certain period.
  - In terms of power efficiency, program does not necessarily execute fast as long as it can meet the deadline.
  - OpenMP power-aware runtime system adjusts the number of core to execute the program for power-aware computing in embedded systems.
- OpenMP can be used as a user-transparent programming model for power-aware computing.



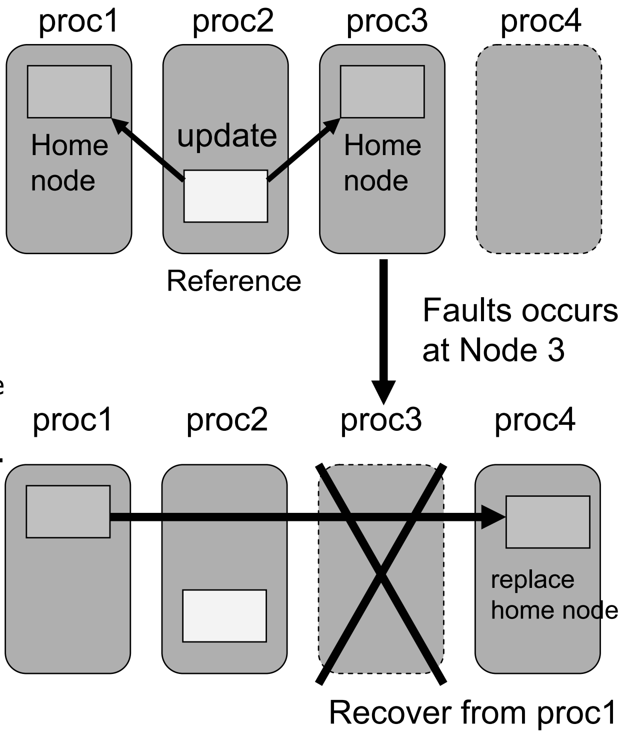
```
……..
/* Parallel loop */
#pragma omp parallel for
for(i = 0; i< N; i++){
  … do some work …
}
……
```

OpenMP Loop-level parallel description by directives
Note that no specification of number of processors in OpenMP programs, but given by runtime

According to the load of task and the time to deadline, control the number of core for power-efficient execution
load (large), time to deadline (near) -> increase #cores -> power (high)
load (small), time to deadline (far) -> decrease #core -> power (low)

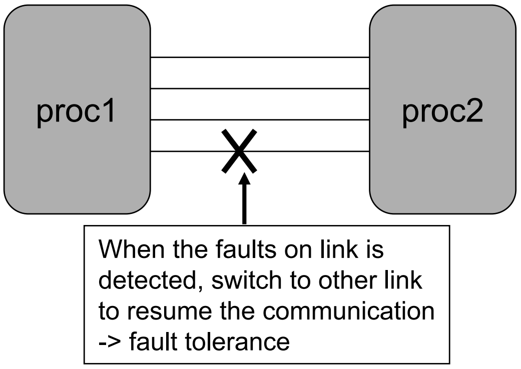
Reliable Software Distributed Shared Memory System for Parallel Embedded Systems

- Software Distributed Shared Memory (DSM)
  - Provides shared memory by software
  - OpenMP can be used to develop parallel program
  - At the point of barrier synchronization, shared memory consistency is maintained.
  - Home node of the pages keep the consistent contents of pages in a conventional DSM
- Reliable Software DSM
  - By having redundant home nodes, the content of a page can be recovered when the faults occurs at one home node.
  - A kind of coordinated checkpoint of parallel program.
  - Local memory also should be check-pointed by conventional check pointing.
- Optimization for embedded systems
  - Remote paging to other processors (swap-out to different processor memory)
  - Disk-less support
  - Small foot-print

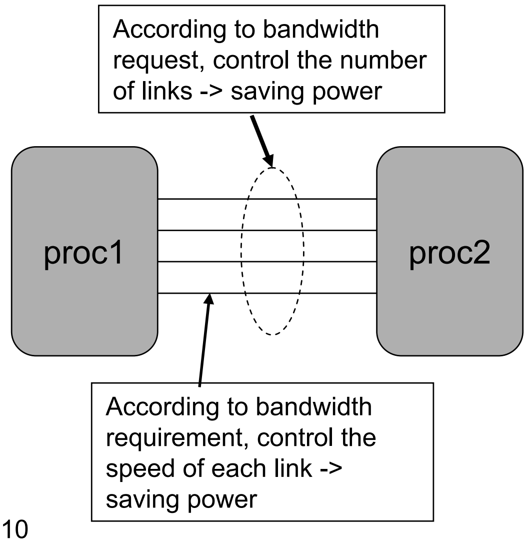


Reliable high-performance system interconnect facility

- We will develop a communication layer to realize high-performance and high-reliability, power-awareness using multiple links of high speed interconnect simultaneously.
  - Use many links (trunking) for high performance
  - Adjust the number of links for power saving
  - Switch between links when faults are detected
  - PCI-Express Gen2 and GbE link



When the faults on link is detected, switch to other link to resume the communication -> fault tolerance



According to bandwidth request, control the number of links -> saving power

According to bandwidth requirement, control the speed of each link -> saving power

- Remote memory communication (one-sided), DMA transfer, page transfer API for software DSM.
- Link fault detection mechanism
- Based on our previous research “RI2N: Redundant Interconnection with Inexpensive Network”
  - T. Okamoto, S. Miura, T. Boku, M. Sato, D. Takahashi, "RI2N/UDP: High bandwidth and fault-tolerant network for a PC-cluster based on multi-link Ethernet", Proc. of CAC2007 (included in Proc. of IPDPS2007), CD-ROM, Long Beach, 2007.

High-speed and Low-power interconnect for parallel embedded systems

- We are currently developing a high-speed and low-power interconnect chip (communicator chip) to connect processors and devices in parallel embedded systems
  - Communicator chip as a network switch with packet routing.
  - Adopt PCI-Express Gen2 as network links
    - 5/2.5Gbps/link (selectable)
    - Power management (ON/OFF)
    - It can be used for both interconnect and I/O
- Development of PCI-Express PHY
  - Implementation using 65nm CMOS tech.
  - Power management by 5/2.5 and ON/OFF.
  - Under verification of PHY

Receiver(RX)

Differential input voltage	100mVppd (nom.)
Inherent timing error (TJ)	0.34 UI (min.)
Inherent timing error (DJ)	0.24 UI (min.)
Minimum pulse width	0.6 UI (min.)
Min/max pulse voltage ratio	5 (max.)
PLL band width	8～16MHz
PLL jitter transfer peaking	3.0dB (max.)
Return−loss (differential)	10dB @0.05-1.25GHz (min.)
	8dB @1.25-2.5GHz (min.)
Return−loss (common mode)	6dB @ 0.05-2.5GHz 8min.)
Input impedance (DC)	50 ohm (nom.)
AC common mode voltage	150mVp-p (max.)
Idle detector Threshold	120 mVppd (nom.)

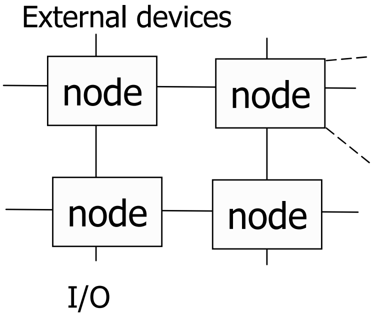
Transmitter(TX)

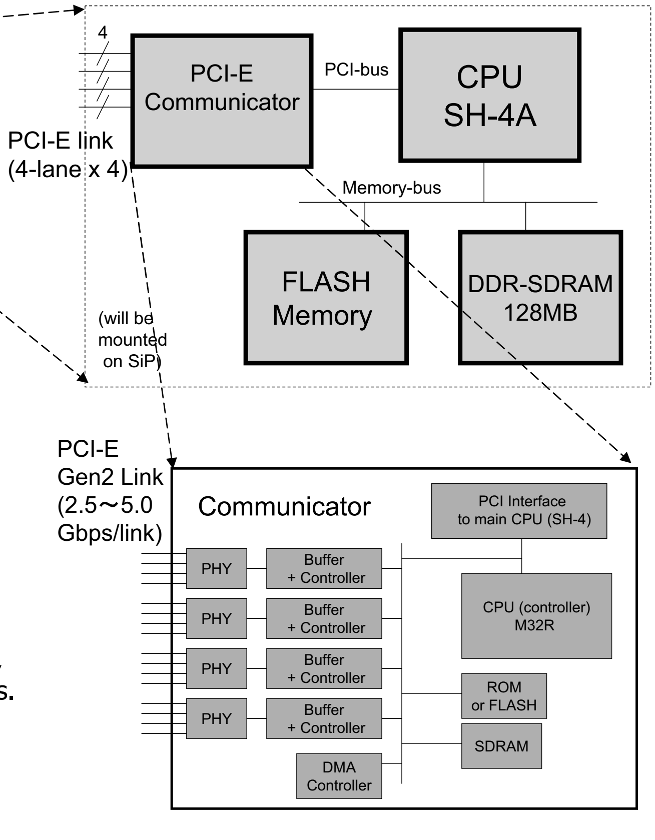
Output voltage	1000 mVppd (nom.)
Output voltage (Low power mode)	500mVppd (nom.)
Eye opening	0.75 UI (min.)
Rise/Fall mismatch	0.1UI (max.)
AC common mode voltage	100mVp-p (max.)
Minimum pulse	0.9UI (min.)
Output impedance	100 ohm (differential nom.)
De-emphasis ratio	-6.0dB @ 5.0Gbps -3.5dB @ 2.5Gbps (nom.)
E-idle peak voltage (AC)	20mVppd (max.)
E-idle voltage (DC)	5mVppd(max.)
PLL band width	8～16MHz
PLL jitter transfer peaking	3.0dB (max.)
Return−loss (differential)	10dB @0.05-1.25GHz (min.)
	8dB @1.25-2.5GHz (min.)
Return−loss (common mode)	6dB @ 0.05-2.5GHz (min.)

Power supply	1.0 V (+/- 5%)
	1.0 V (for TxPLL/RxPLL)
Reference clock	100MHz (+/- 300ppm)

Target Spec of PCI-Express PHY

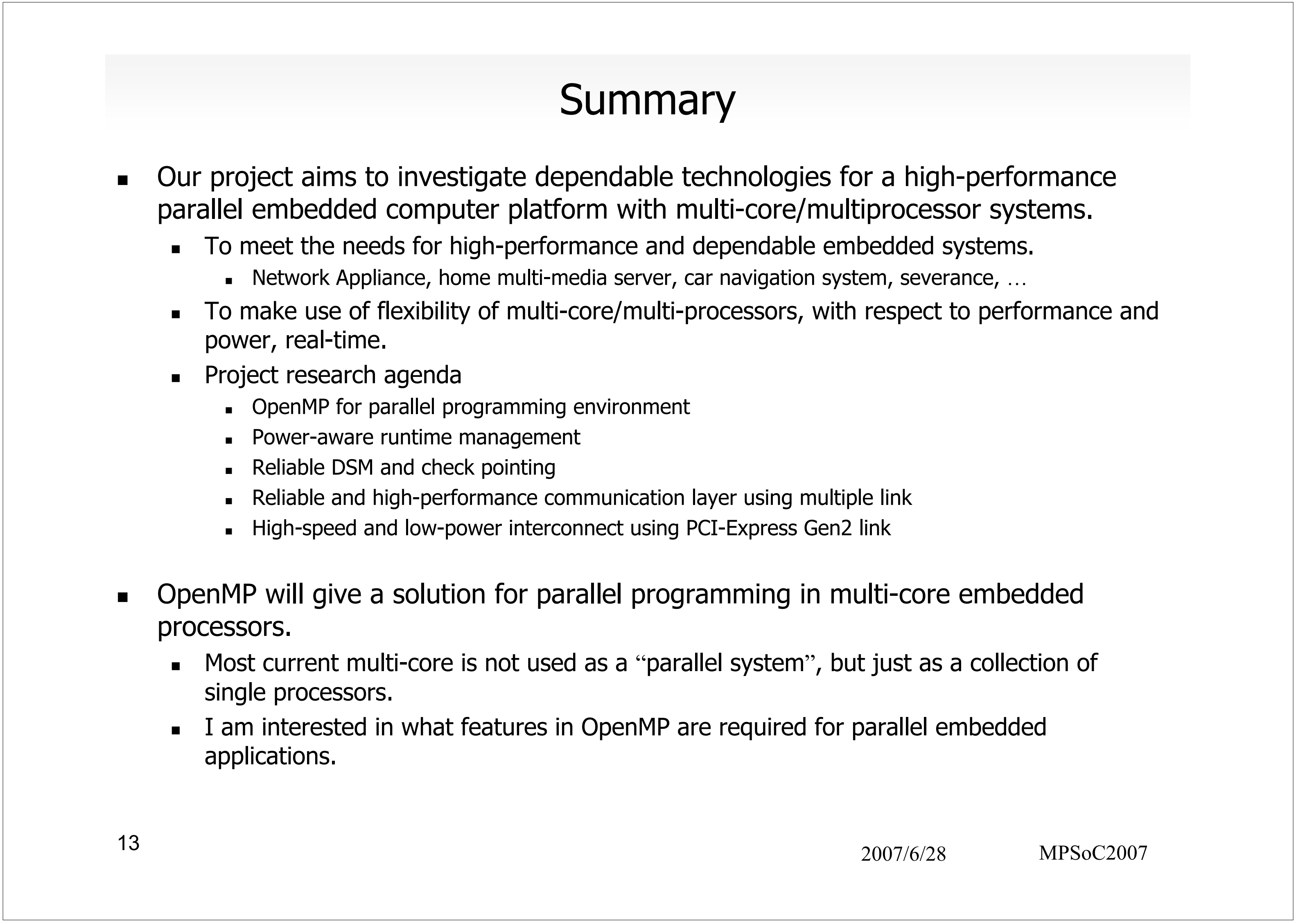
Our Prototype Parallel embedded system





Prototyping of Communicator Chip

- PCI-Exp Link IP
  - Hardware: PCIe Gen2 Link x 4 lane x 4
  - Packet Routing: Software/Hardware co-design for prototyping
    - Switching by CPU (M32R) with DMA Controller
    - CPU checks the header of in-coming packet, and forward to other buffers for destinations.
    - CPU generate the PCI header to send the data.



Summary

- Our project aims to investigate dependable technologies for a high-performance parallel embedded computer platform with multi-core/multiprocessor systems.
  - To meet the needs for high-performance and dependable embedded systems.
    - Network Appliance, home multi-media server, car navigation system, severance, …
  - To make use of flexibility of multi-core/multi-processors, with respect to performance and power, real-time.
  - Project research agenda
    - OpenMP for parallel programming environment
    - Power-aware runtime management
    - Reliable DSM and check pointing
    - Reliable and high-performance communication layer using multiple link
    - High-speed and low-power interconnect using PCI-Express Gen2 link

- OpenMP will give a solution for parallel programming in multi-core embedded processors.
  - Most current multi-core is not used as a "parallel system", but just as a collection of single processors.
  - I am interested in what features in OpenMP are required for parallel embedded applications.