Development of Dependable Network-on-Chip Platform

Tomohiro Yoneda (National Institute of Informatics)
Masashi Imai (Hirosaki University)
Takahiro Hanyu (Tohoku University)
Hiroshi Saito (University of Aizu)
Kenji Kise (Tokyo Institute of Technology)
Recent cars are equipped with many ECUs
- Conventional ECU configuration
Recent cars are equipped with many ECUs
- Centralized ECU approach

Centralized ECUs
Interface cores
CPU cores
Intelligent Sensors/Actuators

CAN, FlexRay, etc.
Recent cars are equipped with many ECUs

- Centralized ECU approach

Any ECU can access any sensors/actuators

ECUs efficiently used by balancing loads
Tasks continuously executed even if some ECUs become faulty (i.e., faulty ECU does not result in malfunction of its specific functions)
Our Project

- Platform for safety-critical automotive applications
  - Dependability in different levels
    - Circuit level
      - variation and delay faults
    - Routing level
      - link, router, chip faults
    - Task execution level
      - processor core faults
  - Evaluation kit
    - Tool for application development
    - Highly practical automotive application example
    - Pseudo plant model
Dependability in circuit level

- Fully asynchronous routers
  - robust against variation and delay faults
  - easily extendable

Odd phase
- 0 → 1
- non-code word
- 0 → 0
- code word for 0
- 1 → 1
- code word for 1

Even phase
- 0 → 1
- non-code word
- 0 → 0
- code word for 0
- 1 → 1
- code word for 1

Code word detection by 0 → 1, 1 → 0
Dependability in routing level

Multi-Chip NoC
- Multiple NoCs are connected via off-chip links
  - On-chip networks seamlessly extended to multi-chip networks
  - Easily implemented thanks to fully asynchronous on-chip network
  - Efficiently implemented thanks to current-mode serial communication links
- Advantages
  - Cost-effective: small NoC chips are cheap, and various sizes of configuration are possible (without developing different sizes of NoCs)
  - Chip-level redundancy: tolerate a chip fault
Dependability in routing level
Dependability in routing level

Dependable routing algorithm
- Online algorithm
  - quick rerouting
  - at most one packet loss
- Simple and distributed algorithm
  - implemented by hardware
  - small performance overhead
- Tolerate single link/router/chip fault
Dependability in task exec. level

- Duplicated execution, comparison, and pair-reconfiguration
Modified Pair & Swap

- Duplicated execution, comparison, and pair-reconfiguration

- Active tasks are also re-executed
  - Transient errors can be masked
Static / Redundant Task Allocation

Task graph

from IO

$T_0$

$T_1$

$T_2$

to IO

Active

$P_0$ $P_1$ $P_2$ $P_3$ $P_4$ $P_5$

Stand-by

Inactive

$T_0$ $T_0$ $T_0$ $T_0$ $T_0$

$T_1$ $T_1$ $T_1$ $T_1$ $T_1$

$T_2$ $T_2$ $T_2$ $T_2$ $T_2$
Static / Redundant Task Allocation

Task graph

from IO

$T_0$

$T_1$

$T_2$

to IO

P0 P1 P2 P3 P4 P5

$T_0$

$T_0$

$T_0$

$T_1$

$T_1$

$T_1$

$T_1$

$T_2$

$T_2$

$T_2$
Static / Redundant Task Allocation

Task graph

from IO

T₀

T₁

T₂

to IO

P₀ P₁ P₂ P₃ P₄ P₅

Alert should be indicated
Temporary TMR configuration

- Active tasks
  - should roll back their state variables
Temporary TMR configuration

- To prepare for TMR configuration, stand-by task usually
  - Receives all input data given to active tasks
  - Receives the state variables updated by active tasks
Example of task execution

Control Cycle

Sensor Inputs
State variables of tasks

Task graph

from IO

to IO
Example of task execution

Control Cycle

Mismatch
P0 fault detected
Fault pattern updated

Roll back state variables

T0

P0

T0

Faulty

i2, i3

TMR

o5, o6, v3

v0

v1

v2

v3

v4

v5

v6

T1

T2

T3

T4

T5

T6

T7

T8

T9

v0

v1

v2

v3

v4

v5

v6

v7

v8

v9

o2, o3

o4, o5

o6, o7

o8

o9

o10

P0

P1

P2

P3

P4

P5

Old state variables are preserved, when state variables are updated

P0 fault detected
Fault pattern updated

Roll back state variables

Mismatch
Evaluation Kit

- Hardware platform
  - 4 chip (4×4 2D mesh), V850E cores ×16
  - Virtex7 FPGA

- Software development tool

- Automotive Application Example

- Pseudo Plant model
Evaluation Kit
Evaluation Kit

Software Development tool

- Given by users
  - Simplex simulink program
  - Task declaration (by specifying atomic subsystems)
  - # of task copies allocated
  - # of processor cores available

- Front-end GUI tool supports
  - Allocation of multiple task copies to redundant processor cores with timing and memory constraints

- Back-end tool supports
  - C code generation for simulink codes
  - Wrapper code templates for receiving and sending data as well as handling TMR configuration
Evaluation Kit

*Automotive Application Example*

- Integrated attitude control system for a four-wheel drive car
  - Torque, brake, and steering control of 4 wheels performed by ECUs
Evaluation Kit

- Pseudo Plant model
  - provide simulation environment like HILS
  - executed on a soft-processor in FPGA
Ongoing work

- IO core duplication
  - IO core plays simple but important roles
    - Implemented by hardware or a small processor
    - Simple crash fault assumed
      - Fixed duplex configuration

Diagram:
- NoC Platform
- IO core
- Data decoder
- A/D, level conv.
- Actuator outputs
- Sensor inputs
- Data generator
- D/A, level conv.
- Data sent to both IO cores at every control cycle
- First data with correct checksum are used
Ongoing work

♦ Maintaining real-time properties
  - Maximum latency should be obtained to check if real-time constraints are satisfied
    - Approach 1: using time slots to avoid congestion
      - E.g. Time-triggered NoCs [C. Paukovits, H. Kopetz: Concepts of Switching in the Time-Triggered Network-on-Chip, RTCSA '08, pp.120-129]
    - Approach 2: using analytical model to estimate maximum latency
Dependable platform for safety-critical automotive applications has been developed

- Multi-Chip NoC based hardware
- Software development tool
- Practical automotive application example
- Pseudo Plant model