Functional Safety in Linux-AP

2013. 10. 24.

Youngsu Kwon
SW-SoC Convergence Research Department
ETRI, Korea

* ETRI=Electronics and Telecommunications Research Institute, Korea
SoCs in Automotive

ASV (Advanced Safety Vehicle)

Safe  |  Smartness  |  Comfort

120 ECUs (~300 CPUs)/Vehicle*

* Source: ETRI Industry Analysis Report 2010

Airbag  |  Blackbox  |  Smart Dashboard (Nav, Audio)

Engine Control  |  Transmission Control  |  TPMS (Tire Pressure Monitoring System)

Smart door  |  Seat Control  |  Steering

Window  |  ABS Control Unit

* Source: ETRI Industry Analysis Report 2010
ADAS and CPU

ADAS (Advanced Driver Assistance System)

Reduce Injuries

Driving - Smartness & Comfort

Standardized Software

High-Performance APs

V2V\(^4\), V2I\(^5\), ITS\(^6\)

Driving-Central

PAS\(^1\)

Danger Warning (LDWS\(^2\), Immediate Braking)

Safe Driving Control (LKAS\(^3\))

1) PAS: Parking Assist System
2) LDWS: Lane Departure Warning System
3) LKAS: Lane Keeping Assist System
4) V2V: Vehicle-to-Vehicle
5) V2I: Vehicle to Infrastructure
6) ITS: Intelligent Transportation System

MCU(40~100MHz)

2000

2010

2020

CPU(1GHz)
ADAS for Smart/Safe Driving; Future of Cars

Smart Driving
- IVI (In-Vehicle Infotainment)
- Voice Recognition
- Gesture Recognition
- Mixed Reality Display
- Route/Parking Guidance
- V2V/ITS

Safe Driving
- Lane Detection
- Traffic Sign Recognition
- Multi-Radar/ACC
- Pedestrian Detection
- Night Vision
- Collision Warning Brake

Automotive Embedded System
- Automotive OS (Future of AGL?)
- Vehicle AP
Why Fault-Tolerance in Linux-AP?

Brake-by-Wire = Control by SW-SoC

Fault Tolerant Automotive SW-SoC

Automotive App. on Linux

Vehicle AP
Malfunction of Linux-AP

ADAS (Advanced Driver Assistance) + Steering requests High-Quality Fault-Tolerant SW-CPU

Sources of Transient Errors
- Noise
- Cosmic ray
- Voltage/Current fluctuation
- Temperature variation

Error Modeling
- SET (Single Event Transient)
- SEU (Single Event Upset)
Challenges in Fault-Tolerant Linux-AP

Fault-tolerance in high-performance AP (>800MHz, ~1.2GHz)
* Plenty of researches in MCU

Redesign of the ‘Core’ for protecting transient error

Redesign of the ‘Core and OS’ to Detect faults and Recover status

Linux kernel/drivers’ design for fault-tolerance
* Previous works are for firmwares

Approval of ‘Core and OS’ as a ‘Safety Element’ in Vehicles
Function Safety Concept
ISO 26262 Roadmap for Automotive APs

- **AP**
  - Systematic Faults
    - Avoidance of Faults in the process
      - Requirements tracking, conf. mgmt.
      - Control & Verification of the design process
      - Control & Verification of usage, maintenance, and changes
    - Avoidance of Bugs in SW
      - Control & Verification of Tool suites (compiler,..)
      - Control & Verification of SW test design
      - Doc. & Verification of HW-SW interfaces
  - HW random faults
    - Analysis
    - Safety Mechanisms
ISO 26262 Roadmap for Automotive APs

- Systematic Faults
- HW random faults

Analysis
- Qualitative Analysis
  - Dependent failures (CCF) analysis
- Quantitative Analysis
  - HW metrics analysis (SPFM, LFM, PMHF)
  - HW metrics verification (fault injection)

Safety Mechanisms
- HW diagnostic Mechanisms (e.g. ECC, DCLS, etc.)
- SW diagnostic mechanisms (e.g. SW tests)
- Measures for CCF avoidance

※ DCLS=Dual-Core Lockstep
### ASIL Definition

<table>
<thead>
<tr>
<th>Controllability Class</th>
<th>C0</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desc.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Controllable in general</td>
<td></td>
<td>Simply Controllable</td>
<td>Normally Controllable</td>
<td>Difficult to control or uncontrollable</td>
</tr>
</tbody>
</table>

#### Severity Class

<table>
<thead>
<tr>
<th>Severity Class</th>
<th>S0</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desc.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No injuries</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Light and moderate injuries</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Severe and life-threatening (probably survive)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Life-threatening (survival?)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Exposure Class

<table>
<thead>
<tr>
<th>Exposure Class</th>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>E3</th>
<th>E4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desc.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Incredible</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Very low probability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low probability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium Probability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High Probability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

QM=Quality Managed, For function but is not a safety concern.
8.3.1 ... Microcontrollers are an integral component of modern automotive systems. They can be developed as a safety element out of context (SEooC).

9.1...An SEooC is a safety-related element which is not developed for a specific item. This means it is not developed in the context of a particular vehicle.
Fault Detection (Current Technology)

- **Protection**
  - Original Data + ECC
  - Mem1
  - Mem2
  - Mem3

- **Redundancy**
  - Thread 1-1
  - Thread 1-2
  - Thread 1-3
  - Core1 ↔ Core2

- **Perturbed Redundancy**
  - Thread 1
  - Thread 2
  - Thread 3
  - Core1
  - Core2

- Physical Isolation gives space diversity
- Delayed lockstep gives time diversity
- It may weak to CCF (Common Cause Failure)
### Redundancy suppressing Failures

#### error sources

<table>
<thead>
<tr>
<th>Type</th>
<th>Cause</th>
<th>Effects</th>
<th>Lifetime</th>
</tr>
</thead>
<tbody>
<tr>
<td>EOS</td>
<td>Electrical Over-Voltage/Current Stress</td>
<td>Hot spots</td>
<td>&gt; 1ms</td>
</tr>
<tr>
<td>ESD</td>
<td>Electro-Static Discharge</td>
<td>Upto 1A discharge current</td>
<td>100ps to 1us</td>
</tr>
<tr>
<td>Cosmic Radiation</td>
<td>External environment (&gt;1MeV)</td>
<td>~100fC charge localized in a few um</td>
<td>&lt; 100ps</td>
</tr>
<tr>
<td>Intrinsic Radiation</td>
<td>Alpha from package (~8MEV)</td>
<td>Hole-electron pair generation in a few um</td>
<td>&lt; 100ps</td>
</tr>
</tbody>
</table>

*Re-analysis of Infineon’s Presentation"
DCLS (1)

- Physical Isolation gives space diversity
- Delayed lockstep gives time diversity
- It may weak to CCF (Common Cause Failure)
DCLS (2)

- Reduced delay lines
- Shadow core does inverse of the core
- Physical separation
DCLS in Action

V850 E2M

Logic BIST
ECC
Flash

SPF
MPU (mem)
INT
DMA
PBUS IF
ECC
RAM IF

CPU Master

2 clock delay

2 clock delay

Compare Unit

Flash IF
ECC
Logic BIST

CPU Checker

WDT

Clock Gen

Ring OSC

Clock input

Logic BIST
ECC
BIST RAM

ECC
RAM
BIST

Peripherals
DCLS in Actions

Thread 1a

Thread 1b

Thread 2

* Excerpts from Infineon
Fault-Tolerant Aldebaran Platform
Automotive AP Dev. Process in ETRI

- Appropriate development procedure
- Measurement for analysis (FMEA)
- Review by required organization
- Documentation

ACCDEP (Automotive CPU Core Design Process in ETRI)

- Overall Architecture Design
- Function Verification
- Fault injection & analysis
- Fault Tolerance Mechanism
- Final Design Tape-out
- Intermediate Review
- Synthesis
- P&R
- Reliability Analysis
- Fab.
- Packaging
- Reliability Test (AEC-Q100)
- Review & Final Document
Basic SoC Design Methodology

1. Requirements
2. RTL Model
3. Synthesize
4. Gate-level Model
5. Place & Route
6. Timing Model
7. ASIC or FPGA
8. Test Bench
9. Simulate
ACCDEP in V Model

Hazard and Risk Analysis

ASIL safety goal

Safety requirements from Customer

Safety requirements (generic, architecture, assumption)

Specification of Safety Integrity Measures (SoC+SW)

Input for Part 5,6

ACCDep

Output for Part 5,6

Item integration, Safety Validation & Assessment

Safety Report (FMEA, Metrics)

Safety Analysis and Validation

SW-SoC Design Group
Required Expertise in ACCDEP

- Involvement & Understanding of Standardization
- Safety Analysis Experiments For Automotive Applications (LKAS, Braking, Airbag, ..)
- Cooperation with external institutes For safety assessment
- Automotive AP/MCU Design Technologies
  - SW-SoC Technology
  - Core development
  - DCLS, Assertion, Voting, ECC, ..
- FMEA Methodology Setup Fault injection & Simulation
Faults and Errors in ACCDEP

- Error Discrepancy
  - Systematic Error
  - Random Error
- Fault
  - Abnormal condition causing failure
  - Error in the process
  - Bugs in SW
  - Bugs in HW design
  - Permanent Chip Failure
  - Transient Error
- Failure
  - Termination of the ability

Control & Verification
- Simulation, Testing, Static analysis
- Simulation, Testing, Formal verification
- Stress test (AEC-Q100)
- Fault-Tolerance Techniques
ETRI’s Roadmap for Embedded Processors

**AP for Embedded Products**
(300K gates, 99.8 mW@65nm/core)

**Compact Embedded DSP**
(35K gates, 3 mW@130nm/core)

**2006**

- **“EMP-S”**
  - Single-Core DSP
  - 160 instructions
  - 180MOPS@130nm

- **“MOSAIC”**
  - MOSAIC Multi-Core Video SoC

**2008**

- **“EMP-D”**
  - Dual-Core
  - Multi-Port SPM
  - 160 inst.
  - 500MOPS@130nm

**2010**

- **“Aldebaran-S”**
  - Multi-Port SPM
  - Touch Sensor2
  - Sound Effect SoC

**2012**

- **“Aldebaran-V”**
  - Fault-Tolerant Vehicle AP
  - x2 1.6GOPS@800MHz
  - 32-bit Dual-Core MMU (TLB, Cache)

**2014**

- **“Aldebaran-R”**
  - Many-Core CPU
  - x32-x64 core CPU
  - Fault-Tolerant CPU as SEooC in ISO 26262

**2016**

- **“Aldebaran-C”**
  - Multi-Channel Video Codec AP
  - HEVC (4K/8K video codec)
  - SPMD Array Processor

---

Automotive Linux Summit Fall 2013
Positioning

Simple “number of cores” war will face the end. The power efficiency war is the next round.

Power (mW)

Performance (DMIPS)

Too Hot for Mobile Applications! (1.1W)

Cortex-A8, x1:
1200DMIPS, 300mW, 600MHz@40nm

Cortex-A9, x2:
4000DMIPS, 500mW, 1GHz@40nm

Cortex-A15 (to come), x2:
8400DMIPS, ~900mW, 1.2GHz@28nm

Aldebaran:
x8, 18400DMIPS, 150mW@45nm
Aldebaran Development Platform

< Architecture >

< Xilinx FPGA Platform >
LCD(1200x800)+FPGA b/d+Base b/d
Aldebaran-S2 (Dual-Core)

< Block Diagram of Aldebaran >

< Aldebaran Layout >
Features of Aldebaran

- Dual-issue in-order superscalar with 32bit I/D
- Target: 800MHz@65nm, 1.1V, 1GHz@45nm, 1.0V
- BTB: 2-way x 256-entry x 58-bit = 3.7Kbytes
- BP: 10-bit GHR, 256x16x2b = 1Kbyte
- I/D cache: Each 32K bytes, Tag 2.12Kbytes, I$+D$ 68.25Kbytes
- TLB: Each 32-entry PTE (Page Table Entry) for iTLB/dTLB
  - Separate iTLB/dTLB, each 32 entries
  - Each 65-bit PTE with selective FLUSH/PROBE
- Dual-rail decode and in-order scheduler w/ Scoreboard
- Execution queue
  - Queue containing decoded/scheduled blocks
  - Run-time OS support for LP execution
- Superscalar execution unit
  - 2 integer units, 1 load store, and FPU for single/double fpu operations
  - Multi-port register file
- 800MHz, 2.63mm²@65nm

Clocks in Aldebaran

<table>
<thead>
<tr>
<th>Clock Net</th>
<th>Frequency</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>osc_clki (ref_clk)</td>
<td>50MHz</td>
<td>PLL reference clock</td>
</tr>
<tr>
<td>SCLKNET/core_clk</td>
<td>500MHz~1GHz</td>
<td>Core block</td>
</tr>
<tr>
<td>SCLKNET/bl_clk</td>
<td>core_clk/2, 250MHz~500MHz</td>
<td>BL bus</td>
</tr>
<tr>
<td>SCLKNET/br_clk</td>
<td>200MHz</td>
<td>BR bus</td>
</tr>
<tr>
<td>SCLKNET/sdr_clk</td>
<td>166MHz</td>
<td>SDR clock</td>
</tr>
<tr>
<td>SCLKNET/video_clk</td>
<td>80MHz</td>
<td>VC clock</td>
</tr>
<tr>
<td>osc_clk48m (usb_clk)</td>
<td>48MHz +/- 0.2%</td>
<td>USBHS clock</td>
</tr>
<tr>
<td>pdd_sys, pdd_ref(ddr2 4 clks)</td>
<td>200MHz</td>
<td>SNAKEM_DDR2</td>
</tr>
<tr>
<td>psd_clk_in</td>
<td>166MHz</td>
<td>SDR feedback</td>
</tr>
<tr>
<td>pjt_tck_in</td>
<td>10MHz</td>
<td>SJTAG clock</td>
</tr>
<tr>
<td>pac_bit_clk_pad_i</td>
<td>12.8MHz</td>
<td>SNAKEM_AC97</td>
</tr>
<tr>
<td>sdio internal clocks</td>
<td>~50MHz (gated from br_clk)</td>
<td></td>
</tr>
</tbody>
</table>
Features of Aldebaran

NoC-Left

- VC: Video Controller
  - EDMA for NoC off-loading
  - 1280x800 resolution, HDMI support
- DRAM Controller
  - 256Mbytes DRAM
  - 166MHz SDR (166Mbps)
  - 200MHz DDR2 (400Mbps)
- iROM: Internal ROM
  - Multiple bootstrap modes
- iRAM: Internal RAM
  - 32Kbytes for complex bootstrapping

NoC-Right

- NFS: NAND Flash controller
  - 128M~32Gbytes NAND support
  - Max 400Mbps
  - Configurable, for various NAND types
- USBHS: USB controller
  - USB (1.1) Host controller
- SDC: SD controller
  - SD card (1/4-bit mode)/SDIO/SPI

NoC-Right

- SMC: SRAM I/F controller
  - Configurable SRAM Interface
  - Interface for LAN9220
- INTC: 32-source PIC
- Timer/WDT:
  - Periodic/One-shot/Watchdog, 4 sets
- CAN:
  - 2-wire CAN for OBD-II, 2 sets
- PWM: Pulse-Width-Modulation
  - Configurable waves
  - LCD backlight, Dimming
- UART: UART 16550
  - 38400/115200 baud rate
- AC97: Audio
  - AC97 codec interface
  - Volume management
- I2C: Inter-IC Control
  - 7-bit/10-bit address
  - I2C master/slave composite
  - LCD Touch Interface
- SJTAG: JTAG Interface
  - PC-Core Communication
  - Core debugging, Program Download
  - On-Chip flash burning
Core Architecture

Legend

- VA: Virtual Address
- BTB: Branch Target Buffer
- BP: Branch Predictor
- FS: Fetch Scheduler
- IQ: Instruction Queue
- DU: Upper Decode
- DD: Down Decode
- S: Scheduler
- SB: Scoreboard
- EQ: Execution Queue
- FS: Fetch Schedule
- IU: Integer Unit
- LS: Load/Store Unit
- RF: Register File
- EP/EE: Execute Prolog/Epilog

- ✔ US 12/832313, “Local stack storage for processors”
- ✔ US 7958321, “Apparatus and method for reducing memory access conflicts among processors”
- ✔ MICPRO 2010, “Partial access conflict-relieving programmable address shuffler for parallel memory system in multi-core processor”
※ Block size is independent of the actual gate count.
Aldebaran Instructions

Load-Store

| LDSB | LDSBA |
| LDSH | LDSHA |
| LDUB | LDUBA |
| LDUH | LDUHA |
| LD | LDA |
| LDD | LDDA |
| LDF | LDF |
| LDF | LDF |
| LDFS | LDFS |
| LDC | LDC |
| LDDC | LDDC |
| LDCSR | LDCSR |
| STB | STBA |
| STH | STA |
| STD | STD |
| STF | STDF |
| STFS | STFS |
| STDFQ | STDFQ |
| STC | STC |
| STDC | STDC |
| STCSR | STCSR |
| STDCQ | STDCQ |
| LDSTUB | LDSTUBA |
| SWAP | SWAPA |
| SETHI | NOP |

Arithmetic

| AND | ANDcc |
| ANDN | ANDNcc |
| OR | ORcc |
| ORN | ORNcc |
| XOR | XORcc |
| XORN | XORNcc |
| SLL | SLL |
| SRL | SRL |
| SRA | SRA |
| ADD | ADDcc |
| ADDX | ADDXcc |
| TADDcc | TADDccTV |
| SUB | SUBcc |
| SUBX | SUBXcc |
| TSUBcc | TSUBccTV |
| TADDcc | TADDccTV |
| MULSc | MULSc |
| UMUL | UMULcc |
| SMUL | SMULcc |
| UDIV | UDIVcc |
| SDIV | SDIVcc |

Flow

| SAVE |
| RESTORE |
| Bicc |
| FBfcc |
| CBccc |
| CALL |
| JMPL |
| RETT |
| Ticc |

Sync

| CAS |
| CASA |
| STBAR |
| UNIMP |
| FLUSH |

Floating-point

| FiTO(s,d,q) |
| F(s,d,q)Toi |
| FsTod |
| FsTq |
| FdTos |
| FdToq |
| FqTos |
| FqToq |
| FMOV |
| FNEG |
| FABS |
| FSQRT(s,d,q) |
| FADD(s,d,q) |
| FSUB(s,d,q) |
| VMUL |
| FSMUL |
| FSHF |
| FCMP |
| FCMP |

Vector

| VLD |
| VST |
| VADD |
| VSUB |
| VMUL |
| VMUL |
| VSUM |
| VABS |
| VAND |
| VOR |
| VSHF |
| VSQR |

Register move

| RDASR |
| RDY |
| RDPSR |
| RDWIM |
| RDTBR |
| WRASR |
| WRY |
| WRPsr |
| WRWIM |
| WRTBR |
I Cache; VIPT

Tag size = (2^8 x 19 x 4)
Data size = (2^8 x 32 bytes x 4)

Hit way

Mux

4-way Blocks


TLB


Inst[31:0]

✔ US 출원중, “Apparatus for saving energy of a cache using scratch pad memory”
✔ JCSC 2010, "Application-adaptive reconfiguration of memory address shuffler"
D Cache; PIPT

Data[31:0] = select

Tag size = (2^19bx4)

Data size = (2^32bytesx4)


TLB

PA[35:2]


Write Buffer

Data[31:0]

Miss

Hit

inst inst

inst

Block
Branch Prediction; Gshare

To reduce “aliasing” by PC, use XOR

GHR[4:0]^PC[4:0]

PHT (Pattern History Table)

GHR (Global History Register)

GHR[9:2]

256 entries

Hit ratio

Gshare

Automation Linux Summit Fall 2013
Instruction Queue & Bandwidth

- Fast Branch Detection
- Dual instruction Selector
- decode_u
- decode_d
Scheduler & Fire-and-go

Scheduler

Flow Control

Hazard Resolution

int0_e0  int1_e0  ldst_e0  fp_e0
int0_e1  int1_e1  ldst_e1  fp_e1
int0_e2  ldst_e2

Fire-and-go

e0 → e1 → e2

e1 → e2 → e3
Page Table and TLB

TLB

VA → PA
Flush (partial/full)
Probe

iTLB
entry 0
entry 1
entry 31

dTLB
entry 0
entry 1
entry 31

c_ctx
r_ctpr
r_fsr
r_far

VA tags  ctx  PTE

PPN  C  M  R  ACC  ET

36-bit physical address

VA
Index1  Index2  Index3  Offset

r_ctpr (12b)

L1 pPTD
L1 pPTD
L1 pPTD

L1 PTD/E
L1 PTD/E
L1 PTD/E

256 entries  64 entries  64 entries

Automotive Linux Summit Fall 2013
## Traps

<table>
<thead>
<tr>
<th>Trap name</th>
<th>#tt</th>
</tr>
</thead>
<tbody>
<tr>
<td>reset</td>
<td>0x2b</td>
</tr>
<tr>
<td>data_store_error</td>
<td>0x3c</td>
</tr>
<tr>
<td>instruction_access_error</td>
<td>0x21</td>
</tr>
<tr>
<td>instruction_access_exception</td>
<td>0x01</td>
</tr>
<tr>
<td>privileged_instruction</td>
<td>0x03</td>
</tr>
<tr>
<td>illegal_instruction</td>
<td>0x02</td>
</tr>
<tr>
<td>fp_disabled</td>
<td>0x04</td>
</tr>
<tr>
<td>cp_disabled</td>
<td>0x24</td>
</tr>
<tr>
<td>window_overflow</td>
<td>0x05</td>
</tr>
<tr>
<td>window_underflow</td>
<td>0x06</td>
</tr>
<tr>
<td>mem_address_not_aligned</td>
<td>0x07</td>
</tr>
<tr>
<td>fp_exception</td>
<td>0x08</td>
</tr>
<tr>
<td>cp_exception</td>
<td>0x28</td>
</tr>
<tr>
<td>data_access_error</td>
<td>0x29</td>
</tr>
<tr>
<td>data_access_exception</td>
<td>0x09</td>
</tr>
<tr>
<td>tag_overflow</td>
<td>0x0a</td>
</tr>
<tr>
<td>division_by_zero</td>
<td>0x2a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Trap name</th>
<th>#tt</th>
</tr>
</thead>
<tbody>
<tr>
<td>trap_instructions</td>
<td>0x80~0xff</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Trap name</th>
<th>#tt</th>
</tr>
</thead>
<tbody>
<tr>
<td>interrupt_level_15</td>
<td>0x1f</td>
</tr>
<tr>
<td>interrupt_level_14</td>
<td>0x1e</td>
</tr>
<tr>
<td>interrupt_level_13</td>
<td>0x1d</td>
</tr>
<tr>
<td>interrupt_level_12</td>
<td>0x1c</td>
</tr>
<tr>
<td>interrupt_level_11</td>
<td>0x1b</td>
</tr>
<tr>
<td>interrupt_level_09</td>
<td>0x1a</td>
</tr>
<tr>
<td>interrupt_level_09</td>
<td>0x19</td>
</tr>
<tr>
<td>interrupt_level_08</td>
<td>0x18</td>
</tr>
<tr>
<td>interrupt_level_07</td>
<td>0x17</td>
</tr>
<tr>
<td>interrupt_level_06</td>
<td>0x16</td>
</tr>
<tr>
<td>interrupt_level_05</td>
<td>0x15</td>
</tr>
<tr>
<td>interrupt_level_04</td>
<td>0x14</td>
</tr>
<tr>
<td>interrupt_level_03</td>
<td>0x13</td>
</tr>
<tr>
<td>interrupt_level_02</td>
<td>0x12</td>
</tr>
<tr>
<td>interrupt_level_01</td>
<td>0x11</td>
</tr>
</tbody>
</table>
### IDE w/ GCC

#### C/C++ Compiler

<table>
<thead>
<tr>
<th>Compiler + Libraries</th>
</tr>
</thead>
<tbody>
<tr>
<td>gcc, g++, ar, as, cpp, gcj</td>
</tr>
<tr>
<td>gcov, gprof, ld, objcopy, objdump, readelf</td>
</tr>
<tr>
<td>crt1.o, lib, pthread, libm, librt, libc, stdc++</td>
</tr>
</tbody>
</table>

#### Aldebaran SW Ecosystem

**OCD**

The small-sized server SW to communicate with JTAG-based OCD implemented in the chip ※ OCD: On-Chip Debugger

**IDE**

Integrated development environment GUI with Compiler, Assembler, Debugger

**Debugger**

Client software for C/C++ Source-Level Debugging

**Emulator**

Modeling of core, tlb, mmu, and the main memory

**Monitor**

Probing and control of the core through JTAG

**Verification Apps**

Core verification, performance measurement for Aldebaran such as SpecCPU, CoreMark, etc.

#### Linux

**Kernel (3.3)**

- Media drivers
- Flash driver
- Frame buffer
- MMU mgt.
- Graphics Library
- Web
- OpenGL
- OpenCL

**Bootloader**

**Applications**

- Linux drivers
- Graphics library
- Media
- Flash
- Frame buffer
- MMU

#### IDE w/ GCC

Automotive Linux Summit Fall 2013
Verification Basics
Area and Power

**Comparison**

- **Aldebaran Core**
  - 0.125mW/MHz

- **MIPS 1074kf**
  - 0.36mW/MHz

- **ARM Cortex-A9**
  - 0.625mW/MHz

* Excerpted from MIPS, ARM’s website
* Power efficiency depends on synthesis constraints
Fault Analysis in ACCDEP

Architecture

Fault Simulation SW

Modeling

\[ P(o_i) = P_m(m, o_i) + \sum_j P_m(i_j, o_i) \]
Aldebaran-V (Concept)

Micro-flushing for fault detection and recovery
- detection: spatial, time, logical diversity
- tolerance: 2oo3 voting
- recovery: micro-flushing on failure detection

Pipeline redundancy

Failure detected $\rightarrow$ Micro-Reset

2 cycle delay

Internal pipeline architecture
Aldebaran-V (Register-based Micro-flushing)

Pre-Core 0

Pre-Core 1

Core (actual)

Failure detected ➔ Micro-Reset

Register File

Update History (for 2 cycles)

r0  r0-ecc
r1  r1-ecc
r31 r31-ecc

Register File

Update History (for 2 cycles)

r0  r0-ecc
r1  r1-ecc
r31 r31-ecc

Register File

TMR

r0  r0-ecc
r1  r1-ecc
r31 r31-ecc
Kernel-AP Interaction in Aldebaran-V

Linux

Timer isr

Do backup for faulty process

Fault Table

Thread ID

PC

Interval

Kernel

AP

Normal operation

2oo3 matches?

yes

Fault detected

Micro-flushing

no

Rewind

AP-driven fault history

Automotive Linux Summit Fall 2013
Implementation
Summary

Development of High-performance CPU for Next-generation Automotive CPU is required. (Integrity + Functional Safety)

The Functional Safety Expert Group calls for participants interested in Functional Safety/Fault-Tolerance in AGL
강사합니다
Automotive V-Model

Development of Car System

Development of Sub-System

Development of Mechanical Parts

ECU Development

ECU SW Development

ECU HW Development

ECU SW Implementation

ECU HW Integration and Test

ECU HW/SW Integration and Test

ECU Sign-Off

ECU sensors, actuators, Mechanical parts Integration, Calibration, and Test

Sub-System Sign-Off

Sub-Systems integration Test, and validation

Car System Sign-Off

Integration, Calibration, and Test

Car System Sign-Off
Trends of High-Performance Automotive AP

Automotive’s Complexity Increases Exponentially

- Need for Functional Safety
- 100 MCUs, 150 pounds of wiring
- $10^7$ lines of code

High-Performance Multi-Core Automotive MCUs

- Freescale MPC5748G: Multi-Core MCU
- Renesas R8A7790X: Octa(8x) Automotive MCU