

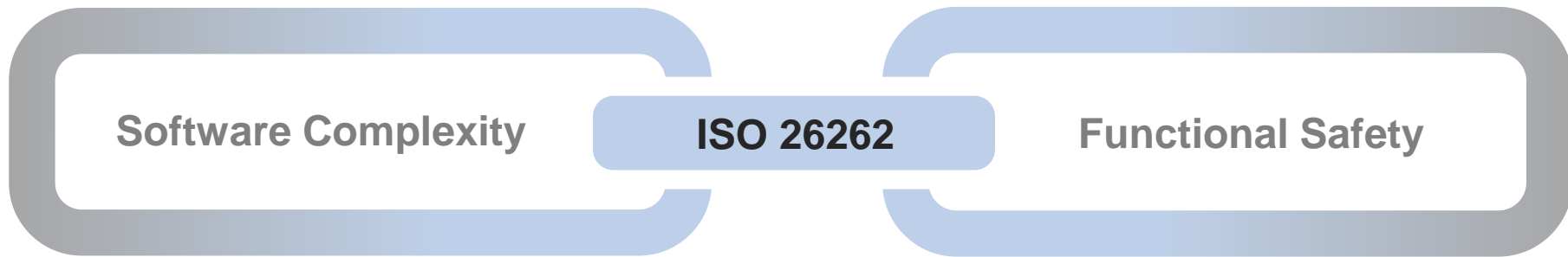
Satisfying ASIL-D Code Coverage Objectives in the Target Environment without Code Instrumentation

Embedded Testing — Munich • 04 / July / 2019

Agenda

- **Introduction**
- **Modern Automotive Chips**
- **Making Use of the Trace Capability**
- **Measuring Code Coverage without Instrumentation**
- **Conclusion**

Functional Safety for Complex Software



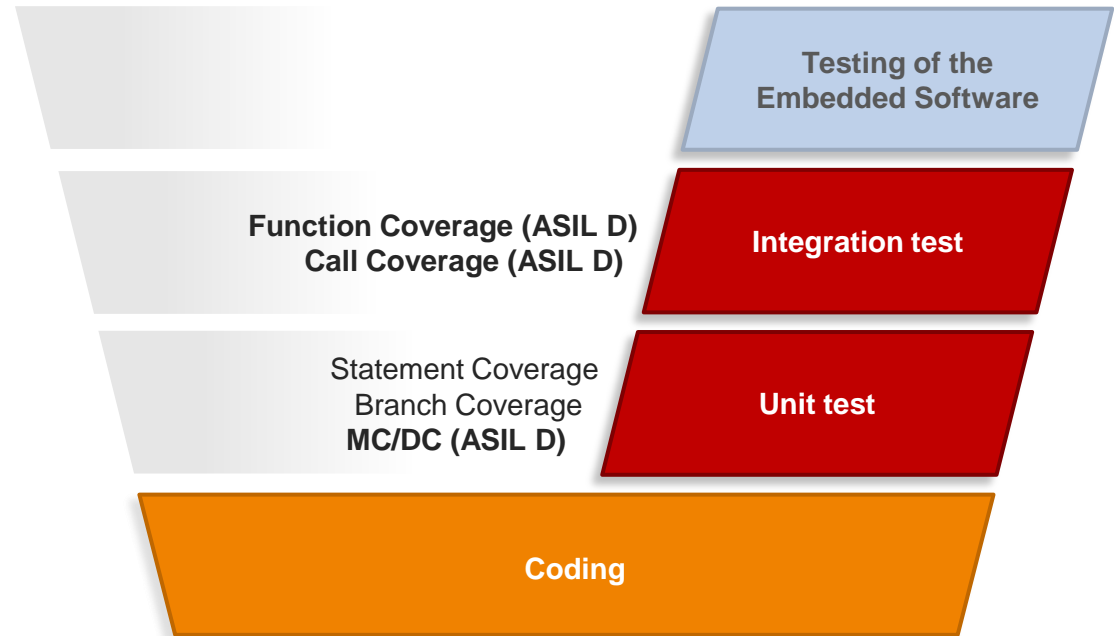
“Software is eating the world”* is catching the automotive industry

- Increasing software complexity
- Bundling of software components with mixed criticality into a small number of powerful ECUs
- Demand for high software portability across different configurations and platforms

* Marc Andreessen, 2011

Software Verification in ISO 26262

- Multiple testing stages with different focuses
- Code coverage as measure of completeness and adequacy of testing activities
- Emphasis of testing in the target environment

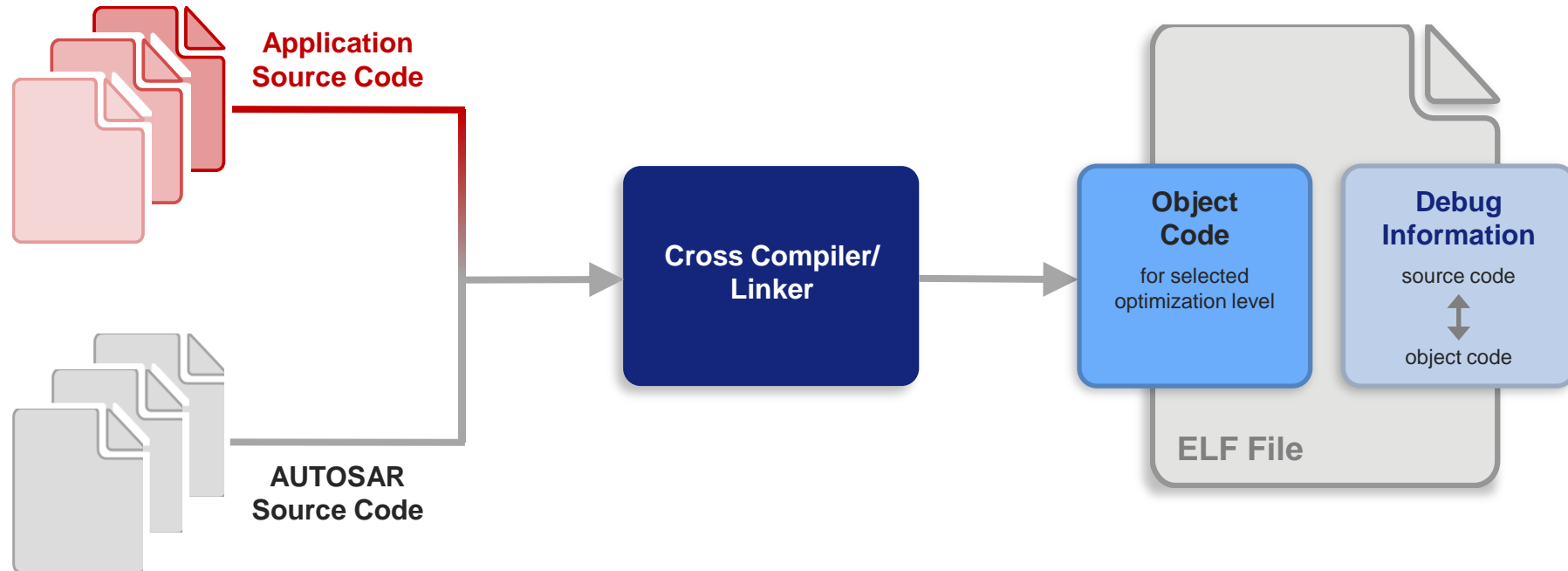


Goal of the Presentation

Introduce a **code coverage measurement method** that

- is applicable to all code coverage metrics mentioned in ISO 26262
- works without code instrumentation
- is carried out in the target environment

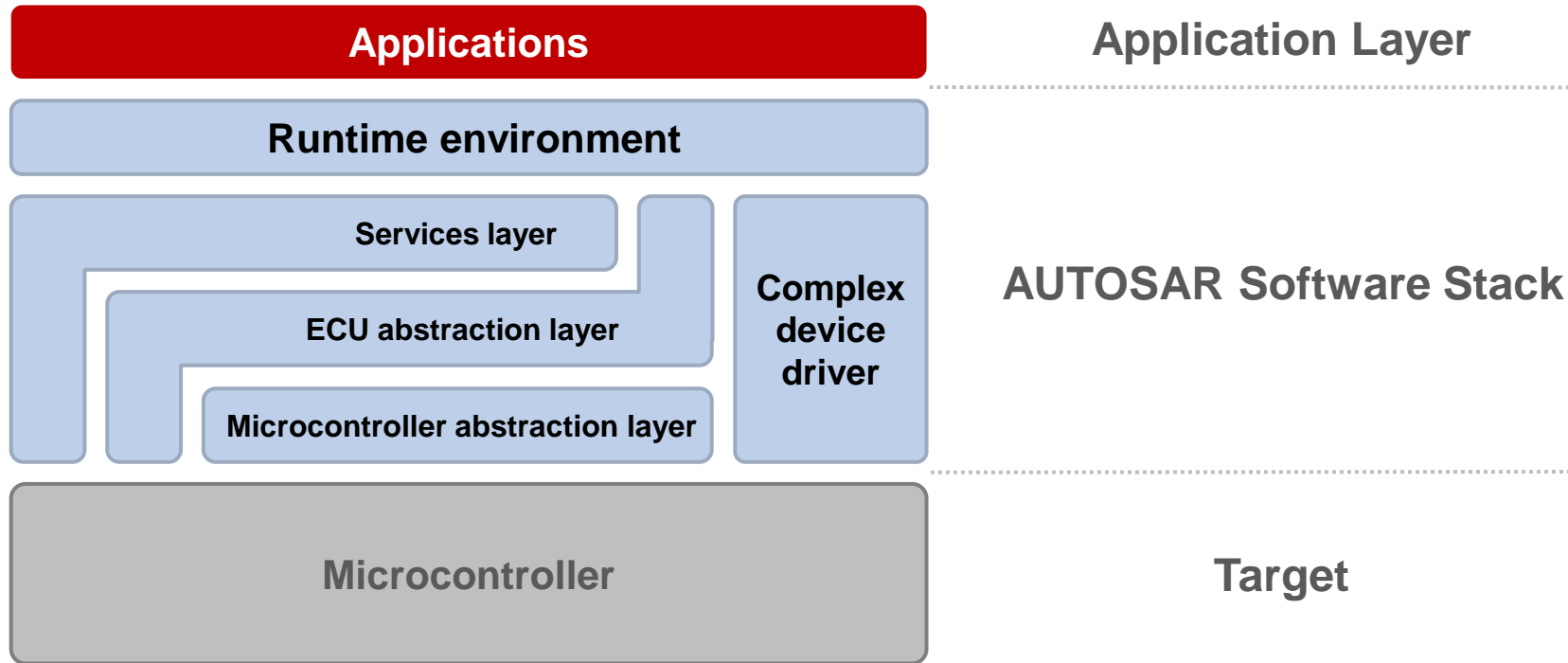
ELF File



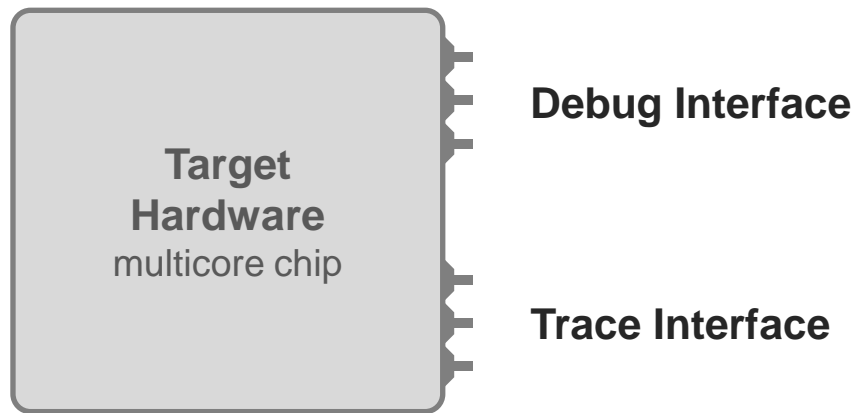
Agenda

- Introduction
- **Modern Automotive Chips**
- Making Use of the Trace Capability
- Measuring Code Coverage without Instrumentation
- Conclusion

Test Environment (Automotive Component)

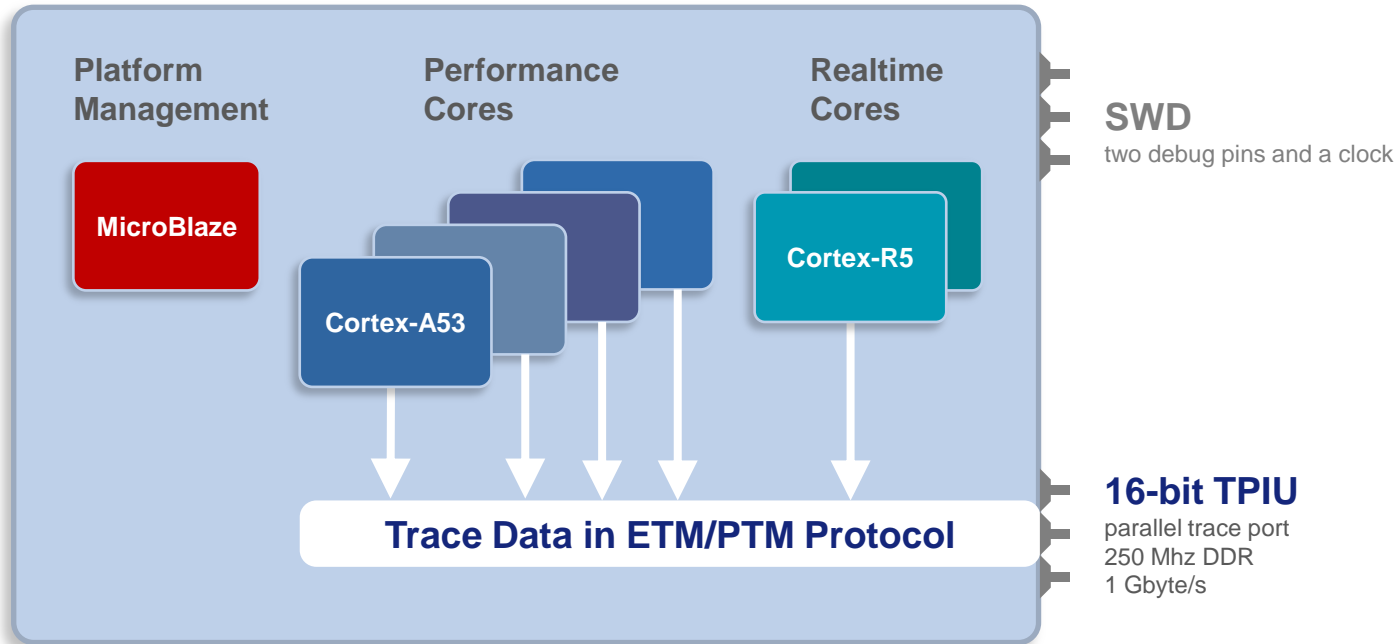


Test Environment (Automotive Component)

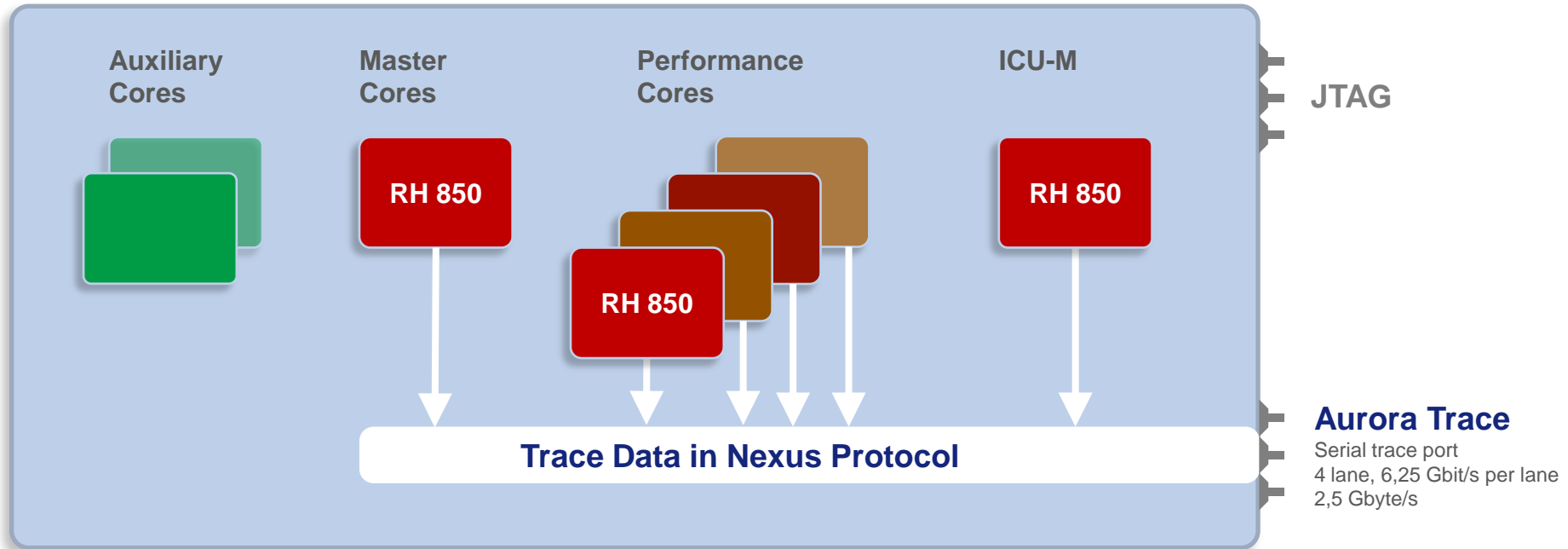


- Debug and Trace interface offer the possibility to get details about the program execution
- Debug and Trace interface is already used in many projects for profiling purposes

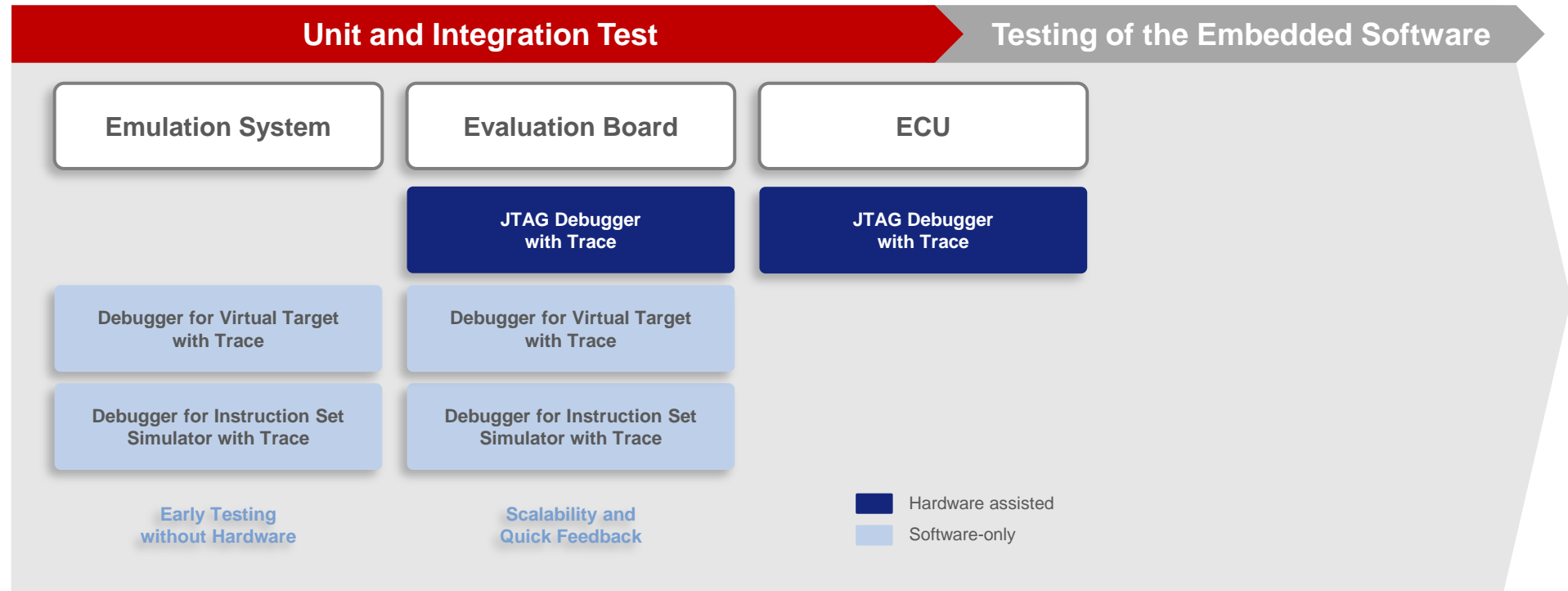
Parallel Trace Port: Zynq UltraScale+



Serial Trace Port: High-End RH850



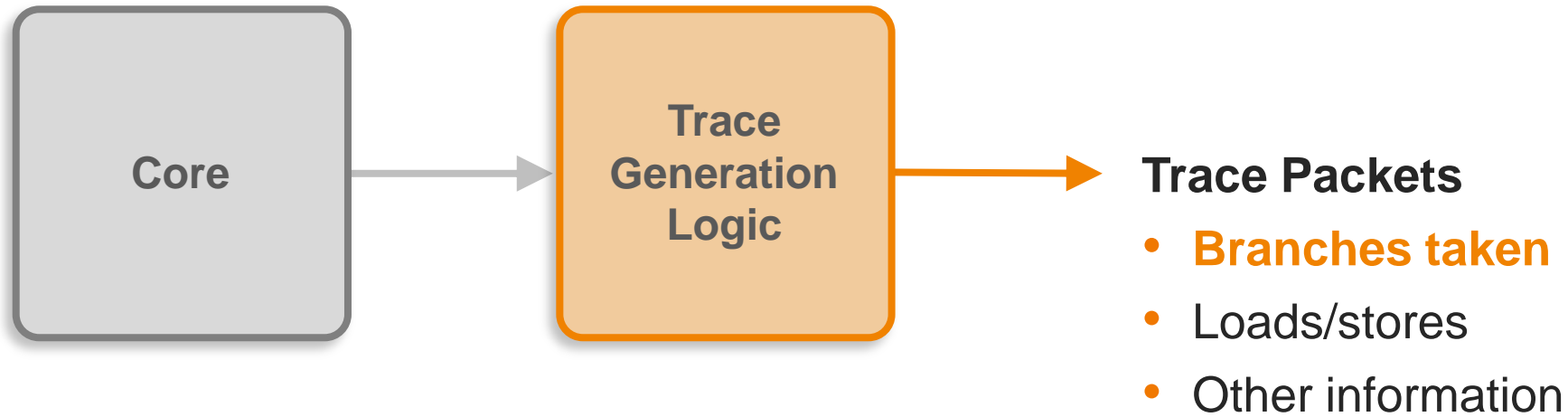
Debug and Trace Tools for All Test Phases



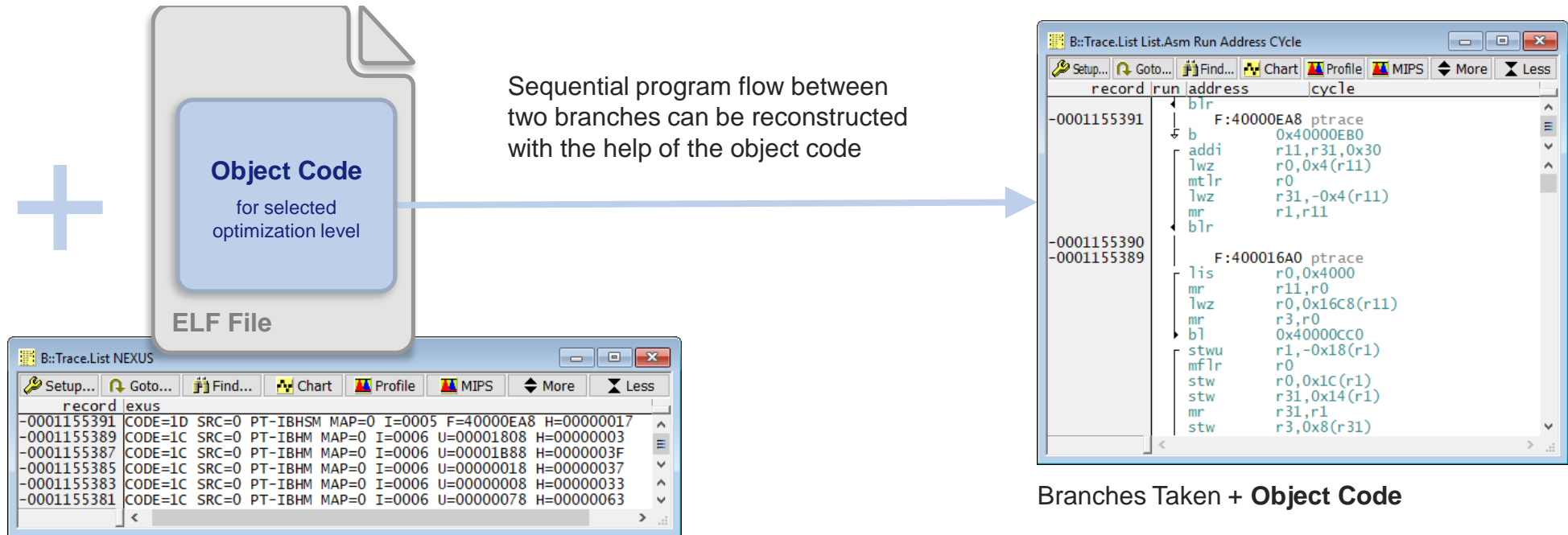
Agenda

- Introduction
- Modern Automotive Chips
- **Making Use of the Trace Capability**
- Measuring Code Coverage without Instrumentation
- Conclusion

Core Trace



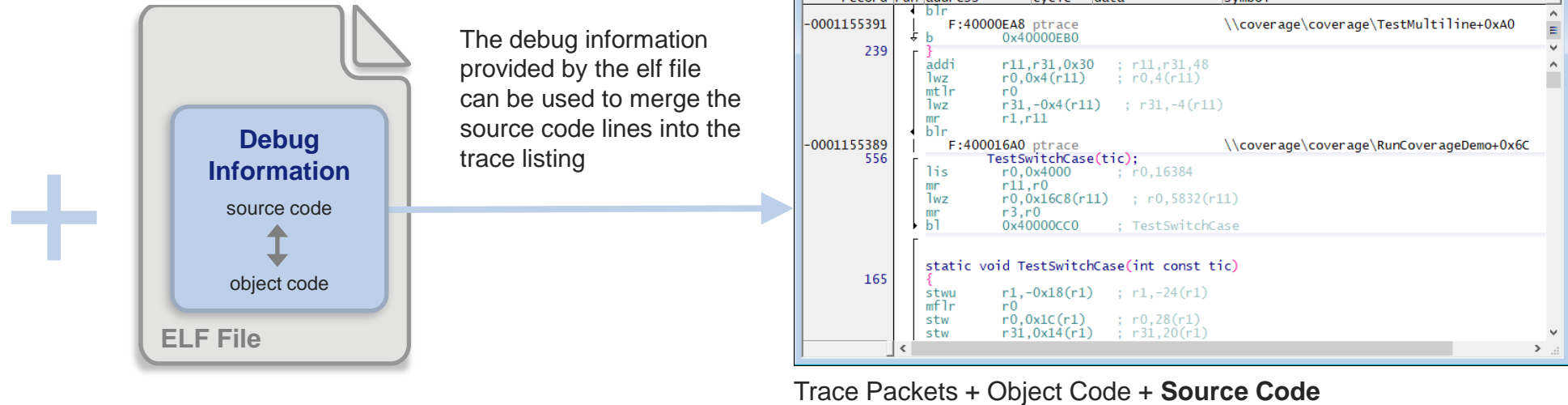
Program Execution Details Derived from Trace Packets



Branches Taken (Nexus)

Branches Taken + Object Code

Bridging the Gap to the Source Code



Agenda

- Introduction
- Modern Automotive Chips
- Making Use of the Trace Capability
- **Measuring Code Coverage without Instrumentation**
- Conclusion

Trace-based Code Coverage

Unit Testing

- Statement Coverage
- Branch Coverage
- **MC/DC (ASIL D)**

Integration Test

- **Function Coverage (ASIL D)**
- Call Coverage (ASIL D)

Function Coverage

Analyzes: Functions

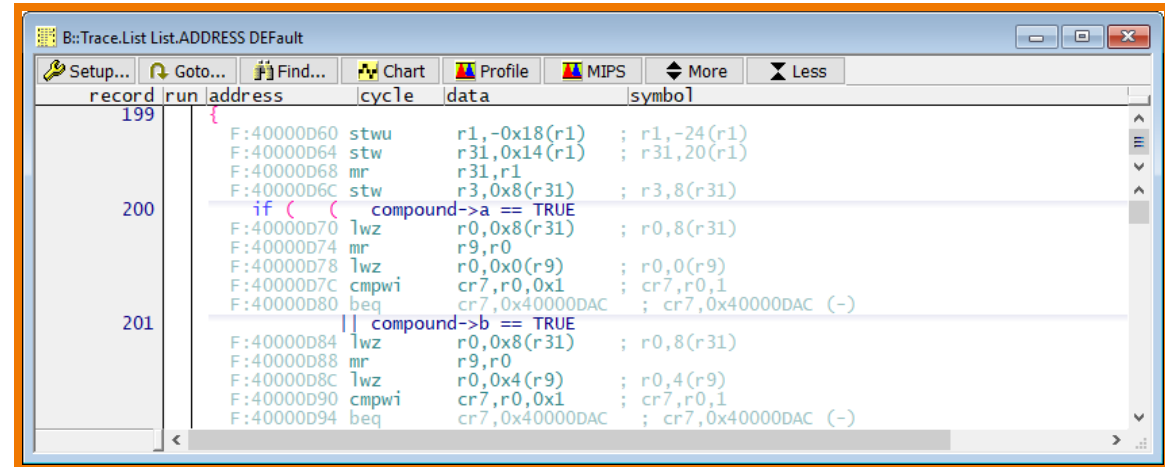
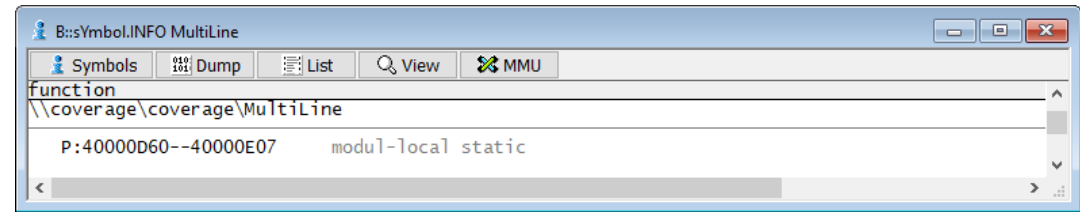
Definition: Every function in the program has been invoked at least once.

Trace-based Code Coverage: A function achieved *Function Coverage* when at least one corresponding object code instruction has been executed

Function Coverage

Execution of an object code line

➔ function coverage 100%



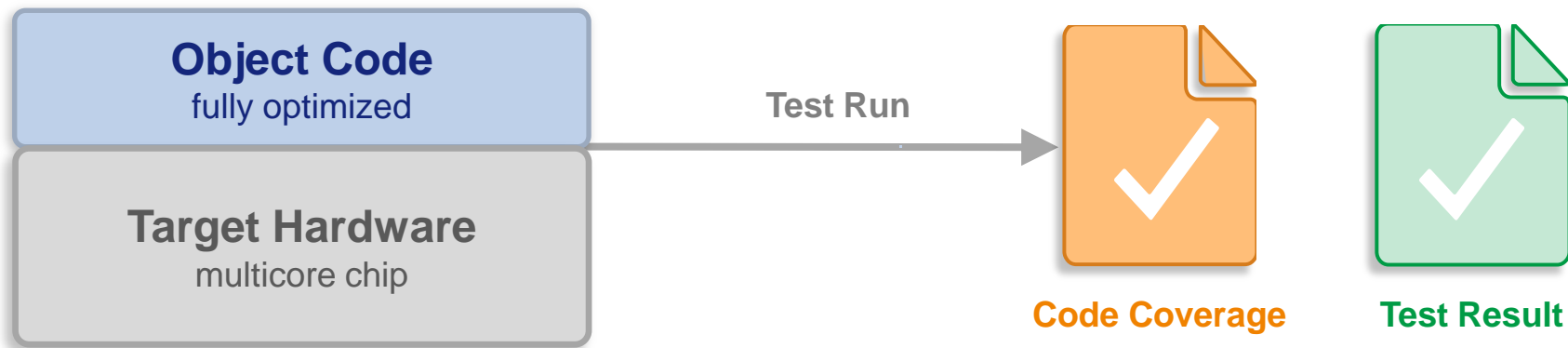
Function Coverage

B::COVerage.ListFunc \coverage

Setup... Goto... List + Add Load... Save... Init

| address | tree | coverage | executed | 0% | 50% | 100% |
|----------------------|--------------------|------------|----------|----|-----|------|
| P:40000610--400006C7 | \main | incomplete | 50.000% | | | |
| P:40000610--4000062B | main | func | 100.000% | | | |
| P:4000062C--400006C7 | background | incomplete | 0.000% | | | |
| P:400006C8--400016C7 | \coverage | func | 100.000% | | | |
| P:400006C8--400006EF | Identity | func | 100.000% | | | |
| P:400006F0--4000079B | ComplexFor | func | 100.000% | | | |
| P:4000079C--4000083F | TestComplexFor | func | 100.000% | | | |
| P:40000840--400008F7 | ComplexDowhile | func | 100.000% | | | |
| P:400008F8--4000099B | TestComplexDowhile | func | 100.000% | | | |
| P:4000099C--40000A53 | Complexwhile | func | 100.000% | | | |
| P:40000A54--40000AE3 | TestComplexwhile | func | 100.000% | | | |
| P:40000AE4--40000B7F | ComplexIf | func | 100.000% | | | |
| P:40000B80--40000CBF | TestComplexIf | func | 100.000% | | | |
| P:40000C10--40000CBF | SwitchCase | func | 100.000% | | | |
| P:40000CC0--40000D5F | TestSwitchCase | func | 100.000% | | | |
| P:40000D60--40000E07 | MultiLine | func | 100.000% | | | |
| P:40000E08--40000EC7 | TestMultiLine | func | 100.000% | | | |
| P:40000EC8--40000F77 | NestedExprTrans | func | 100.000% | | | |
| P:40000F78--40001003 | NestedExpr | func | 100.000% | | | |
| P:40001004--4000106F | TestExprNesting | func | 100.000% | | | |
| P:40001070--400010E7 | TernaryExprTrans | func | 100.000% | | | |

Function Coverage



Function Coverage can be performed on final code

Modified Condition/Decision Coverage (MC/DC)

Analyzes: Decisions based on independence pairs

Definition:

- each point of entry and exit is invoked
- each decision has taken all possible outcomes
- *each condition in a decision is shown to independently affect the outcome of that decision*

Independence Pairs

Independence Pairs for condition (A&&B)||C

- All conditions except the one to be tested are fixed
- The decision has to change its outcome when the condition under test is changed

| # | A | B | C | |
|---|---|---|---|---|
| 1 | F | F | F | F |
| 2 | T | F | F | F |
| 3 | F | T | F | F |
| 4 | F | F | T | T |
| 5 | T | T | F | T |
| 6 | T | F | T | T |
| 7 | F | T | T | T |
| 8 | T | T | T | T |

Bridging the MC/DC Gap

```

B::Trace.List
Setup... Goto... Find... Chart Profile MIPS More Less
record run address cycle data symbol
88 static unsigned ComplexWhile(int const a, int const b, int const c, int const d)
{
  stwu r1,-0x30(r1) ; r1,-48(r1)
  mflr r0
  stw r0,0x34(r1) ; r0,52(r1)
  stw r31,0x2C(r1) ; r31,44(r1)
  mr r31,r1
  stw r3,0x18(r31) ; r3,24(r31)
  stw r4,0x1C(r31) ; r4,28(r31)
  stw r5,0x20(r31) ; r5,32(r31)
  stw r6,0x24(r31) ; r6,36(r31)
89 unsigned num_cycles = 0u;
  li r0,0x0
  stw r0,0x8(r31) ; r0,8(r31)
91 while ((!(a > -70) && !(Identity(b) == 39)) || !(c <= -13) || (Identity(d) < 39)) {
  b 0x400009E4
91 while ((!(a > -70) && !(Identity(b) == 39)) || !(c <= -13) || (Identity(d) < 39)) {
  lwz r9,0x18(r31) ; r9,24(r31)
  li r0,-0x45 ; r0,-69
  cmpw cr7,r9,r0
  
```



```

B::COverage.ListFunc
Setup... Goto... List + Add Load... Save... Init
address tree coverage executed
P:40000610--400006C7 P:40000610--4000062B P:4000062C--400006C7 P:400006C8--400016C7 P:400006C8--400006EF P:400006F0--4000079B P:4000079C--4000083F P:40000840--400008F7 P:400008F8--4000099B P:4000099C--40000A53 P:40000A54--40000AE3 P:40000AE4--40000B7F P:40000B80--40000C0F P:40000C10--40000CBF P:40000CC0--40000D5F P:40000D60--40000E07 P:40000E08--40000EC7 P:40000EC8--40000F77 P:40000F78--40001003 P:40001004--4000106F P:40001070--400010E7 P:400010E8--4000115F P:40001160--400011BB P:400011BC--400011FF P:40001200--4000123F
tree
  main
  background
  coverage
    Identity
    ComplexFor
    TestComplexFor
    ComplexDownWhile
    TestComplexDownWhile
    ComplexWhile
    TestComplexWhile
    ComplexIf
    TestComplexIf
    SwitchCase
    TestSwitchCase
    Multiline
    TestMultiline
    NestedExprTrans
    NestedExpr
    TestExprNesting
    TernaryExprTrans
    TernaryExpr
    TestTernaryExpr
    BooleanAssignmentRelExprTrans
    BooleanAssignmentRelExpr
  
```

Interpretation Trace-based Code Coverage:
All independence pairs are successfully tested

Bridging the MC/DC Gap

The screenshot shows a debugger window titled "[B::List P:0x40000840 /COV]". The window has a menu bar with options: Step, Over, Diverge, Return, Up, Go, Break, Mode, and Find. Below the menu bar is a table with columns: decisions, addr/line, code, label, mnemonic, and comment. The table contains several rows of assembly code with associated addresses and comments. The code is a while loop that checks conditions on Identity(a), Identity(b), and Identity(c). The decisions column shows the execution path taken for each instruction, with arrows indicating the flow of execution.

| decisions | addr/line | code | label | mnemonic | comment |
|-----------|-------------|----------|-------|----------|--|
| | 69 | while | | | (((!Identity(a) >= -45) && Identity(b)) && Identity(c)) d); |
| | SF:40000884 | 807F0018 | | lwz | r3,0x18(r31) ; r3,24(r31) |
| | SF:40000888 | 4BFFFE41 | | bl | 0x400006C8 ; Identity |
| | SF:4000088C | 7C691B78 | | mr | r9,r3 |
| | SF:40000890 | 3800FFD3 | | li | r0,-0x2D ; r0,-45 |
| | SF:40000894 | 7F890000 | | cmpw | cr7,r9,r0 |
| →t/f→ | SF:40000898 | 409C002C | | bge | cr7,0x400008C4 ; cr7,0x400008C4 (-) |
| | SF:4000089C | 807F001C | | lwz | r3,0x1C(r31) ; r3,28(r31) |
| | SF:400008A0 | 4BFFFE29 | | bl | 0x400006C8 ; Identity |
| | SF:400008A4 | 7C601B78 | | mr | r0,r3 |
| | SF:400008A8 | 2F800000 | | cmpwi | cr7,r0,0x0 ; cr7,r0,0 |
| →t/f→ | SF:400008AC | 419E0018 | | beq | cr7,0x400008C4 ; cr7,0x400008C4 (-) |
| | SF:400008B0 | 807F0020 | | lwz | r3,0x20(r31) ; r3,32(r31) |
| | SF:400008B4 | 4BFFFE15 | | bl | 0x400006C8 ; Identity |
| | SF:400008B8 | 7C601B78 | | mr | r0,r3 |
| | SF:400008BC | 2F800000 | | cmpwi | cr7,r0,0x0 ; cr7,r0,0 |
| →t/f→ | SF:400008C0 | 409EFFAC | | bne | cr7,0x4000086C ; cr7,0x4000086C (+) |
| | SF:400008C4 | 801F0024 | | lwz | r0,0x24(r31) ; r0,36(r31) |
| | SF:400008C8 | 2F800000 | | cmpwi | cr7,r0,0x0 ; cr7,r0,0 |
| →t/f→ | SF:400008CC | 409EFFA0 | | bne | cr7,0x4000086C ; cr7,0x4000086C (+) |
| | SF:400008D0 | 48000008 | | b | 0x400008D8 |

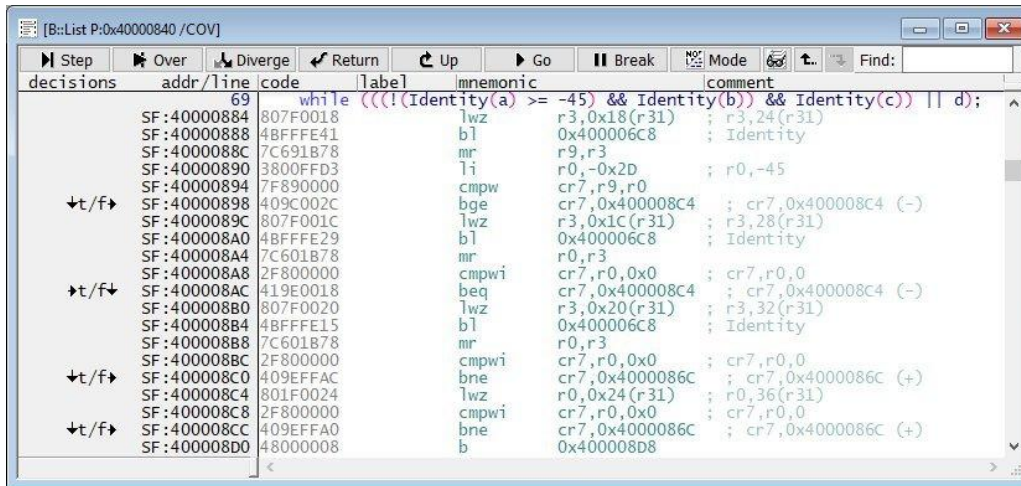
1. To verify the conditions

When analyzing the program execution, it must be clear from the object code whether a condition was TRUE or FALSE.

2. To check the independence pairs

a mapping of decisions to object code is needed.

Bridging the MC/DC Gap – Disable Optimization



```
[B::List P:0x40000840 /COV]
Step Over Diverge Return Up Go Break Mode Find:
decisions addr/line code label mnemonic comment
SF:40000884 69 while (((!(Identity(a) >= -45) && Identity(b)) && Identity(c)) || d);
SF:40000884 807F0018 lwz r3,0x18(r31) ; r3,24(r31)
SF:40000888 4BFFFE41 bl 0x400006C8 ; Identity
SF:4000088C 7C691B78 mr r9,r3
SF:40000890 3800FFD3 li r0,-0x2D ; r0,-45
SF:40000894 7F890000 cmpw cr7,r9,r0
SF:40000898 409C002C bge cr7,0x400008C4 ; cr7,0x400008C4 (-)
SF:4000089C 807F001C lwz r3,0x1C(r31) ; r3,28(r31)
SF:400008A0 4BFFFE29 bl 0x400006C8 ; Identity
SF:400008A4 7C601B78 mr r0,r3
SF:400008A8 2F800000 cmpwi cr7,r0,0x0 ; cr7,r0,0
SF:400008AC 419E0018 beq cr7,0x400008C4 ; cr7,0x400008C4 (-)
SF:400008B0 807F0020 lwz r3,0x20(r31) ; r3,32(r31)
SF:400008B4 4BFFFE15 bl 0x400006C8 ; Identity
SF:400008B8 7C601B78 mr r0,r3
SF:400008BC 2F800000 cmpwi cr7,r0,0x0 ; cr7,r0,0
SF:400008C0 409EFFAC bne cr7,0x4000086C ; cr7,0x4000086C (+)
SF:400008C4 801F0024 lwz r0,0x24(r31) ; r0,36(r31)
SF:400008C8 2F800000 cmpwi cr7,r0,0x0 ; cr7,r0,0
SF:400008CC 409EFFA0 bne cr7,0x4000086C ; cr7,0x4000086C (+)
SF:400008D0 48000008 b 0x400008D8
```

1. When analyzing the program execution, it must be clear from the object code whether a condition was TRUE or FALSE.

For this, each condition must be mapped to the object code by a conditional jump / instruction. This can, however, only be ensured if code optimization is switched off.

Bridging the MC/DC Gap – Perform Static Code Analysis

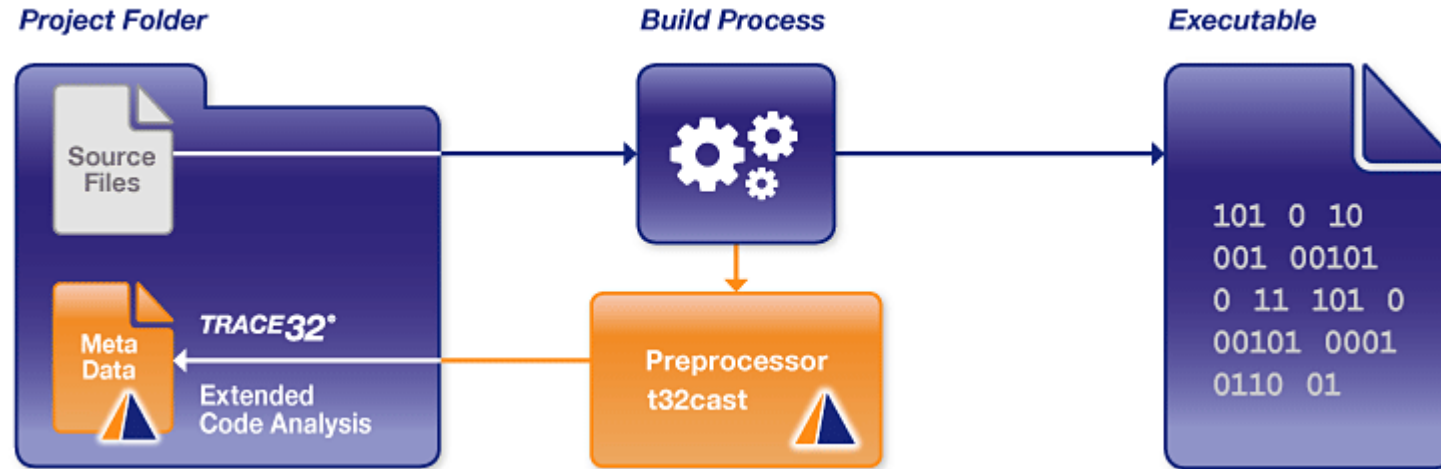
The screenshot shows the Lauterbach Development Tools interface with a list of assembly instructions. The 'decisions' column on the left contains markers for conditional execution: 't/f' (true/false), 't/f' (true/false), 't/f' (true/false), and 't/f' (true/false). The 'addr/line' column shows memory addresses and line numbers. The 'code' column contains assembly instructions. The 'label' column contains labels. The 'mnemonic' column contains the instruction mnemonic. The 'comment' column contains comments.

| decisions | addr/line | code | label | mnemonic | comment |
|-----------|-------------|----------|-------|----------|--|
| | 69 | while | | | ((!(Identity(a) >= -45) && Identity(b)) && Identity(c)) d); |
| | SF:40000884 | 807F0018 | | lwz | r3,0x18(r31) ; r3,24(r31) |
| | SF:40000888 | 4BFFFE41 | | bl | 0x400006C8 ; Identity |
| | SF:4000088C | 7C691B78 | | mr | r9,r3 |
| | SF:40000890 | 3800FFD3 | | li | r0,-0x2D ; r0,-45 |
| | SF:40000894 | 7F890000 | | cmpw | cr7,r9,r0 |
| t/f | SF:40000898 | 409C002C | | bge | cr7,0x400008C4 ; cr7,0x400008C4 (-) |
| | SF:4000089C | 807F001C | | lwz | r3,0x1C(r31) ; r3,28(r31) |
| | SF:400008A0 | 4BFFFE29 | | bl | 0x400006C8 ; Identity |
| | SF:400008A4 | 7C601B78 | | mr | r0,r3 |
| | SF:400008A8 | 2F800000 | | cmpwi | cr7,r0,0x0 ; cr7,r0,0 |
| t/f | SF:400008AC | 419E0018 | | beq | cr7,0x400008C4 ; cr7,0x400008C4 (-) |
| | SF:400008B0 | 807F0020 | | lwz | r3,0x20(r31) ; r3,32(r31) |
| | SF:400008B4 | 4BFFFE15 | | bl | 0x400006C8 ; Identity |
| | SF:400008B8 | 7C601B78 | | mr | r0,r3 |
| | SF:400008BC | 2F800000 | | cmpwi | cr7,r0,0x0 ; cr7,r0,0 |
| t/f | SF:400008C0 | 409EFFAC | | bne | cr7,0x4000086C ; cr7,0x4000086C (+) |
| | SF:400008C4 | 801F0024 | | lwz | r0,0x24(r31) ; r0,36(r31) |
| | SF:400008C8 | 2F800000 | | cmpwi | cr7,r0,0x0 ; cr7,r0,0 |
| t/f | SF:400008CC | 409EFAA0 | | bne | cr7,0x4000086C ; cr7,0x4000086C (+) |
| | SF:400008D0 | 48000008 | | b | 0x400008D8 |

2. Mapping of decisions to object code is needed

Answers the question of whether Branch Taken/Instruction Executed means TRUE or FALSE

Modified Condition/Decision Coverage (MC/DC)



Extended code analysis:

Mapping between a decision statement and its conditions to the conditional branches/instructions in the object code

Modified Condition/Decision Coverage (MC/DC)

[B::List P:0x40000840 / COV]

StepOverDivergeReturnUpGoBreakModeFind:coverage.c

decisionstruefalsecoverageaddr/linecodelabelmnemoniccomment

111mc/dc69while (((!(Identity(a) >= -45) && Identity(b)) && Identity(c)) || d);

okSF:40000884807F0018lwzr3,0x18(r31); r3,24(r31)

okSF:400008884BFFFE41bl0x400006C8; Identity

okSF:4000088C7C691B78mr r9,r3

okSF:400008903800FFD3li r0,-0x2D; r0,-45

okSF:400008947F890000cmpw cr7,r9,r0

okSF:40000898409C002Cbge cr7,0x400008C4; cr7,0x400008C4 (-)

okSF:4000089C807F001Clwz r3,0x1C(r31); r3,28(r31)

okSF:400008A04BFFFE29bl 0x400006C8; Identity

okSF:400008A47C601B78mr r0,r3

okSF:400008A82F800000cmpwi cr7,r0,0x0; cr7,r0,0

okSF:400008AC419E0018

okSF:400008B0807F0020

okSF:400008B44BFFFE15

okSF:400008B87C601B78

okSF:400008BC2F800000

okSF:400008C0409EFAAC

okSF:400008C4801F0024

okSF:400008C82F800000

okSF:400008CC409EFA0

okSF:400008D048000008

B::COVerge.ListFunc

Setup...Goto...ListAddLoad...

addresstree

P:40000610--400006C7main

P:40000610--4000062Bmain

P:4000062C--400006C7background

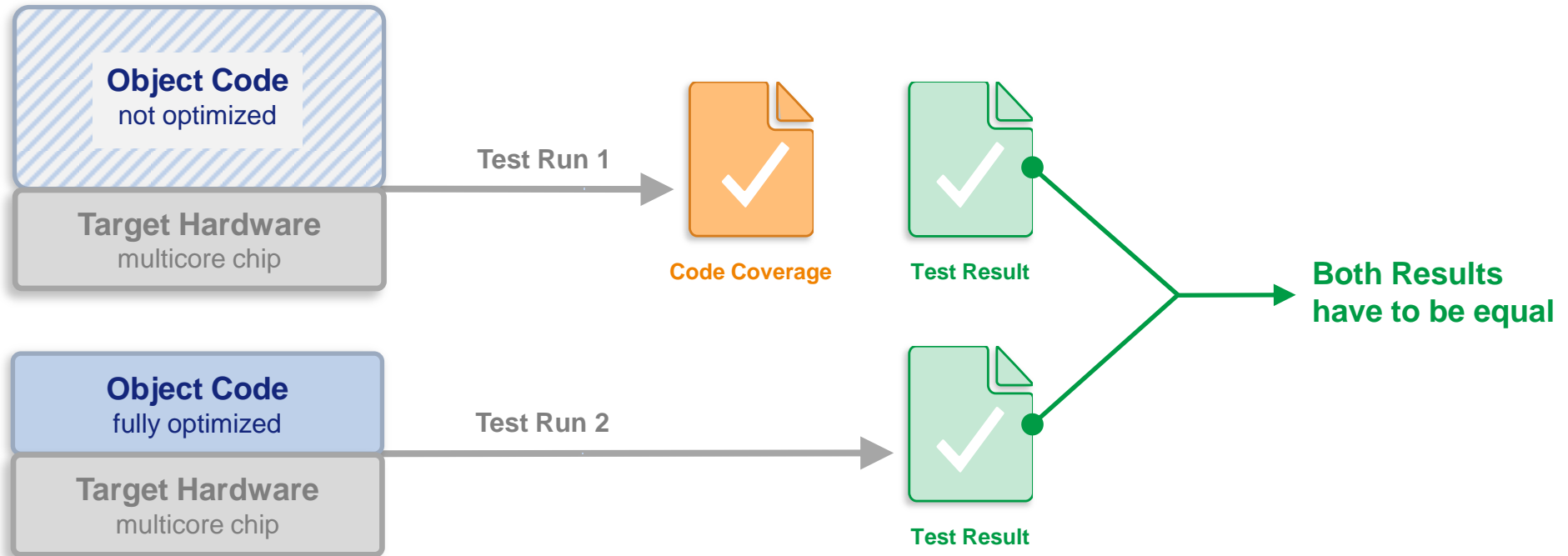
P:400006C8--400016C7coverage

P:400006C8--400006EFIdentity

B::COverage.ListFunc

| Setup... | Goto... | List | + Add | Load... | Save... | Init | coverage | executed | 0% | 50% | 100% |
|----------|---------|------|-------|---------|---------|------|------------|----------|----|-----|------|
| | | | | | | | incomplete | 0.000% | | | |
| | | | | | | | incomplete | 0.000% | | | |
| | | | | | | | incomplete | 0.000% | | | |
| | | | | | | | incomplete | 94.805% | | | |
| | | | | | | | stmt+mc/dc | 100.000% | | | |
| | | | | | | | stmt+mc/dc | 100.000% | | | |
| | | | | | | | stmt+mc/dc | 100.000% | | | |
| | | | | | | | stmt+mc/dc | 100.000% | | | |
| | | | | | | | stmt+mc/dc | 100.000% | | | |
| | | | | | | | stmt+mc/dc | 100.000% | | | |
| | | | | | | | stmt+mc/dc | 100.000% | | | |
| | | | | | | | stmt+mc/dc | 100.000% | | | |
| | | | | | | | stmt+mc/dc | 100.000% | | | |
| | | | | | | | stmt+mc/dc | 100.000% | | | |
| | | | | | | | stmt+mc/dc | 100.000% | | | |
| | | | | | | | stmt+mc/dc | 100.000% | | | |
| | | | | | | | stmt+mc/dc | 100.000% | | | |

Modified Condition/Decision Coverage (MC/DC)



Agenda

- Introduction
- Modern Automotive Chips
- Making Use of the Trace Capability
- Measuring Code Coverage without Instrumentation
- **Conclusion**

Summary

| Metric | Code Coverage | Extended Code Analysis |
|--------------------|-----------------------|--|
| Statement Coverage | On final code | — |
| Branch Coverage | Reduced optimization | Mapping of decisions to object code |
| MC/DC | On not-optimized code | Mapping of decisions/conditions to object code |
| Function Coverage | On final code | — |
| Call Coverage | On final code | To tag inline function, function pointer, call-less function |

Advantages

- Trace-based code coverage is applicable to all methods of ISO 26262
- Trace-based code coverage can be performed at any stage of the project
- Trace-based code coverage allows testing at handwritten and autogenerated code
- Trace tools already employed can be used at no extra cost
- Long-term testing by streaming trace to host computer
- Testing with cross-compiler used to generate final code
- Virtual verification or hardware-based testing

Further Improvements

Trace-based Code Coverage

- Live coverage for all metrics
- Improved display for MC/DC
- ADA support

Compiler

- Special optimizations adapted to Code Coverage requirements
- Source code analysis results for method in use included in debug information

QUESTIONS?

THANK YOU!

Andrea Martin • andrea.martin@lauterbach.com • 04 / July / 2019

www.lauterbach.com

