Non-Intrusive Online Observation of Multicore Processors

ARVI Meeting Tallinn, 10.12.2015 Alexander Weiss

AGENDA

- Multicore observation requirements
- State-of-the-art observation methodologies
  - Software instrumentation
  - Embedded trace
- Online observation

AGENDA

- *Multicore observation requirements*
- State-of-the-art observation methodologies
  - Software instrumentation
  - Embedded trace
- Online observation

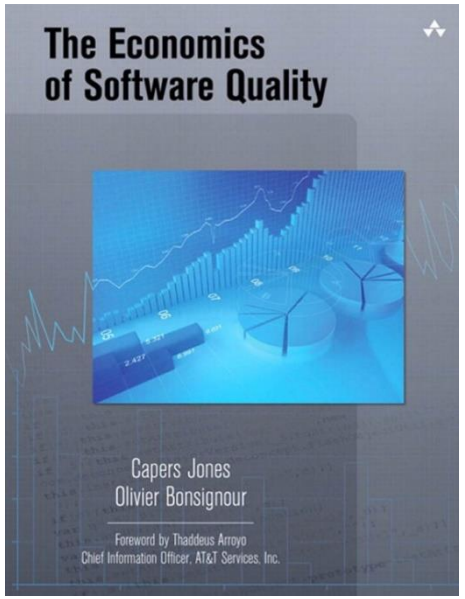
PROBLEM DESCRIPTION

U.S. SOFTWARE QUALITY AVERAGES CIRCA 2012

	Commercial Software	**Embedded Software**
Defect Potentials (Defects/FP)	5	**5,5**
Defect Removal Efficiency	90%	**95%**
Delivered Defects (Defects/FP)	0,5	**0,3**

SOFTWARE SIZE VS DEFECT REMOVAL EFFICIENCY

	10 FP	100 FP	**1000 FP**	**10000 FP**
Defect Potentials (Defects/FP)	2,45	3,68	**5,00**	**7,60**
Defect Removal Efficiency	92%	90%	**85%**	**78%**
Delivered Defects (Defects/FP)	0,2	0,37	**0,75**	**1,67**

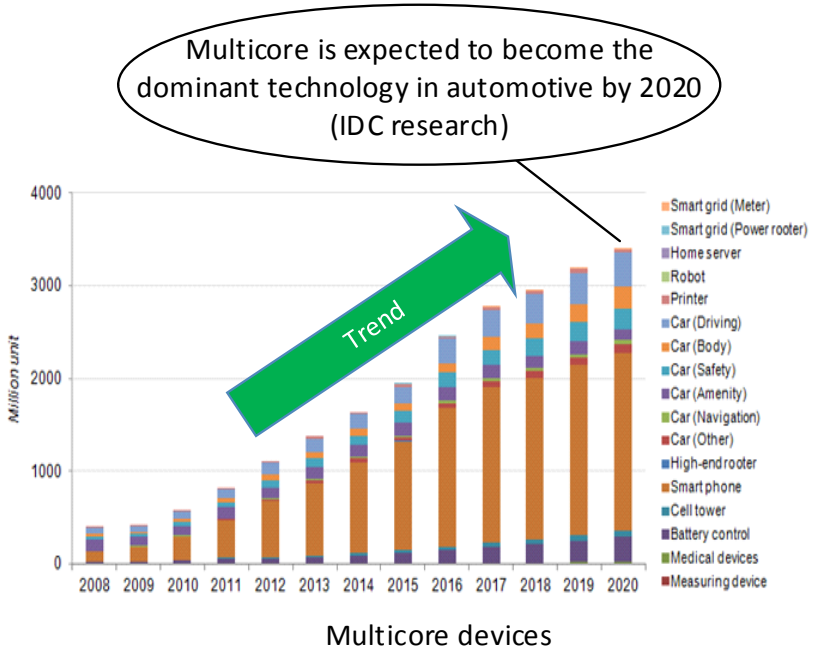


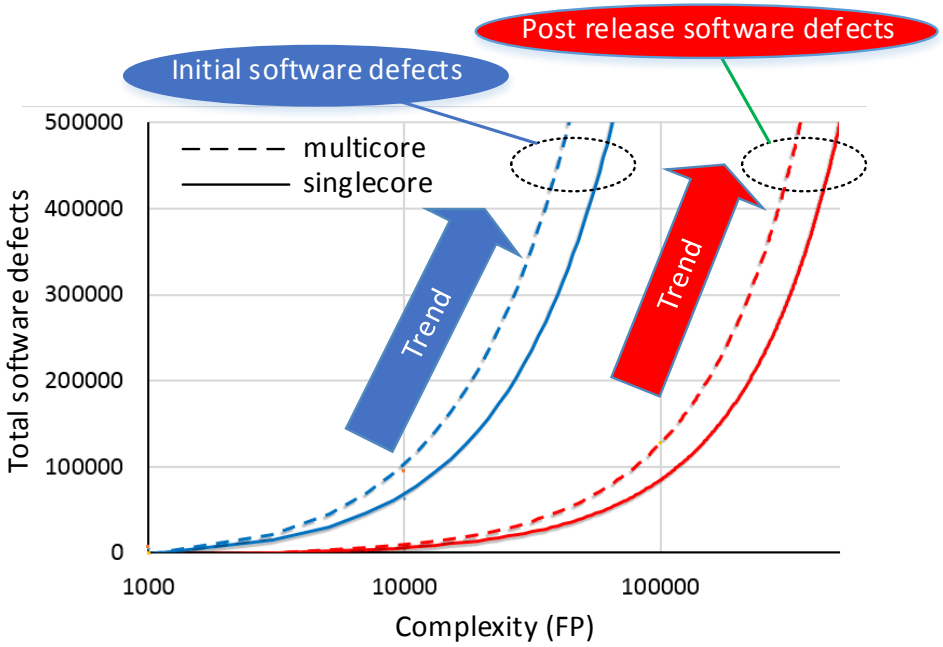
Source: C. Jones, O. Bonsignour, "The Economics of Software Quality", Pearson Education, Inc. 2012

- Data collected from 1984 through 2012
- About 675 companies (150 clients in Fortune 500 set)
- About 13,500 total projects

1 **FP** (Function Point): ~160 LOC (C language), ~64 LOC (ADA), ~32 LOC (C++)

PROBLEM DESCRIPTION





Costs of failure reproduction

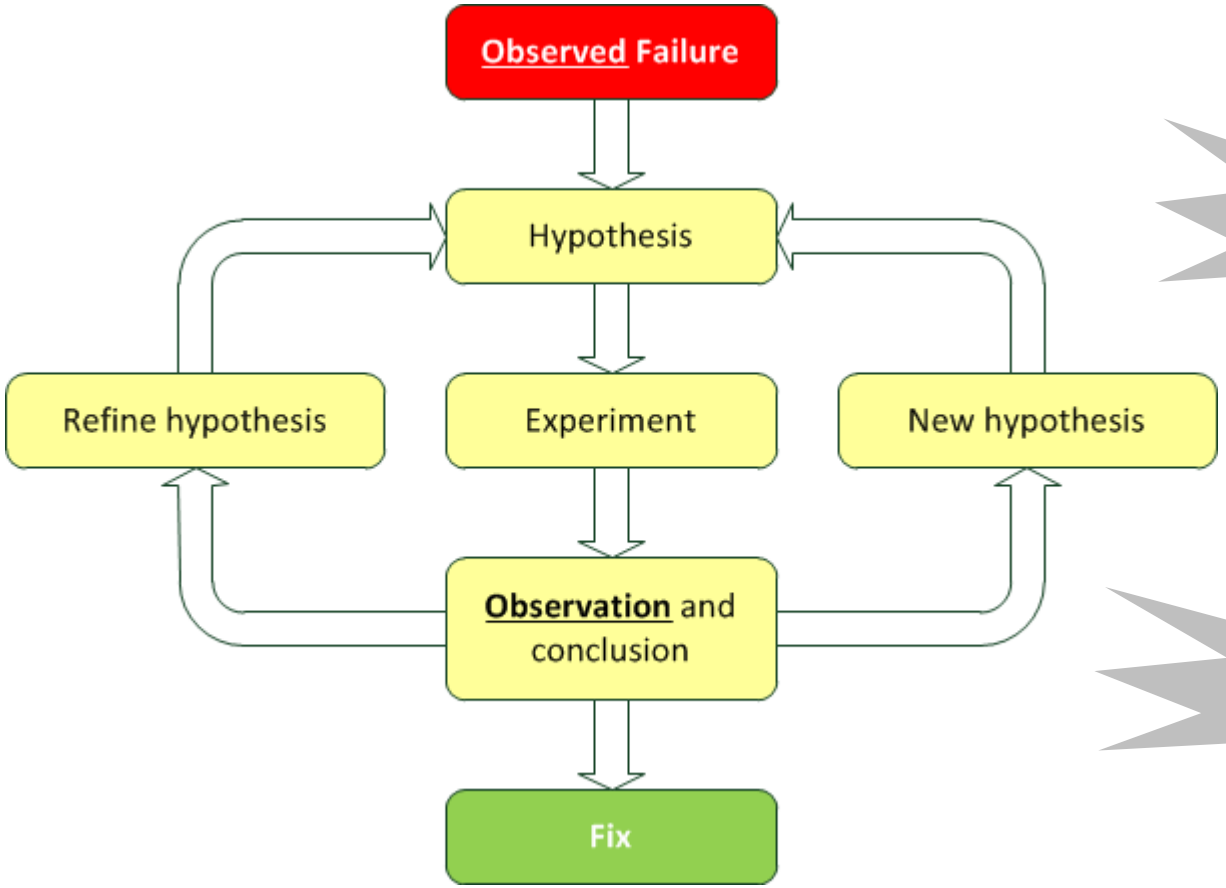
Cost components	Costs
Hourly labor rate	200,00 €*
Number of personnel	4*
Rent	41,67 €*
Maintenance Hardware	33,00 €*
Energy consumption	12,00 €*
Other	30,00 €*
Total cost per hour	916,67 €*

Reproducibility [%]	100	75,108	50	35,008	21,04	10,004	5	2
Tests per Reproduction	0,5	1,66	3,32	5,34	9,75	21,76	44,9	113,97
Number of Reproductions	5	5	5	5	5	5	5	5
Total Quantity of Tests	5	8,3	16,6	26,7	48,75	108,8	224,5	569,85
Time per Test [h]	1	1	1	1	1	1	1	1
Total Time	5	8,3	16,6	26,7	48,75	108,8	224,5	569,85
Hourly Rate RIG [€]	917*	917*	917*	917*	917*	917*	917*	917*
Cost per Failure [10^3 €]	2,29*	7,61*	15,22*	24,48*	44,70*	99,77*	205,87*	522,55*

(B. Hanke and F. Schulz, "Assessment of multi-core integration infrastructure for military avionics," University of German Armed Forces Munich, Manching, Munich, 2014)

PROBLEM DESCRIPTION

Scientific debugging



Brain power

Observability

PROBLEM DESCRIPTION

24.07.13 Entwickler fühlen sich beim Multicore-Debugging im Stich gelassen

ELEKTRONIK PRAXIS

Fraunhofer-Umfrage

Entwickler fühlen sich beim Multicore-Debugging im Stich gelassen

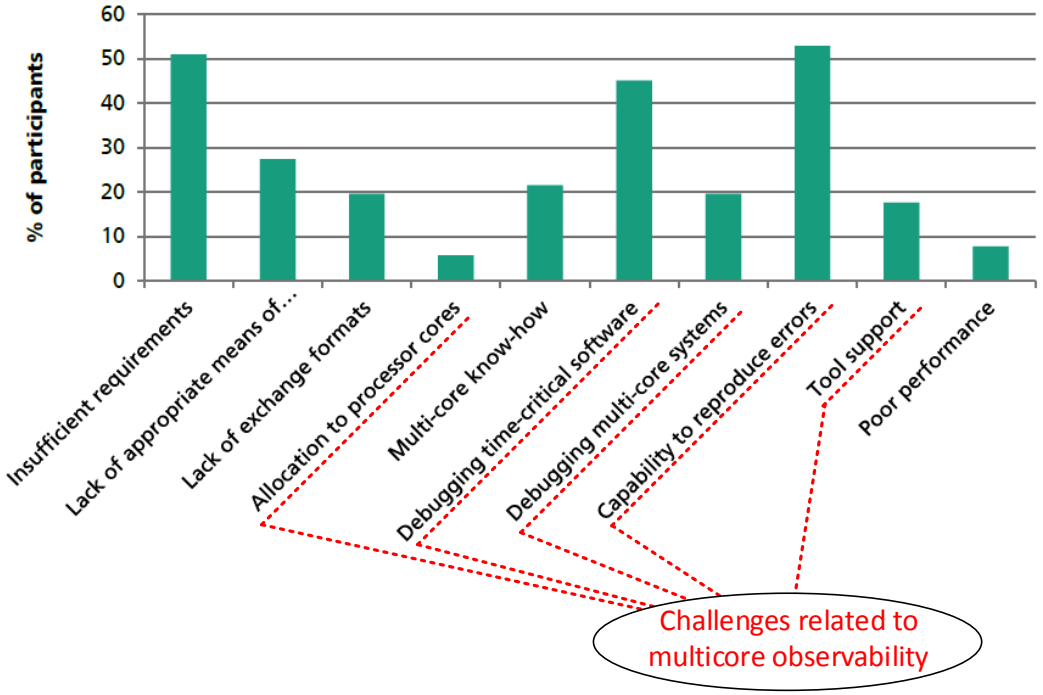
19.07.13 | Redakteur: Franz Graser

Beim Debugging von Software für Embedded-Multicore-Prozessoren wünschen sich viele Softwareentwickler bessere Unterstützung. (Bild: Clipdealer)

Eine Umfrage unter Softwareentwicklern zeigt gravierende Probleme bei der Multicore-Programmierung auf: Beklagt wurden unter anderem Defizite bei Werkzeugen für Debugging und Test.

Die Umfrage wurde vom Münchner Fraunhofer-Institut für Eingebettete Systeme und Kommunikationstechnik vorgenommen. Die 51 per Online-Fragebogen interviewten Teilnehmer definierten Performanz, deterministisches Verhalten und Echtzeit als die wichtigsten Anforderungen an Software für eingebettete Systeme. Dabei spielt sich Echtzeit in der Regel zwischen 10µs und 10ms ab.

What are the major challenges you face in developing software for embedded multicore systems?

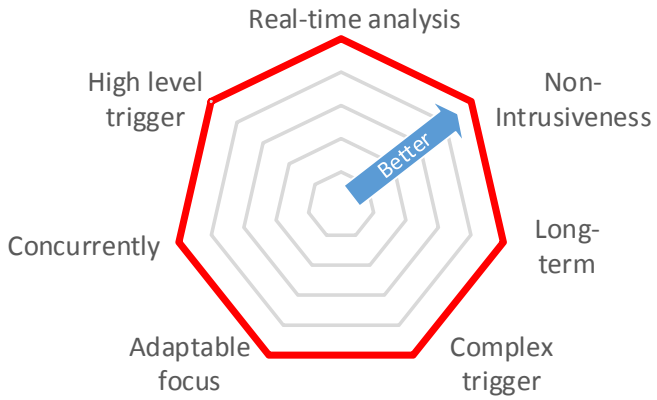


Source: Fraunhofer ESK: Trends and Challenges in Developing Software for Embedded Systems, 2013

REQUIREMENTS

Multicore debugging requirements

- Root causes of sporadic failures must be identified quicker and more efficiently
- New multicore observation method which needs to:
  - support **analysis of trace data in real-time**
  - be **non-intrusive**
  - trace processors **for many hours** not just a few seconds
  - be capable of **triggering on complex conditions** which may have a high temporal spread
  - **adaptable** in terms of **observation focus** without changing the software
  - be capable to **observe multiple failures concurrently** due to low occurrence rate of single failure
  - support **high level description of trigger conditions** and post-processing actions of trace extracts
- Multicore processors useable for mission avionic must support observation



[Franz Münz: *Sporadic failures – Implicationd for Multicore*, Multicore Application Debugging Workshop 2013]

AGENDA

- Multicore observation requirements
- *State-of-the-art observation methodologies*
  - Software instrumentation
  - Embedded trace
- Online observation

STATE OF THE ART: SOFTWARE-INSTRUMENTATION

Original source code

```
void foo()
{
  bool found=false;
  for (int i=0; (i<100) && (!found); ++i)

  {
    if (i==50) break;


    if (i==20) found=true;


    if (i==30) found=true;



  }
  printf("foo\n");

}
```

Instrumented source code

```
char inst[15];
void foo()
{
   bool found=false;
   for (int i=0;((i<100)?inst[0]=1:inst[1]=1,0) &&
     ((!found)?inst[2]=1:inst[3]=1,0); ++i)
   {
     if ((i==50?inst[4]=1:inst[5]=1,0))
     { inst[6]=1; break;
     }
     if ((i==20?inst[7]=1:inst[8]=1,0))
     { inst[9]=1; found=true;
     }
     if ((i==30?inst[10]=1:inst[11]=1,0))
     { inst[12]=1; found=true;
     }
     inst[13]=1;
   }
   printf("foo\n");
   inst[14]=1;
}
```

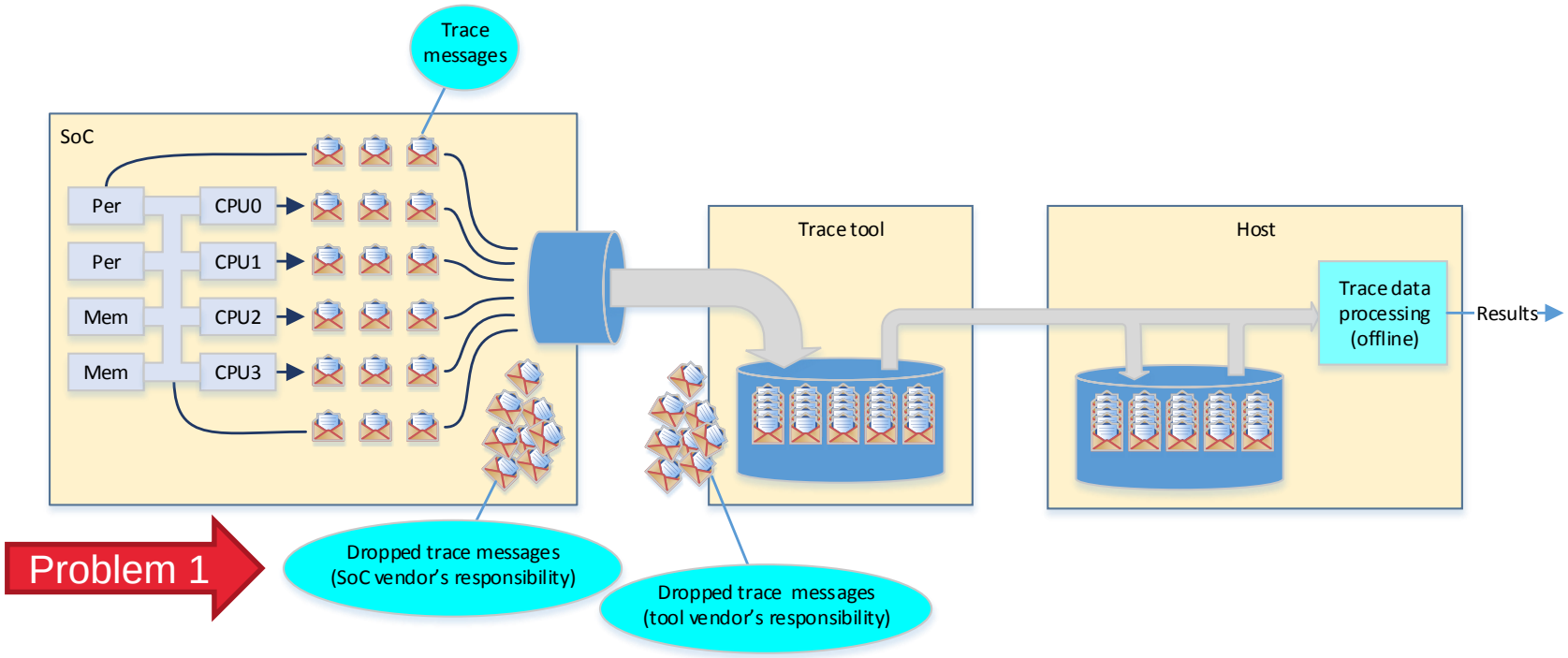
[*Squish Coco - Code Coverage*, froglogic GmbH, 2012: Figure 2.1 and Figure 2.8]

+ Easy to implement
+ Wide tool support

– Intrusive
– Resources
(application blow up, execution slow down)
– Limited observability

STATE OF THE ART: EMBEDDED TRACE

Recording of trace messages and offline analysis

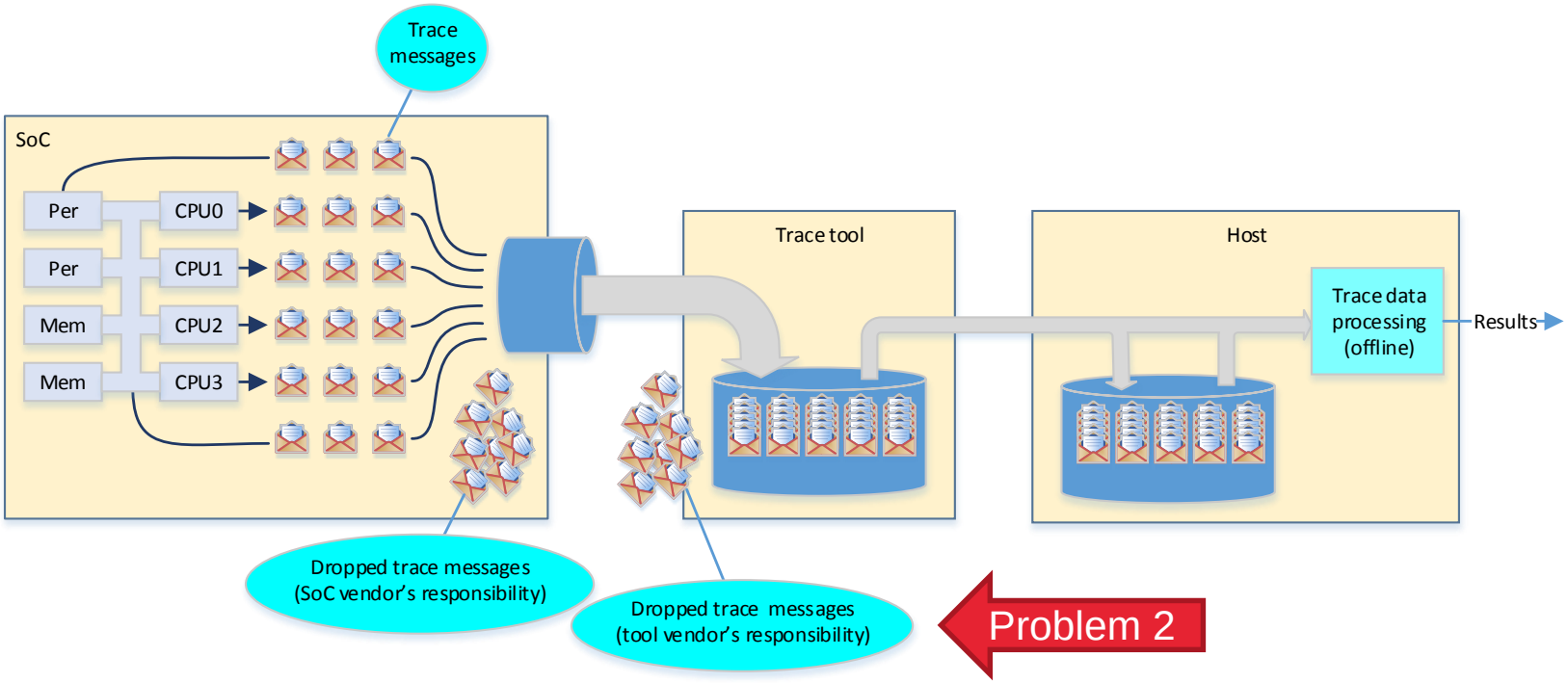


Limitations

- Trace-Messages inside the SoC >> Bandwidth of the trace port
- Bandwidth of the trace port >> Trace messages processing bandwidth
- Latency between SoC internal events and processing result
- Limited SoC internal trace trigger

STATE OF THE ART: EMBEDDED TRACE

Recording of trace messages and offline analysis



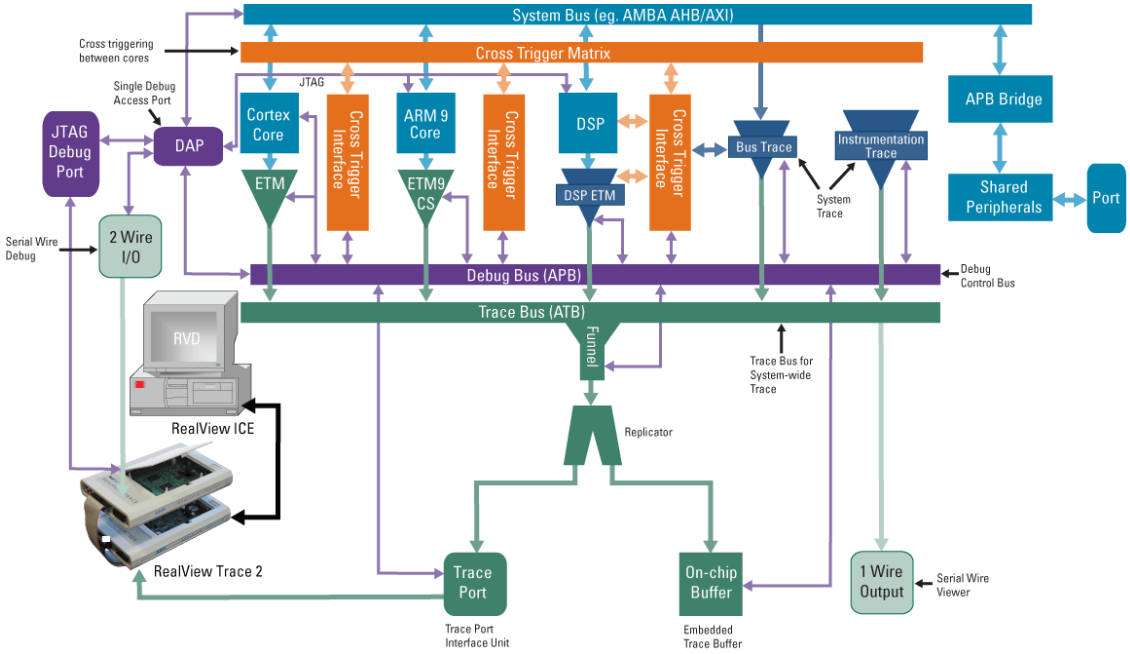
Limitations

- Trace-Messages inside the SoC >> Bandwidth of the trace port
- Bandwidth of the trace port >> Trace messages processing bandwidth
- Latency between SoC internal events and processing result
- Limited SoC internal trace trigger

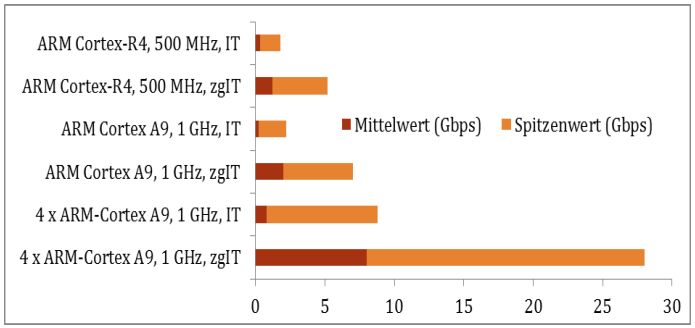
STATE OF THE ART: EMBEDDED TRACE

Implementations

Source: Green Hills



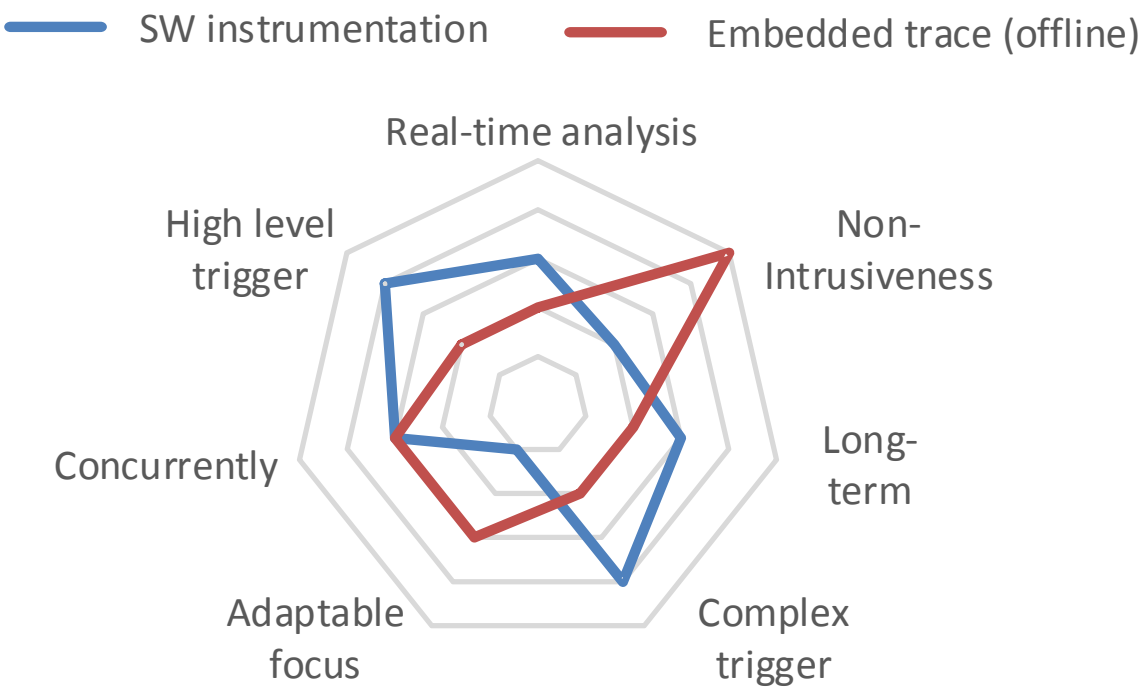
Source: ARM



Source: Lauterbach

STATE OF THE ART: EMBEDDED TRACE

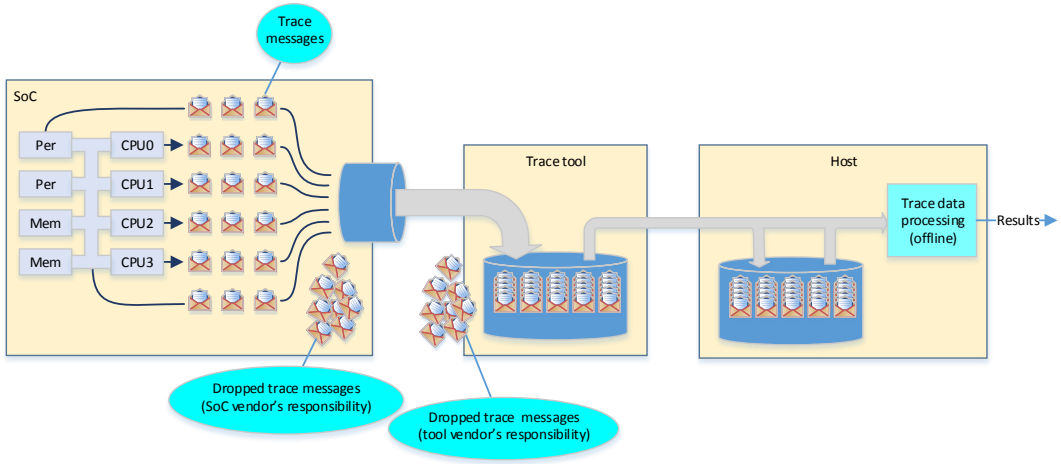
Software instrumentation vs. (offline) embedded trace



AGENDA

- Multicore observation requirements
- State-of-the-art observation methodologies
  - Software instrumentation
  - Embedded trace
- *Online observation*

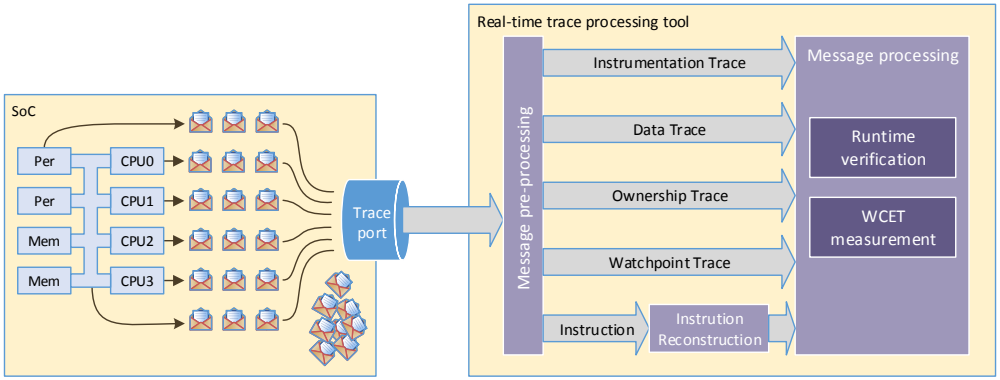
ONLINE OBSERVATION



- Offline observation

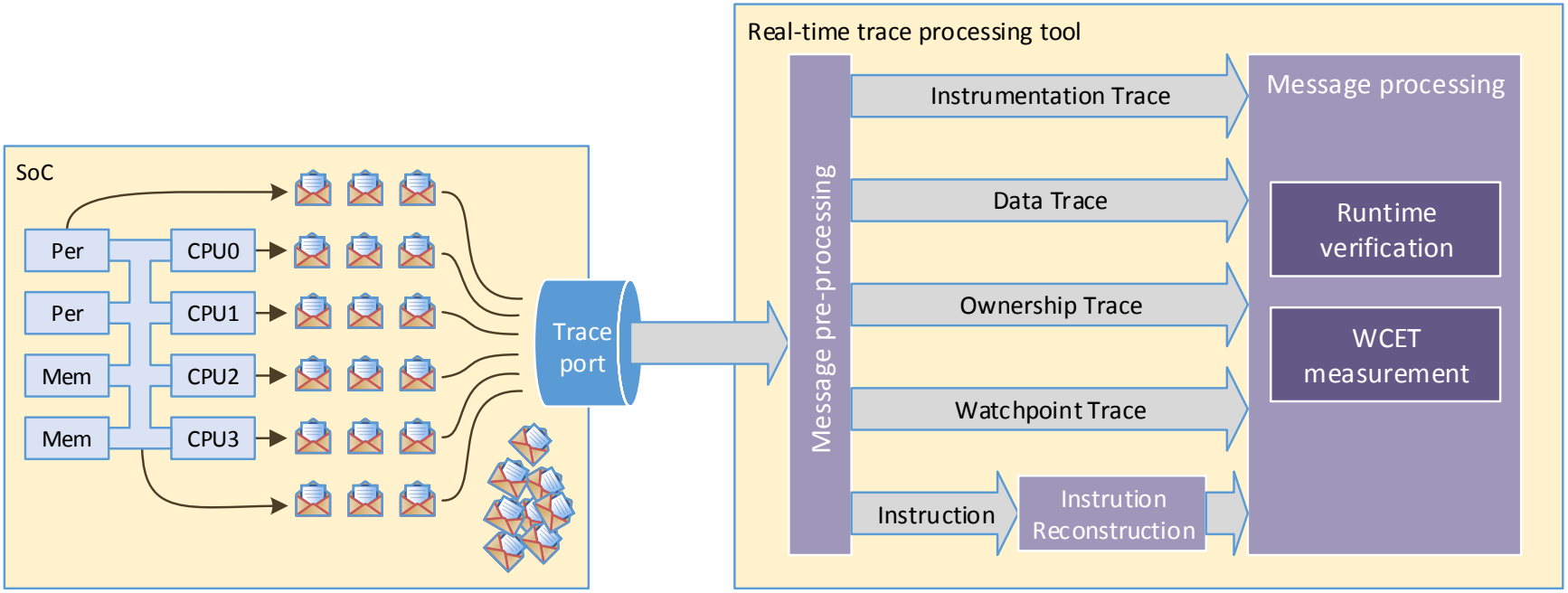
CONIRAS

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung



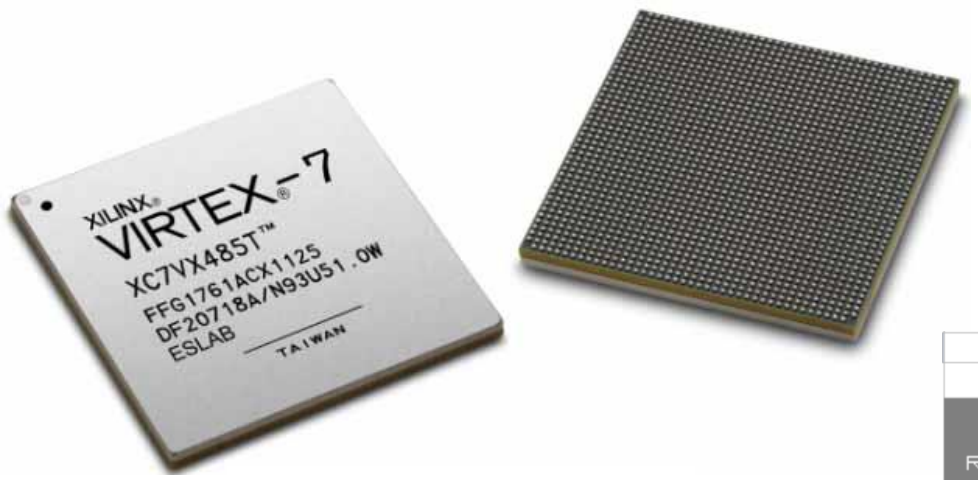
- Combination of
  - Online observation
  - Runtime verification
  - WCET measurement

ONLINE OBSERVATION



- Trace bandwidth ≤ processing bandwidth
- µs latency between SoC activity and processing results
- Continuous observation
- Autonomous observation

ONLINE OBSERVATION



Source: Xilinx

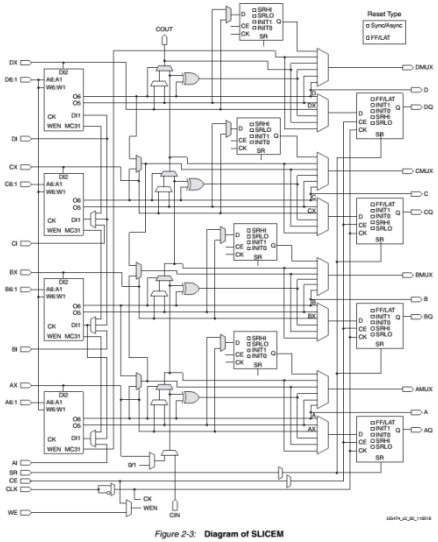


Figure 2-3: Diagram of SLICEM

[*7 Series FPGAs Configurable Logic Block User Guide*, Xilinx 2015]

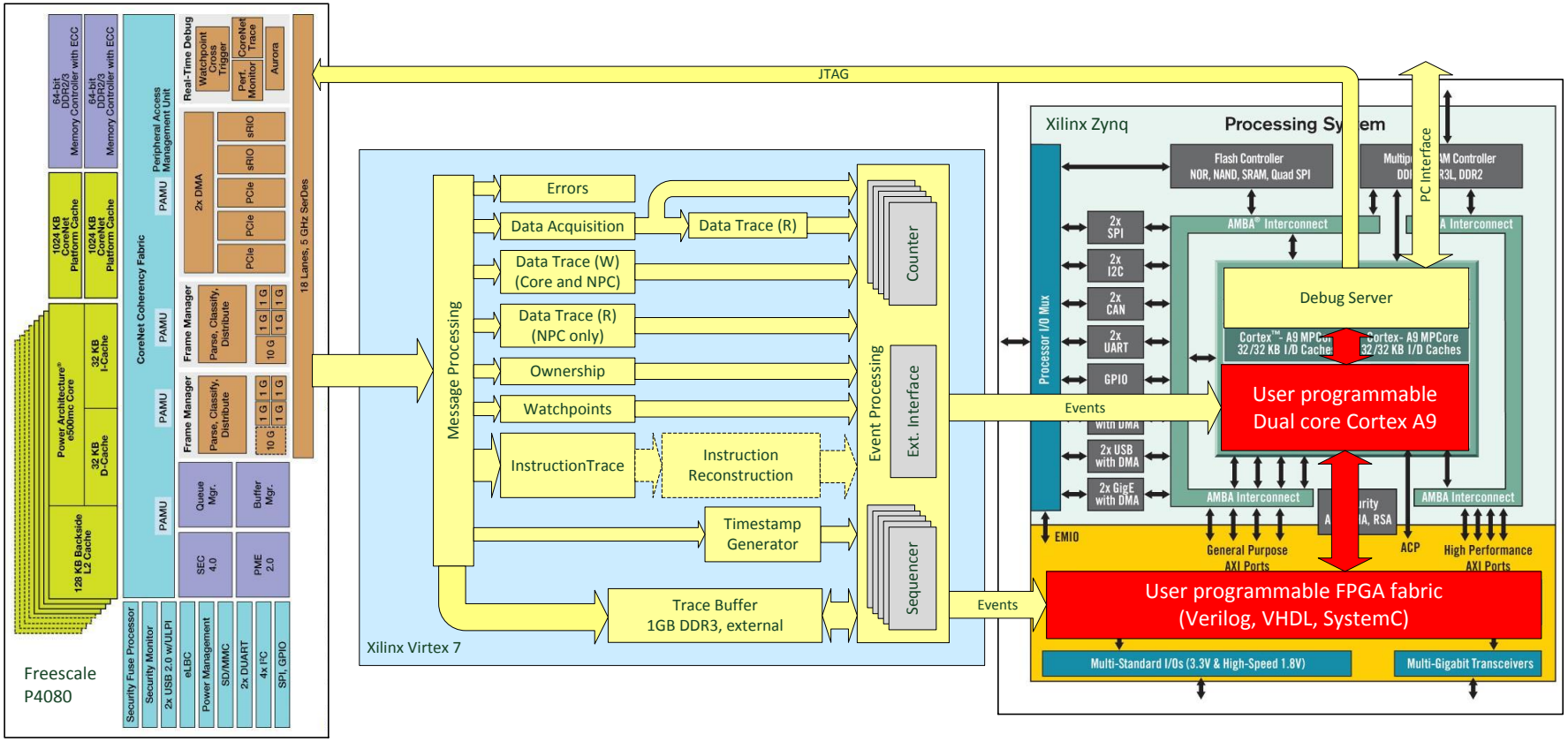
	Part Number	XC7V585T
	syPath™ Cost Reduction Solutions(1)	XCE7V585T
Logic Resources	Slices	91,050
	Logic Cells	582,720
	CLB Flip-Flops	728,400
Memory Resources	Maximum Distributed RAM (Kb)	6,938
	ck RAM/FIFO w/ ECC (36 Kb each)	795
	Total Block RAM (Kb)	28,620
Clocking	CMTs (1 MMCM + 1 PLL)	18
I/O Resources	Maximum Single-Ended I/O	850
	Maximum Differential I/O Pairs	408
Embedded IP Resources	DSP48E1 Slices	1,260
	PCI Express Gen2	3
	PCI Express Gen3	—
	Agile Mixed Signal (AMS) / XADC	1
	Configuration AES / HMAC Blocks	1
	GTX 12.5 Gb/s Transceivers(2)	36
	GTH 13.1 Gb/s Transceivers(3)	—
		—

[*Xilinx 7 Series Feature Overview*, Xilinx 2012]

ONLINE OBSERVATION

Online observation implementation example

(Airbus ARAMiS demonstrator: Freescale P4080 (8 cores e500mc @1.3 GHz))



ONLINE OBSERVATION

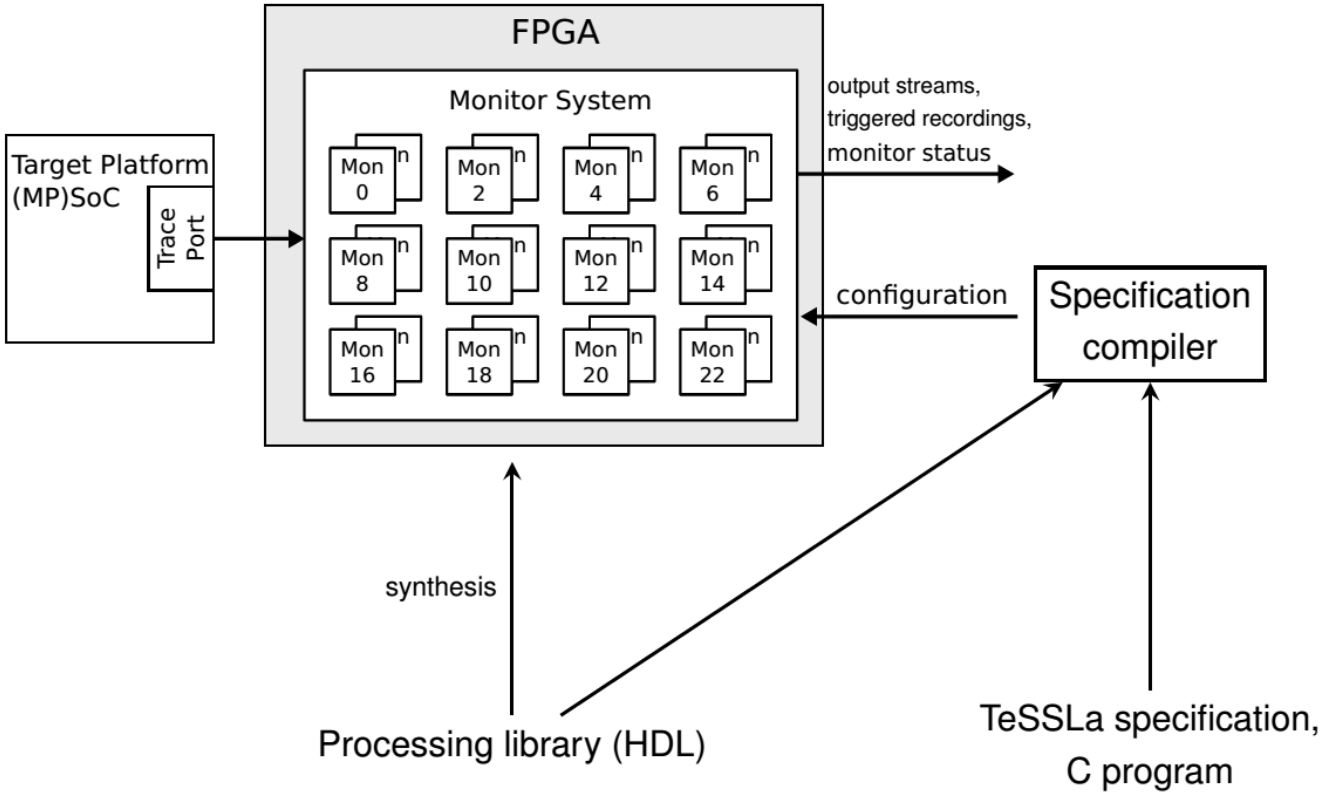
Online processable trace messages

- Program flow (non cycle accurate / cycle accurate for instruction or basic block)
  - Exceptions (interrupts…)
  - Ownership / Context ID
  - Indirect branches
  - Direct branches (high effort for online reconstruction)
  - ➢ Arbitrary amount of watchpoints
- Data access (address, direction, value -> high bandwidth)
- Hardware supported instrumentation
  - guaranteed transactions
  - non-guaranteed transactions
- Vendor defined messages

Trace message content depends on target architecture.

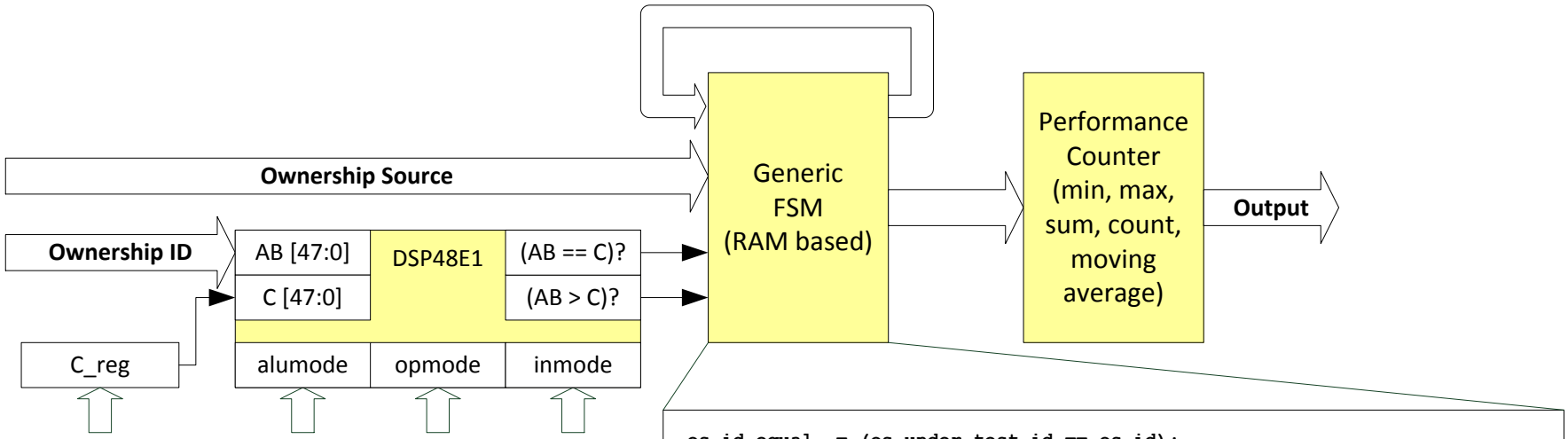
- ➢ IEEE-ISTO, "The Nexus 5001 Forum - Standard for a Global Embedded Processor Debug Interface," IEEE-ISTO 5001TM-2012, Jun. 2012
- ➢ various CoreSight Technical Reference Manuals, ARM
- ➢ Intel® 64 and IA-32 Architectures Software Developer's Manual
- ➢ Infineon MCDS (NDA required)
- ➢ MIPI Alliance Specification for System Trace Protocol

ONLINE OBSERVATION



Runtime verification implementation example

(ownership observation on multiple cores)

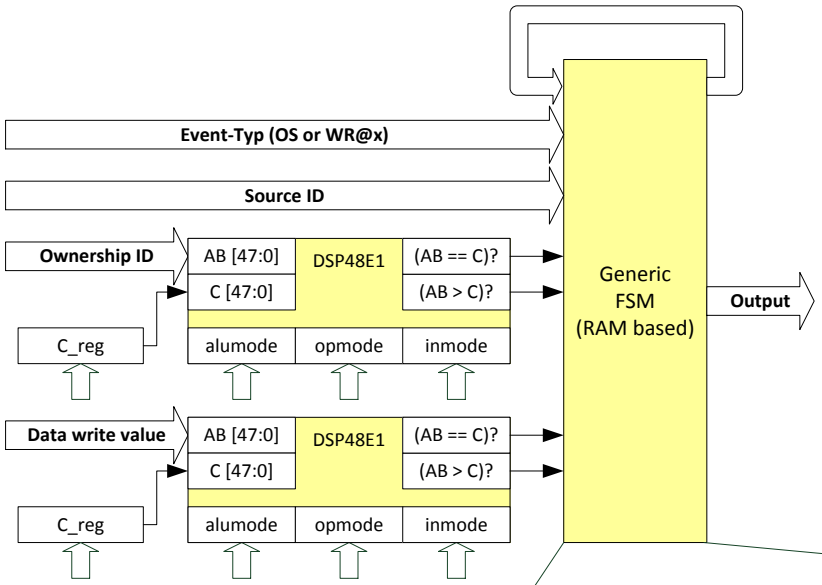



```
os_id_equal  = (os_under_test_id == os_id);
os_src_equal = (last_os_src == os_src);

if (event_valid) begin
  case ({os_id_equal, os_src_equal})
    2'b00: begin     // different os start at different source: don't care
    end
    2'b01: begin    // different os start at same source
      os_active <= 0;
    end
    2'b10: begin    // equal os starts at different source
      os_active <= 1;
      last_os_src <= os_src;
    end
    2'b11: begin    // equal os starts at same source
      os_active <= 1;
    end
  endcase
end
```

Runtime verification implementation example

(Check if x < y before Task z was running on cores 0 to 3)




```
input [3:0] src_id;
input [1:0] bctrl;              // Bit0: OS message, Bit1: Data write @0x10 message
input [1:0] data_value_equal;   // Bit0: OS_ID, Bit1:
input [1:0] data_value_compare; // Bit0: OS_ID, Bit1:

reg [1:0] state;
reg error;
reg success;

// input selector
// addr0:      equal_in[0]      -> os_id_equal
// addr1:      compare_in[0]    -> x
// addr2:      equal_in[1]      -> x
// addr3:      compare_in[1]    -> data_value > 0x100
// addr[6:4]:  src_in[3:1]      -> core 0..7
// addr7:      bctrl_in[0]      -> 0: os message, 1: data write message
// addr[9:8]:  local_fb_in [1:0]  -> state

if (event_valid) begin
  if (rst) begin
    state  <= 0;
    error  <= 0;
    success<= 0;
  end
  else begin
    case (state)
      2'h0: begin // wait for OS1
        if ((src_id == 0) ||(src_id == 1) ||(src_id == 2) ||(src_id == 3)) && os_id_equal && (bctrl_in[0]== 0)) begin
          state <= 2'h1;
        end
        if ((bctrl_in[0] == 1 ) &&  (compare_in[1] == 1) && (src_id == 4)) begin
          state <= 2'h3;
        end
      end
      2'h1: begin // core 4 writes a value > 0x100 to (mapped) address 0x10
        if ((bctrl_in[0] == 1 ) &&  (compare_in[1] == 1) && (src_id == 4)) begin
          state <= 2'h2;
        end
      end
      2'h2: begin // OK
        success <= 1;
      end
      2'h3: begin // FAILURE
        error <= 1;
      end
    endcase
  end
end
```

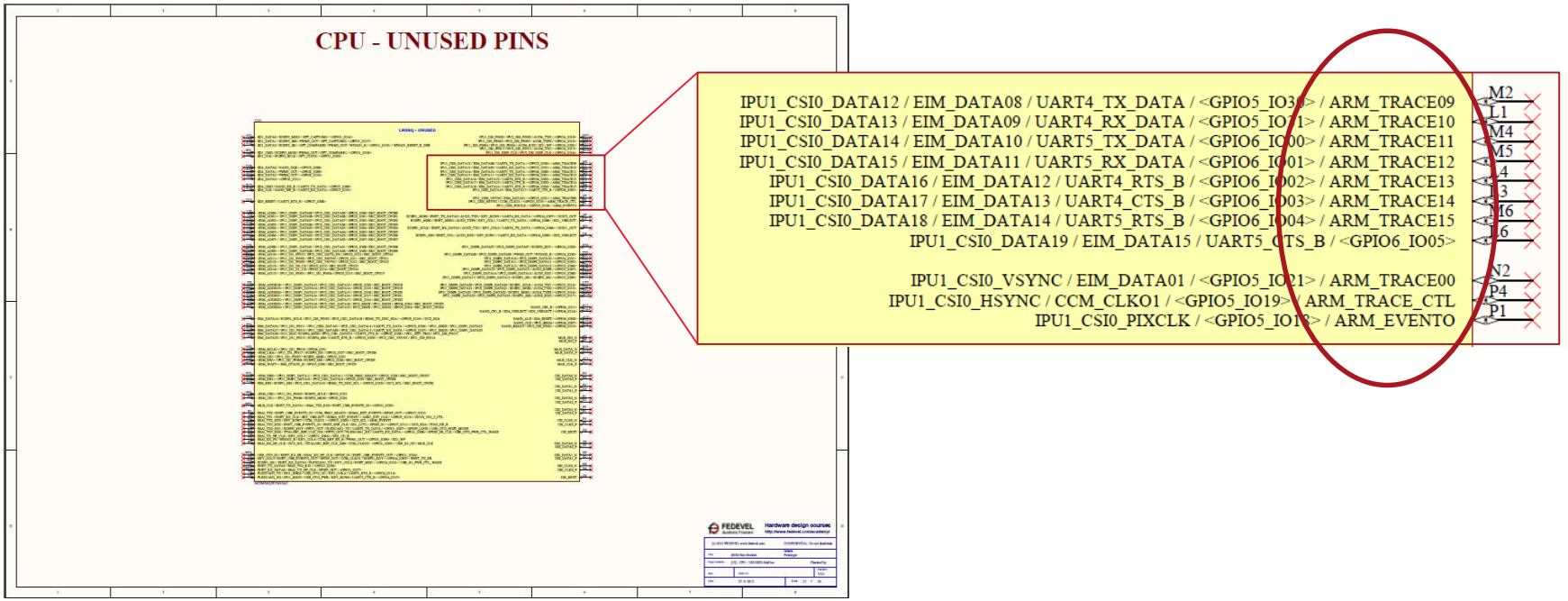
ONLINE OBSERVATION

Design recommendations

- Hardware design:
  - consider trace port pins
  - provide max. trace port bandwidth

Samtec QSH-030-01

AMP 5767096-8



ONLINE OBSERVATION

Design recommendations

- Consider hardware supported instrumentation in software architecture

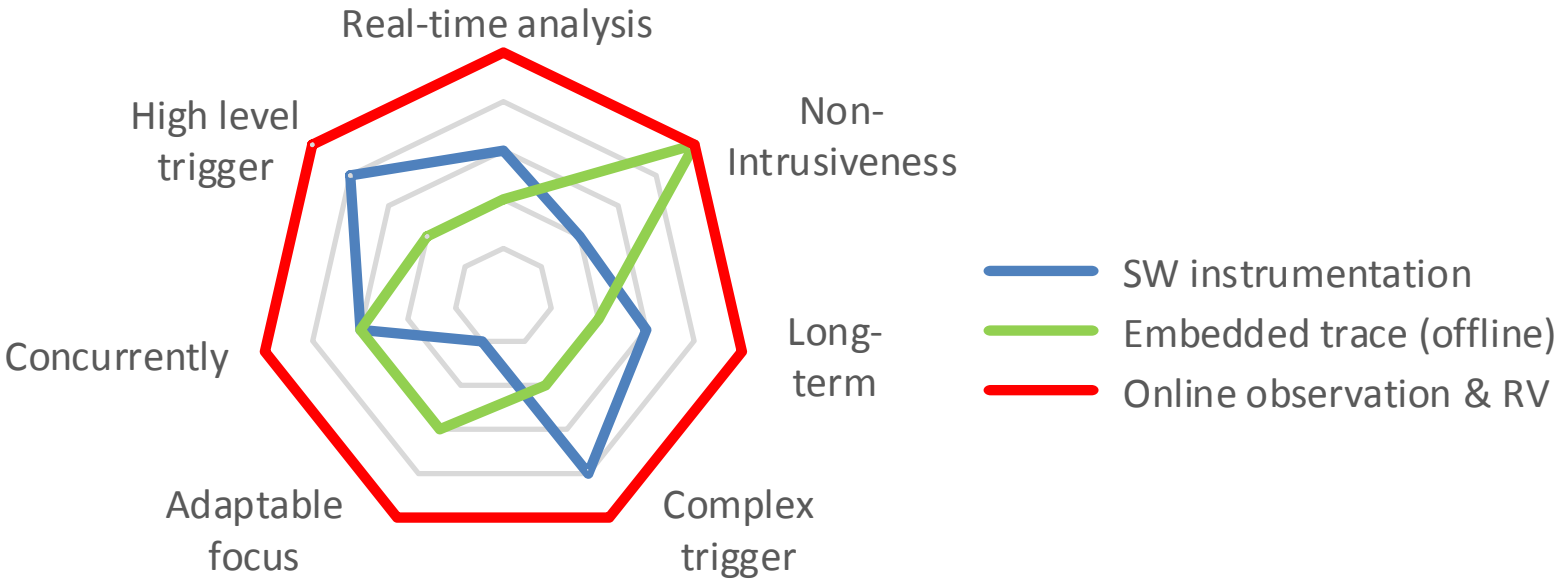
```
void classTest::set_varA(int value)
{          varA = value;
           OnVariableSet(SET_VARA, varA);
}
int classTest::get_varA(void)
{          OnVariableGet(GET_VARA,varA);
           return varA;
}
```

```
inline void OnVariableSet(unsigned short slot, int value)
{STM_SEND_DATA(stm_master0,slot)=value;}

void OnVariableGet(unsigned short slot, int value)
{STM_SEND_DATA(stm_master0,slot)=value;}
```

```
#define STM_SEND_DATA(m,slot) (*( volatile uint32_t*)((unsigned int)m+slot*256+0x18))
```

ONLINE OBSERVATION



The combination of **Online observation** and **Runtime verification** will be a key factor for the exploitation of the multicore potentials.

THANK YOU

Contact:

Accemic GmbH & Co. KG
Franz-Huber-Str. 39
83088 Kiefersfelden

Dr.-Ing. Alexander Weiss
aweiss@accemic.com
+49 8033 6039795

in cooperation with

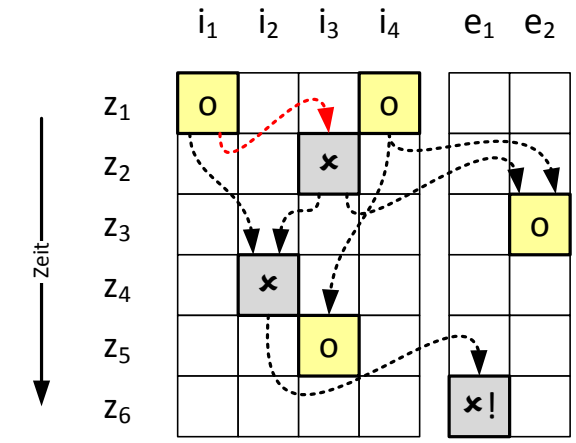
- AbsInt Angewandte Informatik GmbH
- TU Darmstadt
- Universität zu Lübeck

www.coniras.org

CONIRAS

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung



Korrekter Verarbeitungsschritt

Verarbeitungsschritt mit Aktivierung eines Defekts

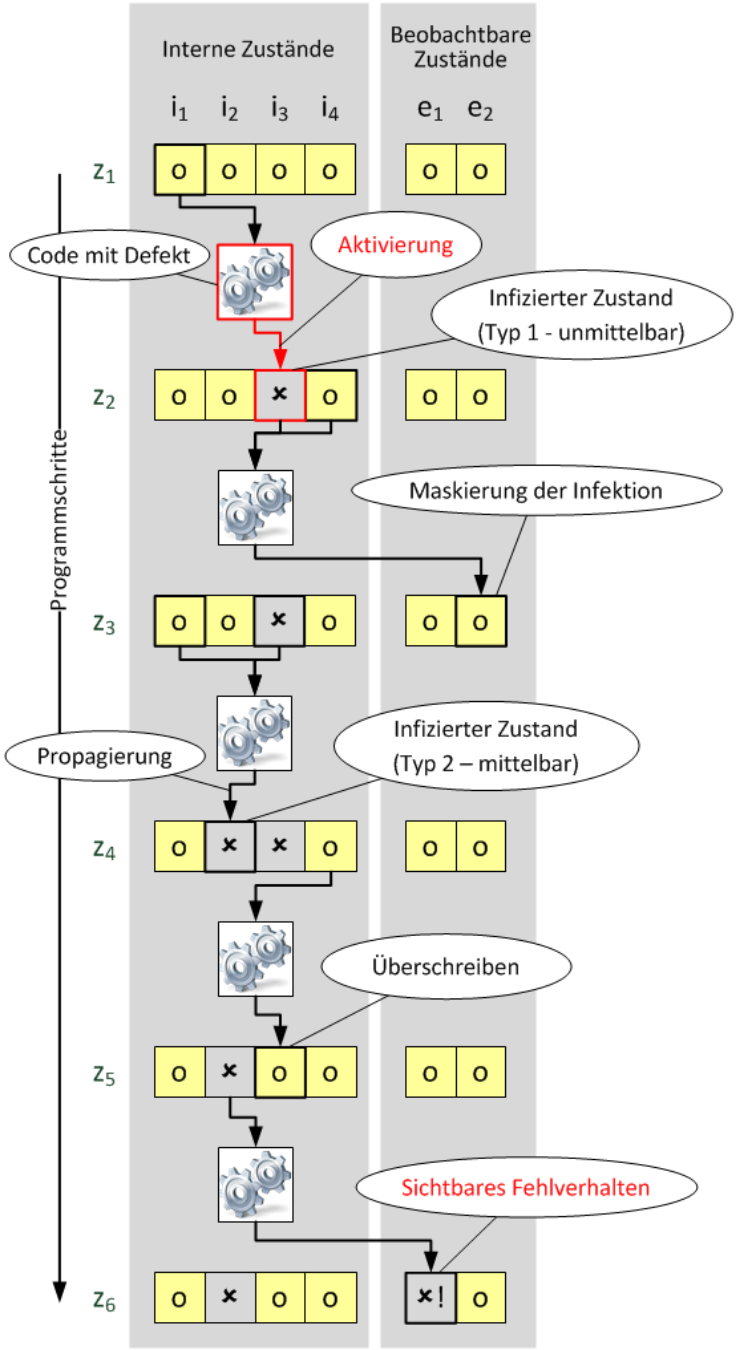
i_x: interner Zustand e_x: extern sichtbarer Zustand

z_x: Programmschritt

o korrekter Zustand

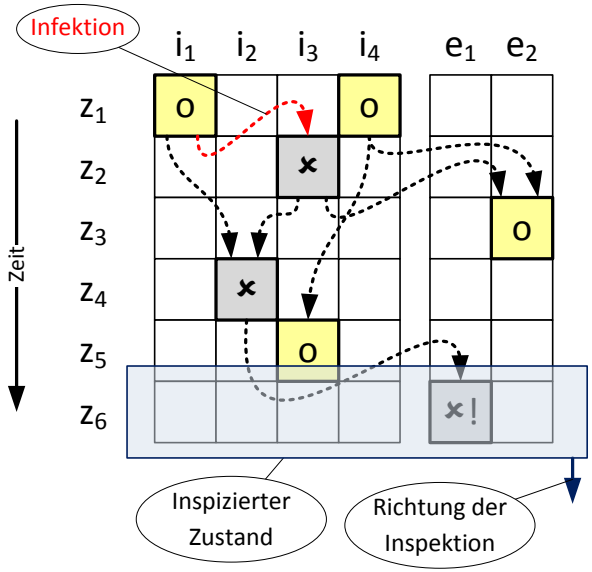
× fehlerhafter Zustand

×! fehlerhafter Zustand, extern sichtbar



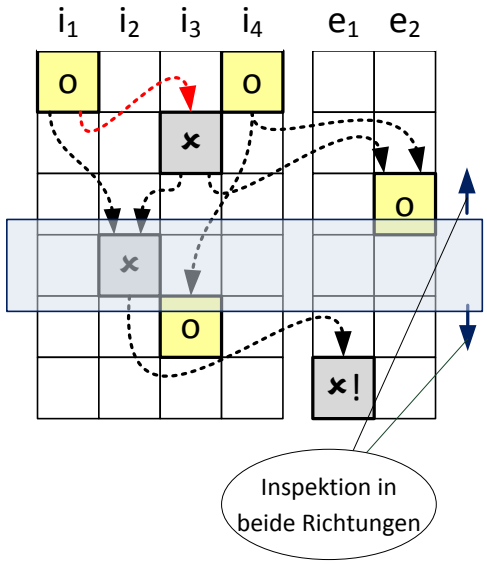
Reaktives Debuggen

Run/Stop Debuggen (auch Trace-Debuggen)



- häufigste Methode (noch…)
- schwierige Triggerdefinition
- intrusiv

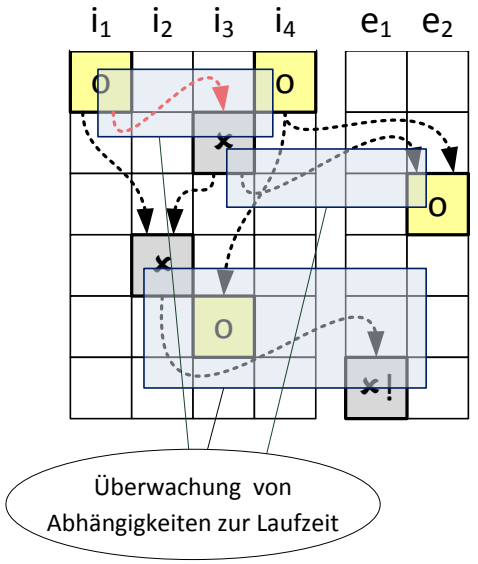
Omniscientes Debuggen



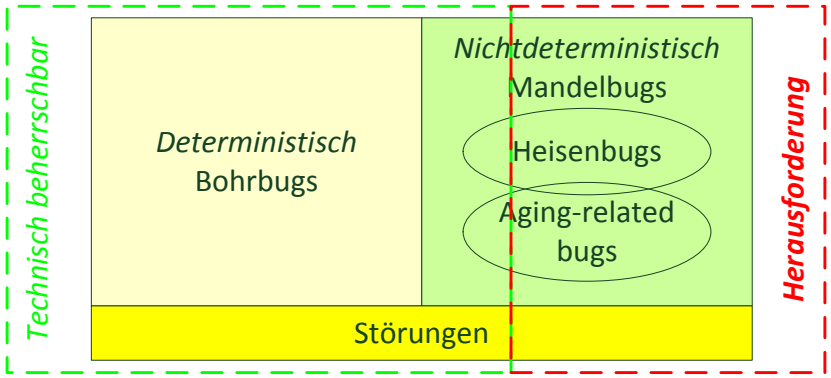
- schwierige Triggerdefinition
- kurze Beobachtungsfenster

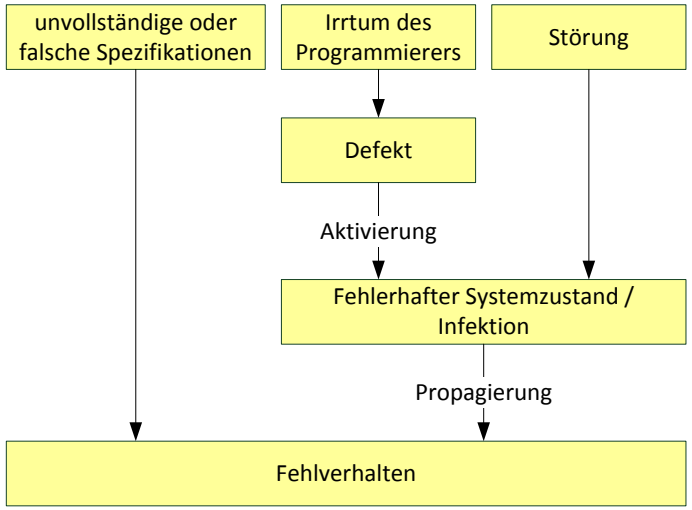
Proaktives Debuggen

Runtime Verification

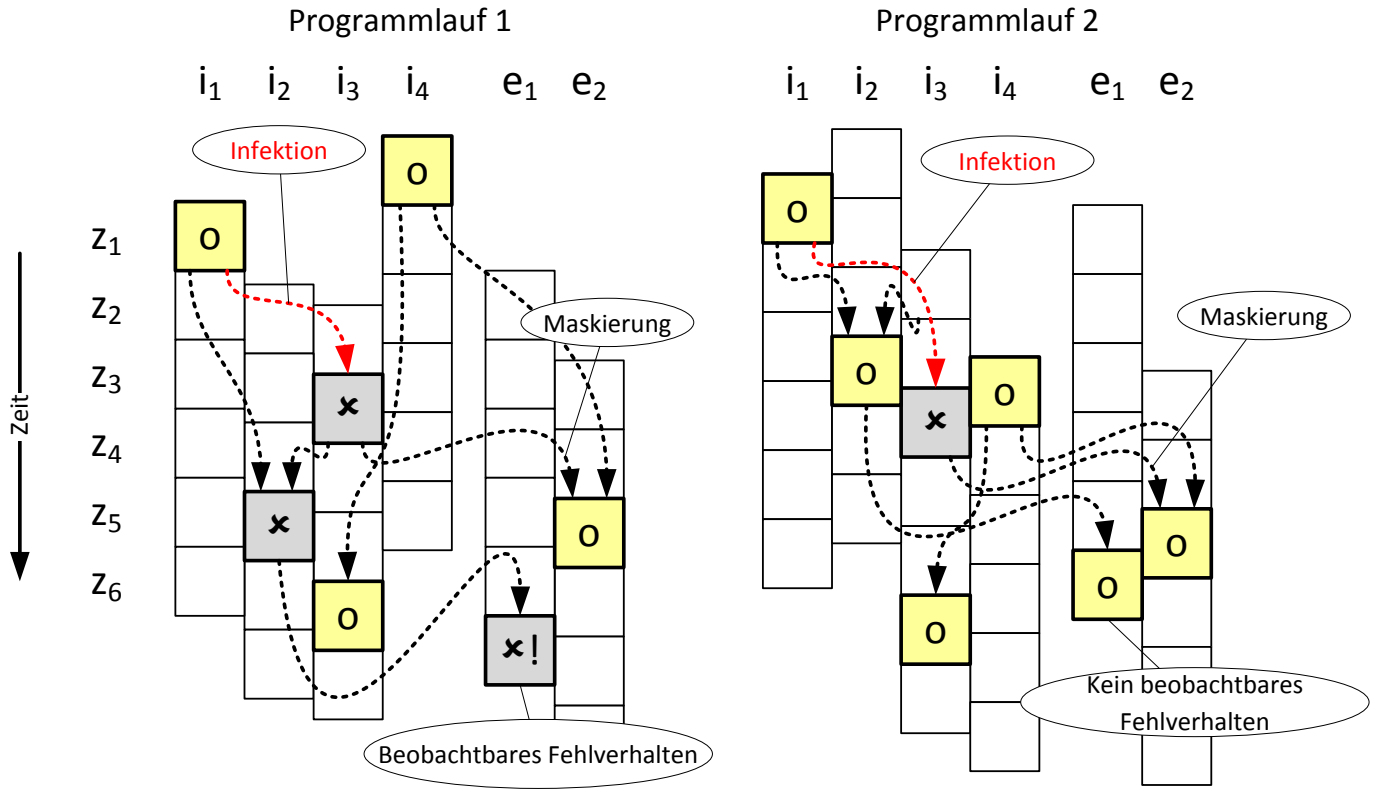


- Orakel-basiert
- automatisierbar
- parallele Operation
- technisch sehr anspruchsvoll





Bohrbug:	deterministisch beobachtbares Fehlverhalten
Mandelbug:	nichtdeterministisch, chaotisch erscheinendes Fehlverhalten
Heisenbug:	durch Beobachtung aktiviertes oder maskiertes Fehlverhalten (z.B. bei Software-Instrumentierung)
Aging-related bug:	von Laufzeit abhängiges Fehlverhalten


Programmlauf 1
i1 i2 i3 i4 e1 e2
Infektion
Maskierung
Beobachtbares Fehlverhalten
z1
z2
z3
z4
z5
z6
Zeit
Programmlauf 2
i1 i2 i3 i4 e1 e2
Infektion
Maskierung
Kein beobachtbares Fehlverhalten

Quelle: C. Jones, O. Bonsignour, "The Economics of Software Quality", Pearson Education, Inc. 2012