ISE
SRA

Leibniz Universität Hannover

Synthesis of Optimized AUTOSAR Embedded Systems

Fachgruppentreffen Bamberg 2023

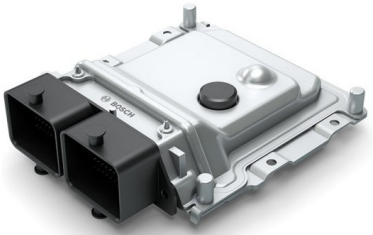
Andreas Kässens

kaessens@sra.uni-hannover.de

29.09.2023

Introduction

- Embedded real-time systems
- Large number of units
- Constrained hardware to save costs



AUTOSAR

freeRTOS

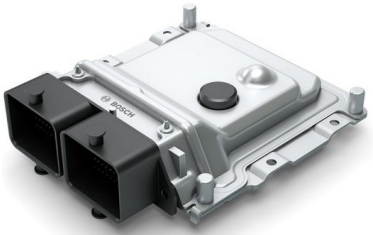
https://www.bosch-mobility.com/de/loesungen/steuergeraete/motormanagementsysteme-2rad/

Introduction

- Embedded real-time systems
- Large number of units
- Constrained hardware to save costs

Goal

Optimization of non-functional properties

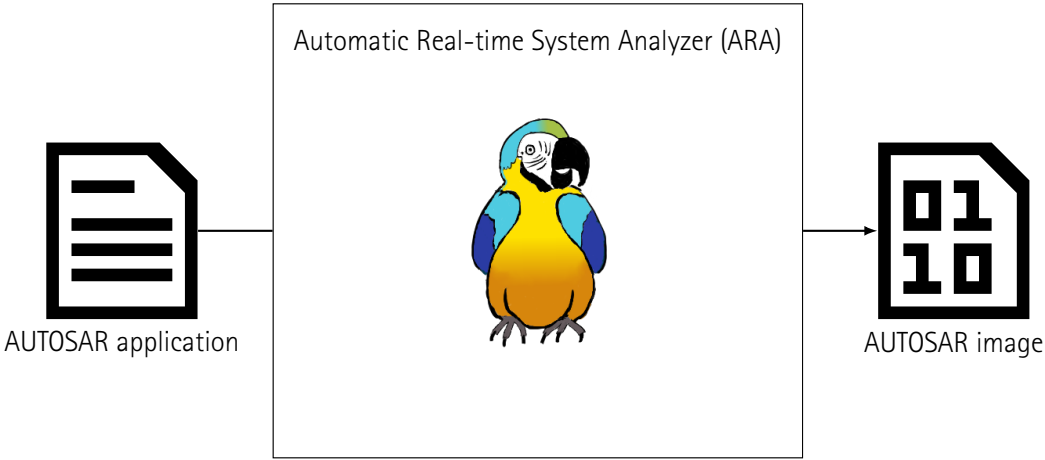


AUTOSAR

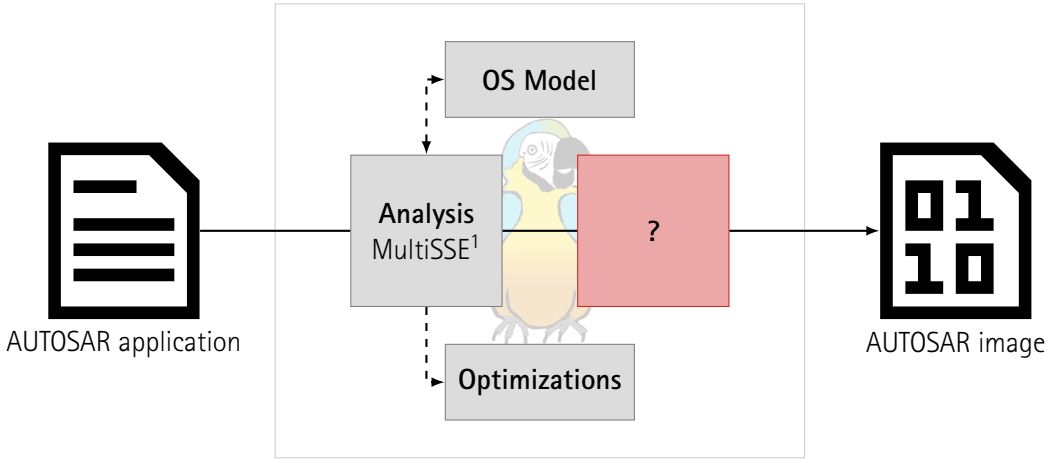
freeRTOS

https://www.bosch-mobility.com/de/loesungen/steuergeraete/motormanagementsysteme-2rad/

Overview

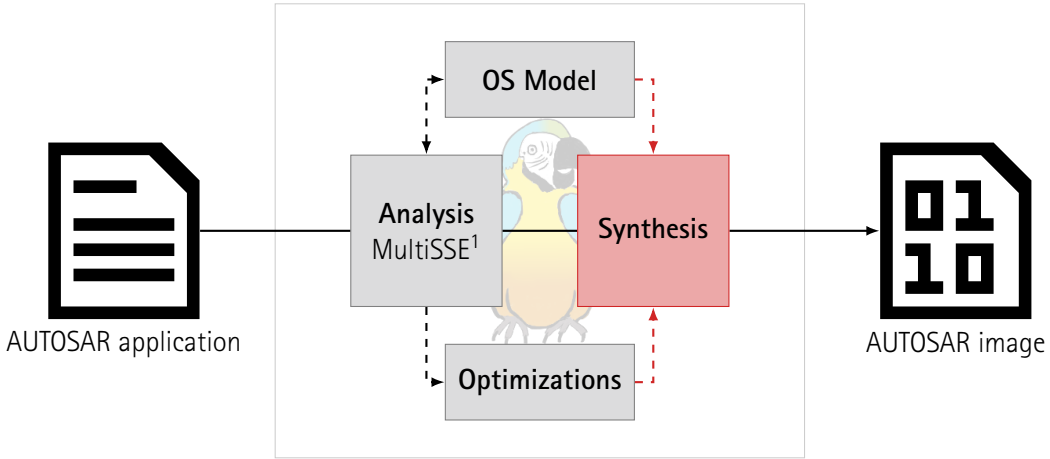

Automatic Real-time System Analyzer (ARA)
AUTOSAR application
AUTOSAR image

Overview



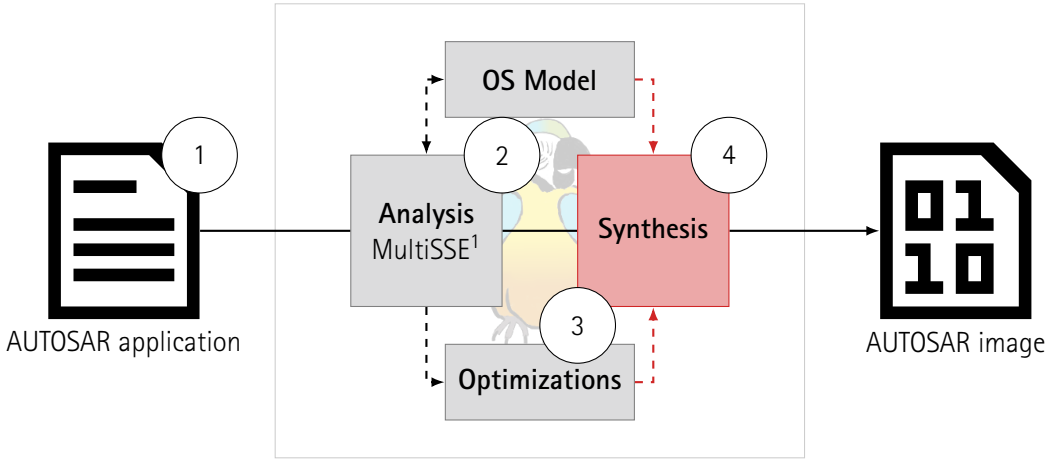
[1] [EFL23] MultiSSE: Static Syscall Elision and Specialization for Event-Triggered Multi-Core RTOS

Overview



[1] [EFL23] MultiSSE: Static Syscall Elision and Specialization for Event-Triggered Multi-Core RTOS

Overview



[1] [EFL23] MultiSSE: Static Syscall Elision and Specialization for Event-Triggered Multi-Core RTOS

- Static configuration of RTOS objects

```
cpus:
- id: 0
  tasks:
  - T01:
      autostart: true
      priority: 1
      spinlocks: [S1]
- id: 1
  tasks:
  - T12:
      autostart: false
      priority: 2
      spinlocks: [S1]
  - T13:
      autostart: false
      priority: 3
      spinlocks: []
```

```
1  TASK(T01) {
2    GetSpinlock(S1);
3    ActivateTask(T13);
4    ActivateTask(T12);
5    ReleaseSpinlock(S1);
6  }
7
8  TASK(T12) {
9    GetSpinlock(S1);
10   // critical section
11   ReleaseSpinlock(S1);
12 }
13
14 TASK(T13) {
15   long_computation();
16 }
17
```

- Static configuration of RTOS objects


```
cpus:
- id: 0
  tasks:
  - T01:
      autostart: true
      priority: 1
      spinlocks: [S1]
- id: 1
  tasks:
  - T12:
      autostart: false
      priority: 2
      spinlocks: [S1]
  - T13:
      autostart: false
      priority: 3
      spinlocks: []
```

```
1  TASK(T01) {
2    GetSpinlock(S1);
3    ActivateTask(T13);
4    ActivateTask(T12);
5    ReleaseSpinlock(S1);
6  }
7
8  TASK(T12) {
9    GetSpinlock(S1);
10   // critical section
11   ReleaseSpinlock(S1);
12 }
13
14 TASK(T13) {
15   long_computation();
16 }
17
```

- Static configuration of RTOS objects

```
cpus:
- id: 0
  tasks:
  - T01:
      autostart: true
      priority: 1
      spinlocks: [S1]
- id: 1
  tasks:
  - T12:
      autostart: false
      priority: 2
      spinlocks: [S1]
  - T13:
      autostart: false
      priority: 3
      spinlocks: []
```

```
1  TASK(T01) {
2    GetSpinlock(S1);
3    ActivateTask(T13);
4    ActivateTask(T12);
5    ReleaseSpinlock(S1);
6  }
7
8  TASK(T12) {
9    GetSpinlock(S1);
10   // critical section
11   ReleaseSpinlock(S1);
12 }
13
14 TASK(T13) {
15   long_computation();
16 }
17
```

- Static configuration of RTOS objects

```
cpus:
- id: 0
  tasks:
  - T01:
      autostart: true
      priority: 1
      spinlocks: [S1]
- id: 1
  tasks:
  - T12:
      autostart: false
      priority: 2
      spinlocks: [S1]
  - T13:
      autostart: false
      priority: 3
      spinlocks: []
```

```
1  TASK(T01) {
2    GetSpinlock(S1);
3    ActivateTask(T13);
4    ActivateTask(T12);
5    ReleaseSpinlock(S1);
6  }
7
8  TASK(T12) {
9    GetSpinlock(S1);
10   // critical section
11   ReleaseSpinlock(S1);
12 }
13
14 TASK(T13) {
15   long_computation();
16 }
17
```

- Static configuration of RTOS objects

```
cpus:
- id: 0
  tasks:
  - T01:
      autostart: true
      priority: 1
      spinlocks: [S1]
- id: 1
  tasks:
  - T12:
      autostart: false
      priority: 2
      spinlocks: [S1]
  - T13:
      autostart: false
      priority: 3
      spinlocks: []
```

```
1  TASK(T01) {
2    GetSpinlock(S1);
3    ActivateTask(T13);
4    ActivateTask(T12);
5    ReleaseSpinlock(S1);
6  }
7
8  TASK(T12) {
9    GetSpinlock(S1);
10   // critical section
11   ReleaseSpinlock(S1);
12 }
13
14 TASK(T13) {
15   long_computation();
16 }
17
```

ARA: MultiSSE Analysis

- Framework for whole-system compilation
  - Pipeline of analysis passes
  - LLVM Intermediate Representation (IR)

ARA: MultiSSE Analysis

- Framework for whole-system compilation
  - Pipeline of analysis passes
  - LLVM Intermediate Representation (IR)
- MultiSSE: enumerate all possible system states
  - Static application analysis with OS model
  - Result: Multicore State Transition Graph (MSTG)
  - Enables cross-core system call optimization

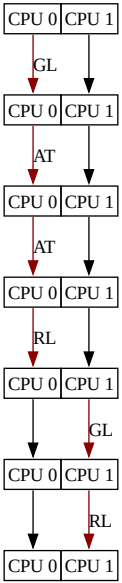
ARA: MultiSSE Analysis

- Framework for whole-system compilation
  - Pipeline of analysis passes
  - LLVM Intermediate Representation (IR)
- MultiSSE: enumerate all possible system states
  - Static application analysis with OS model
  - Result: Multicore State Transition Graph (MSTG)
  - Enables cross-core system call optimization
- Synthesis only for FreeRTOS[1]

[1] [Fie+21] ARA: Static Initialization of Dynamically-Created System Objects

ARA: MultiSSE Analysis - Example

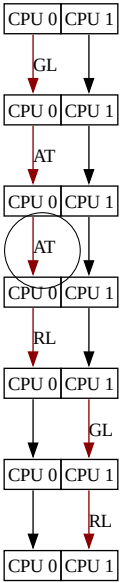
```
1  TASK(T01) { // core 0, prio 1, autostart
2    GetSpinlock(S1);
3    ActivateTask(T13);
4    ActivateTask(T12);
5    ReleaseSpinlock(S1);
6
7  }
8
9  TASK(T12) { // core 1, prio 2
10   GetSpinlock(S1);
11   // critical section
12   ReleaseSpinlock(S1);
13 }
14
15 TASK(T13) { // core 1, prio 3
16   long_computation();
17 }
18
```



GL = GetSpinlock
RL = ReleaseSpinlock
AT = ActivateTask

ARA: MultiSSE Analysis - Example

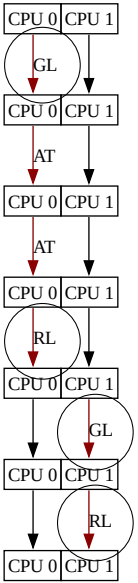
```
1  TASK(T01) { // core 0, prio 1, autostart
2    GetSpinlock(S1);
3    ActivateTask(T13);
4    ActivateTask(T12);
5    ReleaseSpinlock(S1);
6
7  }
8
9  TASK(T12) { // core 1, prio 2
10   GetSpinlock(S1);
11   // critical section
12   ReleaseSpinlock(S1);
13 }
14
15 TASK(T13) { // core 1, prio 3
16   long_computation();
17 }
18
```



GL = GetSpinlock
RL = ReleaseSpinlock
AT = ActivateTask

ARA: MultiSSE Analysis - Example

```
1  TASK(T01) { // core 0, prio 1, autostart
2    GetSpinlock(S1);
3    ActivateTask(T13);
4    ActivateTask(T12);
5    ReleaseSpinlock(S1);
6
7  }
8
9  TASK(T12) { // core 1, prio 2
10   GetSpinlock(S1);
11   // critical section
12   ReleaseSpinlock(S1);
13 }
14
15 TASK(T13) { // core 1, prio 3
16   long_computation();
17 }
18
```



GL = GetSpinlock
RL = ReleaseSpinlock
AT = ActivateTask

Optimizations

Goal

Optimization of non-functional properties

Optimizations

Goal

Optimization of non-functional properties
Remove costly but unnecessary operations to improve timing

Generated by DALL-E

Optimizations

Goal

Optimization of non-functional properties
Remove costly but unnecessary operations to improve timing

- Lock Elision
  - Lock is never spinning between `Get-` and `ReleaseSpinlock`
  - Remove locking operations! (or system calls)

Generated by DALL·E

Optimizations

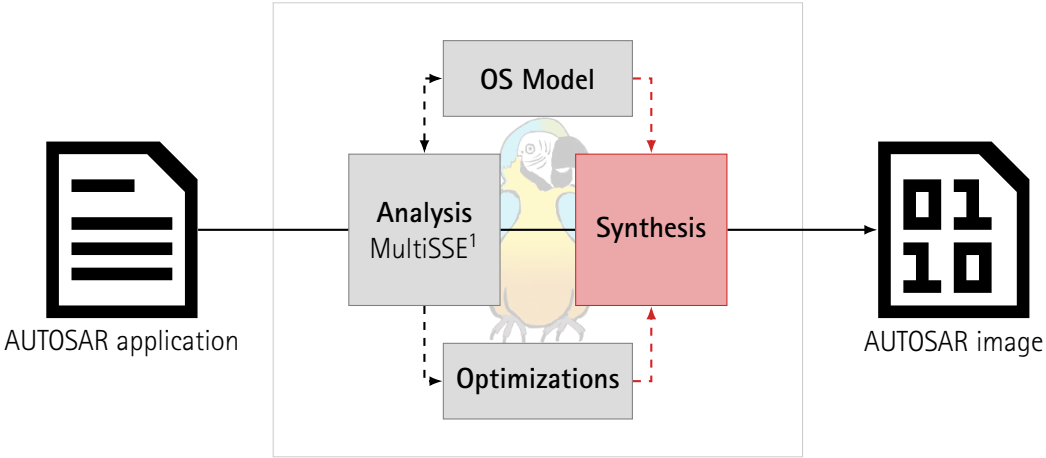
Goal

Optimization of non-functional properties
Remove costly but unnecessary operations to improve timing

- Lock Elision
  - Lock is never spinning between `Get-` and `ReleaseSpinlock`
  - Remove locking operations! (or system calls)
- IPI Avoidance
  - Cross-core system calls like `ActivateTask`
  - IPI does not lead to rescheduling
  - Don't trigger IPI!

Generated by DALL·E

Overview



[1] [EFL23] MultiSSE: Static Syscall Elision and Specialization for Event-Triggered Multi-Core RTOS

MACAW System Synthesis

Multicore **A**UTOSAR **C**ompatible **A**pplication-specific **W**hole-system-optimizer (MACAW[1])

[1] Macaws are a group of New World parrots that are long-tailed and often colorful. (Wikipedia)

MACAW System Synthesis

Multicore **A**UTOSAR **C**ompatible **A**pplication-specific **W**hole-system-optimizer (MACAW[1])

- AUTOSAR specification
  - Multicore system
  - OS objects, RTOS functionality
  - Interrupts
- Support ARA test applications
- System call optimization

[1] Macaws are a group of New World parrots that are long-tailed and often colorful. (Wikipedia)

MACAW System Synthesis

Multicore **A**UTOSAR **C**ompatible **A**pplication-specific **W**hole-system-optimizer (MACAW[1])

- AUTOSAR specification
  - Multicore system
  - OS objects, RTOS functionality
  - Interrupts
- Support ARA test applications
- System call optimization

- Extend *d*OSEK[2]
  - OSEK: single-core only
  - System call specialization for dependability
- POSIX platform
  - Cores → Threads
  - Interrupts → Signals

[1] Macaws are a group of New World parrots that are long-tailed and often colorful. (Wikipedia)
[2] [Hof+15] dOSEK: The Design and Implementation of a Dependability-Oriented Static Embedded Kernel

Implementation - MACAW

- Generic AUTOSAR functionality
  - Tasks, Events, Spinlocks, ...
  - Per-core scheduling and timers, IPIs, multicore handling

System Libraries

IR Modification

Generator

Implementation - MACAW

- Generic AUTOSAR functionality
  - Tasks, Events, Spinlocks, ...
  - Per-core scheduling and timers, IPIs, multicore handling
- Modification of application code
  - Specialized call sites: `ActivateTask` → `ActivateTask_BB18`
  - Automatic insertion: initialization, kickoff, error handling

Implementation - MACAW

- Generic AUTOSAR functionality
  - Tasks, Events, Spinlocks, ...
  - Per-core scheduling and timers, IPIs, multicore handling
- Modification of application code
  - Specialized call sites: `ActivateTask` → `ActivateTask_BB18`
  - Automatic insertion: initialization, kickoff, error handling
- System generation
  - Connecting the application to the AUTOSAR libraries
  - Application-specific instantiation and specialization

Implementation - System Call Specialization

- Analysis results:
  - IPI needed?
  - Locking needed?

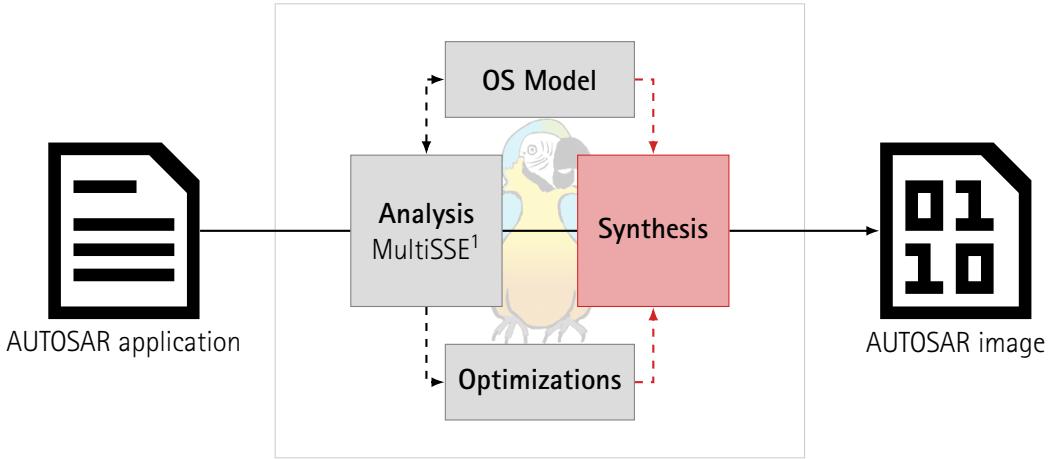
```
1  StatusType AUTOSAR_ActivateTask_BB18(TaskType arg0)
2  {
3    StatusType result = E_OK;
4    Machine::disable_interrupts();
5
6    scheduler_[1].SetReady_impl(OS_T10_task);
7    Machine::trigger_interrupt(SIGUSR1, 1, true);
8
9    Machine::enable_interrupts();
10   return result;
11 }
```

Implementation - System Call Specialization

- Analysis results:
  - IPI needed?
  - Locking needed?
- Elision of unnecessary operations
  - Reduced kernel path overhead
  - Improved timing

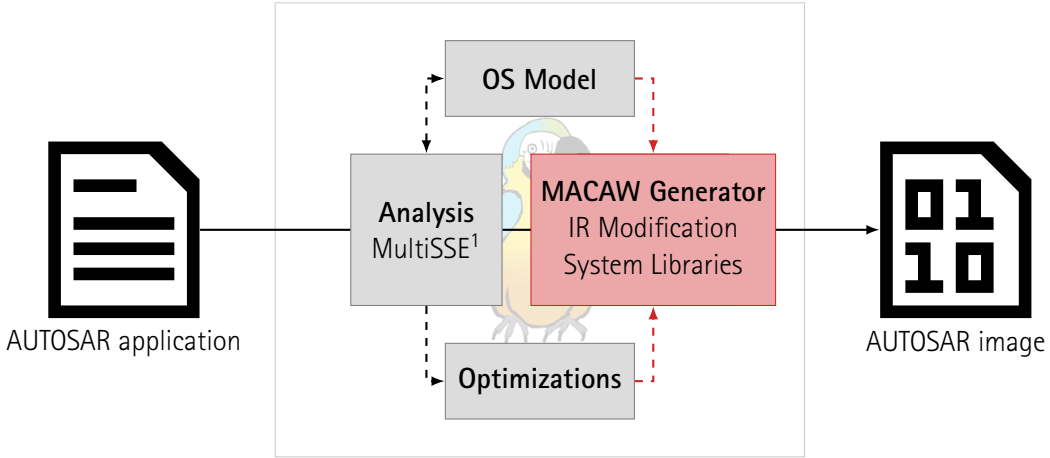
```
1  StatusType AUTOSAR_ActivateTask_BB18(TaskType arg0)
2  {
3    StatusType result = E_OK;
4    Machine::disable_interrupts();
5
6    scheduler_[1].SetReady_impl(OS_T10_task);
7    Machine::trigger_interrupt(SIGUSR1, 1, true);
8
9    Machine::enable_interrupts();
10   return result;
11 }
```

Overview



[1] [EFL23] MultiSSE: Static Syscall Elision and Specialization for Event-Triggered Multi-Core RTOS

Overview



[1] [EFL23] MultiSSE: Static Syscall Elision and Specialization for Event-Triggered Multi-Core RTOS

Results - AUTOSAR compatibility

- Synthesis supports all AUTOSAR model features
- Test applications

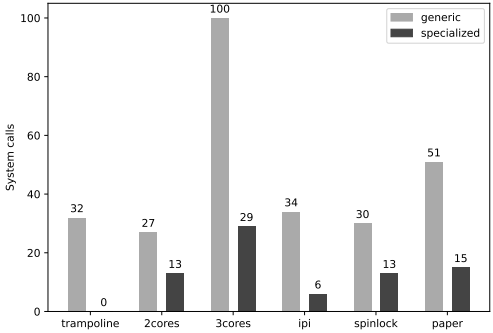
Real-world applications	✘
*d*OSEK tests	✔
Trampoline tests[1]	(✔)
MultiSSE tests	✔
MACAW tests	✔

```
Ok:                 105
Expected Fail:      0
Fail:               0
Unexpected Pass:    0
Skipped:            0
Timeout:            0

---
Not compiled:       9
```

[1] Missing AUTOSAR features: ScheduleTable, Timing Protection, TerminateApplication

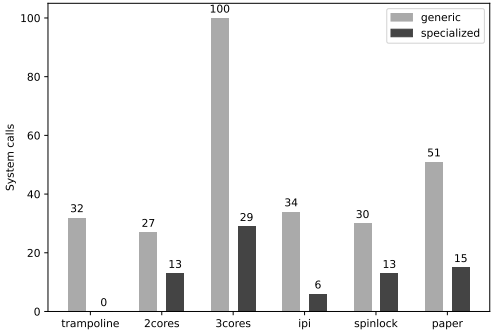
Results - Specialization

- How many system calls can be specialized?
  - Analyze test applications



  - 28 % of relevant system calls

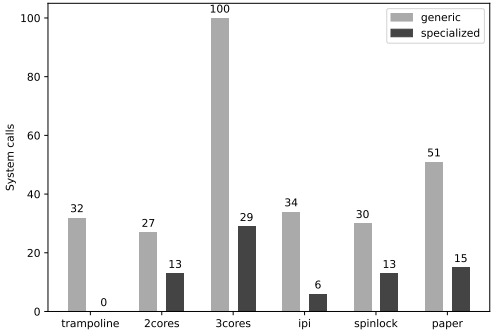
- How many system calls can be specialized?
  - Analyze test applications



  - 28 % of relevant system calls
  - Depends on the specific application

Results - Specialization

- How many system calls can be specialized?
  - Analyze test applications

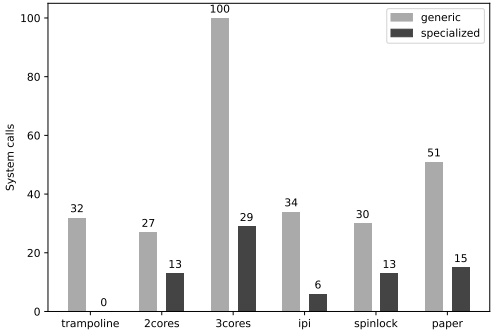


  - 28 % of relevant system calls
  - Depends on the specific application

- How does this improve timing?
  - Microbenchmarks:
    2.5µs for synchronized IPI
    0.01µs for Spinlock

Results - Specialization

- How many system calls can be specialized?
  - Analyze test applications



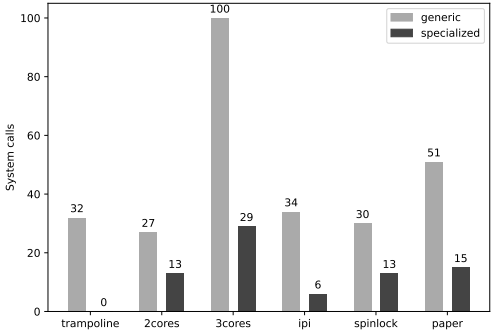
  - 28 % of relevant system calls
  - Depends on the specific application

- How does this improve timing?
  - Microbenchmarks:
    2.5µs for synchronized IPI
    0.01µs for Spinlock
  - Test applications

Application	Generic	Specialized
`ipi2_f`	22µs	19µs
`ipi2_d`	26µs	23µs
`spinlock_a`	29µs	29µs

Results - Specialization

- How many system calls can be specialized?
  - Analyze test applications



  - 28 % of relevant system calls
  - Depends on the specific application

- How does this improve timing?
  - Microbenchmarks:
    2.5µs for synchronized IPI
    0.01µs for Spinlock
  - Test applications

Application	Generic	Specialized
`ipi2_f`	22µs	19µs
`ipi2_d`	26µs	23µs
`spinlock_a`	29µs	29µs

  - Depends on the architecture

Conclusion

- MACAW: AUTOSAR synthesis in ARA
- Implemented on POSIX
- MultiSSE for system call optimization
- Measurable timing improvements

Conclusion

- MACAW: AUTOSAR synthesis in ARA
- Implemented on POSIX
- MultiSSE for system call optimization
- Measurable timing improvements

- Future work
  - ARM port, exact performance measurement
  - Higher specialization, system call elision (mode switch)

Conclusion

- MACAW: AUTOSAR synthesis in ARA
- Implemented on POSIX
- MultiSSE for system call optimization
- Measurable timing improvements

- Future work
  - ARM port, exact performance measurement
  - Higher specialization, system call elision (mode switch)

Thank You

Synchronization Function

```
_Atomic(int) cores = 1;
_Atomic(int) syncing_cores[arch::SYNC_MAX] = {0};

void sync_all_hardware_threads(cpu_sync_point_t sync_point) {
    syncing_cores[sync_point]++;
    // wait until all cores have synced
    while (syncing_cores[sync_point] < cores);
}
```

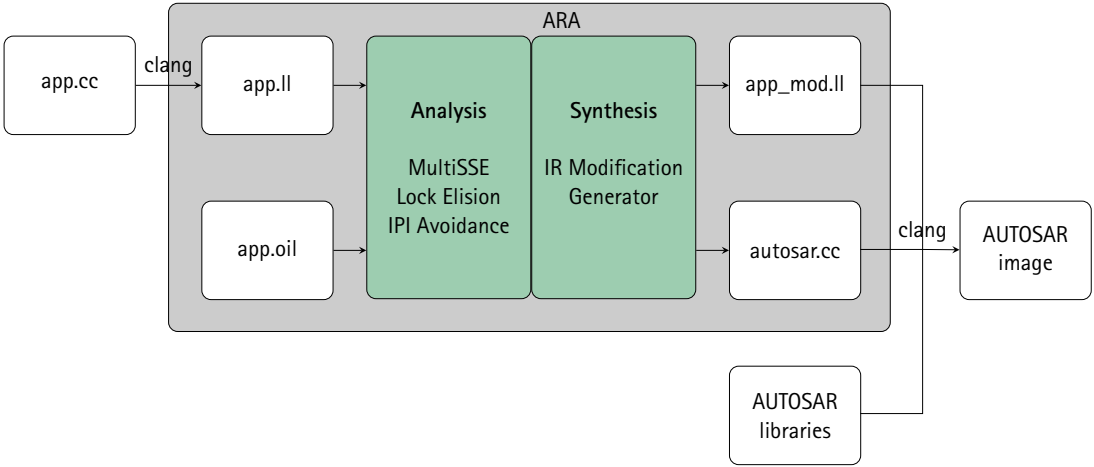
IR Modification

```
Function* kickoff = Function::Create(void_type_void,
    Function::ExternalLinkage,"AUTOSAR_kickoff", &module);

for (auto& function : module) {
    // all TASK(foo) functions start with this string, see os/os.h
    if (function.getName().startswith("AUTOSAR_TASK_FUNC")) {
        IRBuilder<> builder(context);
        BasicBlock& bb = *function.begin();
        Instruction* inst = &*bb.begin();
        builder.SetInsertPoint(inst);
        std::vector<Value*> ArgsV;
        builder.CreateCall(kickoff, ArgsV);
        /* ... */
    }
}
```

Interrupt function

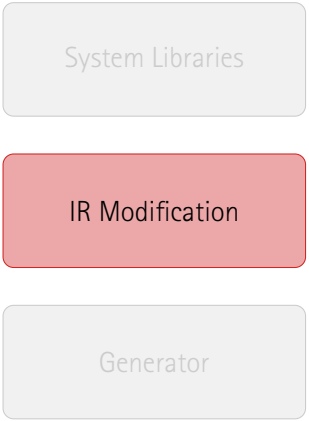
```
void IRQ::trigger_interrupt(int irq, int cpuid, bool sync) {
    if (!cpu_online(cpuid)) {
        debug_core << "cpuid " << cpuid << " offline" << endl;
        return;
    }

    if (!sync) {
        pthread_kill(get_thread_id(cpuid), irq);
        return;
    }

    /* synchronized interrupt: wait until target core does reschedule */
    ipi_cleared[cpuid] = false;
    pthread_kill(get_thread_id(cpuid), irq);
    while (ipi_cleared[cpuid] == false);
}
```

ARA Build overview



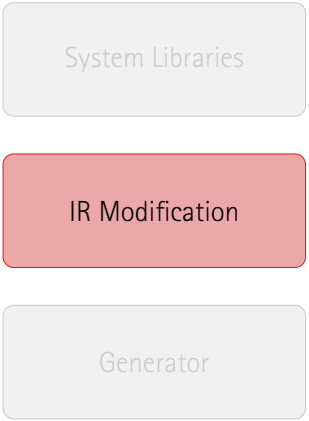
Implementation - IR Modification

- Insertions in the application's IR code
  - Kickoff function on `Task` function entry
  - Error handling on return from `Task`
  - Architecture-specific initialization in `main()`



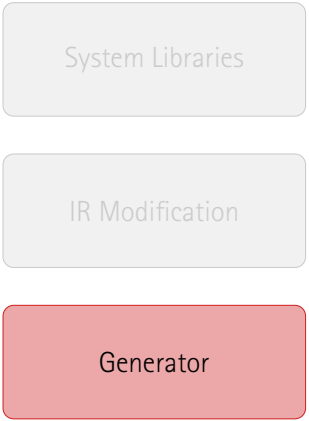
Implementation - IR Modification

- Insertions in the application's IR code
  - Kickoff function on `Task` function entry
  - Error handling on return from `Task`
  - Architecture-specific initialization in `main()`
- System call renaming
  - IR: Functions are split into Basic Blocks
  - Rename function name at each system call site
  - `ActivateTask` → `ActivateTask_BB18`
  - Specialization per call site possible



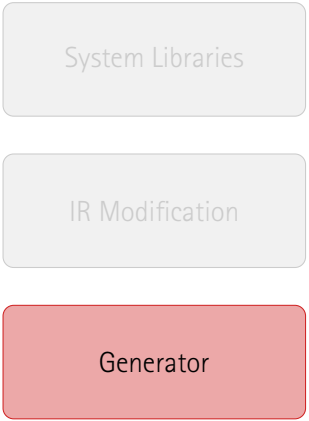
Implementation - Generator

- System generator (based on *d*OSEK)
  - Access ARA data structures and ValueAnalyzer results
  - Modify templates
  - Add new system calls



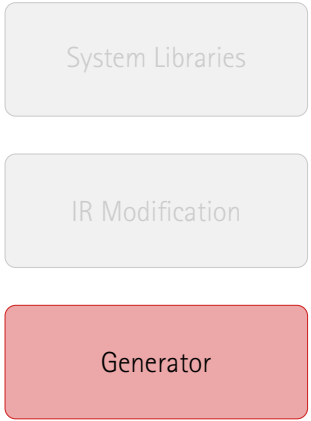
Implementation - Generator

- System generator (based on *d*OSEK)
  - Access ARA data structures and ValueAnalyzer results
  - Modify templates
  - Add new system calls
- Generator components
  - Generic OS instances
  - Architecture-specific code
  - System calls

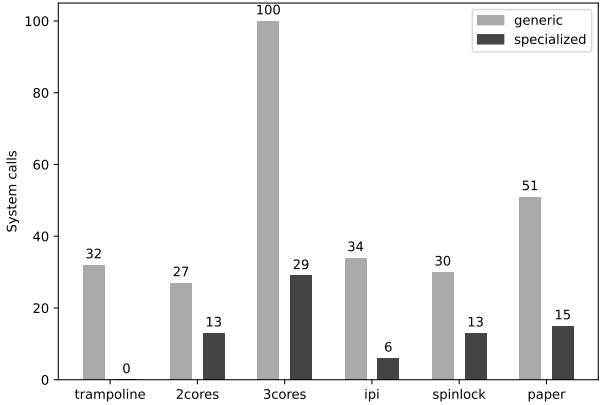


Implementation - Generator

- System generator (based on *d*OSEK)
  - Access ARA data structures and ValueAnalyzer results
  - Modify templates
  - Add new system calls
- Generator components
  - Generic OS instances
  - Architecture-specific code
  - System calls
- Result: `autosar.cc`



Results - Specialization



- 76/274 → 28 %
- Mostly Lock Elision

Results - Specialization

- POSIX Microbenchmarks

IPI	1.7 ms
IPI (sync)	**2.5 ms**
Spinlock	0.01 ms

- Test applications

Application	Generic	Specialized	Description
`ipi2_g`	22 µs	7 µs	1 / 1 IPIs avoided
`ipi2_f`	22 µs	19 µs	1 / 2 IPIs avoided
`ipi2_d`	26 µs	23 µs	1 / 3 IPIs avoided
`spinlock_a`	29 µs	29 µs	0 / 1 IPIs avoided, 4 / 8 (un)locks elided