NTNU
Norwegian University of
Science and Technology

Improving the performance of execution time control by using a hardware Time Management Unit

Kristoffer Nyborg Gregertsen

Department of Engineering Cybernetics

Ada-Europe 2012 – Stockholm – 2012-06-14

Summary

- Ada 2012 brings execution time control for interrupt handling

NTNU – Trondheim
Norwegian University of
Science and Technology

Summary

- Ada 2012 brings execution time control for interrupt handling
- Makes low overhead even more important

NTNU – Trondheim
Norwegian University of
Science and Technology

Summary

- Ada 2012 brings execution time control for interrupt handling
- Makes low overhead even more important
- Designed specialized Time Management Unit (TMU)

Summary

- Ada 2012 brings execution time control for interrupt handling
- Makes low overhead even more important
- Designed specialized Time Management Unit (TMU)
- Shown to significantly reduce execution time control overhead

NTNU – Trondheim
Norwegian University of
Science and Technology

Outline

Background and motivation

NTNU – Trondheim
Norwegian University of
Science and Technology

Outline

Background and motivation

Execution time control for interrupt handling

NTNU – Trondheim
Norwegian University of
Science and Technology

Outline

Background and motivation

Execution time control for interrupt handling

Implementation of Ada 2012 execution time control

NTNU – Trondheim
Norwegian University of
Science and Technology

Outline

Background and motivation

Execution time control for interrupt handling

Implementation of Ada 2012 execution time control

Time Management Unit (TMU)

NTNU – Trondheim
Norwegian University of
Science and Technology

Outline

Background and motivation

Execution time control for interrupt handling

Implementation of Ada 2012 execution time control

Time Management Unit (TMU)

Conclusion

NTNU – Trondheim
Norwegian University of
Science and Technology

Worst-case execution time (WCET)

- Need WCET for scheduling analysis

NTNU – Trondheim
Norwegian University of
Science and Technology

Worst-case execution time (WCET)

— Need WCET for scheduling analysis

— Hard to find on modern architectures:

- Deep pipelines
- Branch-prediction and speculative execution
- Multi-level cache and DRAM refresh cycle
- Multi-core with shared memory and coherent cache

Worst-case execution time (WCET)

— Need WCET for scheduling analysis

— Hard to find on modern architectures:

- Deep pipelines
- Branch-prediction and speculative execution
- Multi-level cache and DRAM refresh cycle
- Multi-core with shared memory and coherent cache

— Reported **30-50%** overestimation

Worst-case execution time (WCET)

- Need WCET for scheduling analysis
- Hard to find on modern architectures:
  - Deep pipelines
  - Branch-prediction and speculative execution
  - Multi-level cache and DRAM refresh cycle
  - Multi-core with shared memory and coherent cache
- Reported **30-50%** overestimation
- Also very pessimistic: real WCET $\gg$ average ET

Worst-case execution time (WCET)

- Need WCET for scheduling analysis
- Hard to find on modern architectures:
  - Deep pipelines
  - Branch-prediction and speculative execution
  - Multi-level cache and DRAM refresh cycle
  - Multi-core with shared memory and coherent cache
- Reported **30-50%** overestimation
- Also very pessimistic: real WCET $\gg$ average ET
- Using WCET as budget $\implies$ low utilization

Execution time control

— Dynamic control – not just static analysis

NTNU – Trondheim
Norwegian University of
Science and Technology

Execution time control

- Dynamic control – not just static analysis
- *Mechanism*:
  - Execution time measurement and monitoring
  - Handler called when timer expires

NTNU – Trondheim
Norwegian University of
Science and Technology

Execution time control

- Dynamic control – not just static analysis
- *Mechanism*:
  - Execution time measurement and monitoring
  - Handler called when timer expires
- *Policy*:
  - Task overrun handling
  - Execution time servers
  - Support advanced scheduling policies. . .

Execution time control

- Dynamic control – not just static analysis
- *Mechanism*:
    - Execution time measurement and monitoring
    - Handler called when timer expires
- *Policy*:
    - Task overrun handling
    - Execution time servers
    - Support advanced scheduling policies. . .
- Still need some timing analysis for budgets

NTNU – Trondheim
Norwegian University of
Science and Technology

Ada 2005 execution time control

— Execution time measurement for tasks:

- Package Ada.Execution_Time
- Type CPU_Time and function Clock

NTNU – Trondheim
Norwegian University of
Science and Technology

Ada 2005 execution time control

- Execution time measurement for tasks:
  - Package Ada.Execution_Time
  - Type CPU_Time and function Clock
- Execution time monitoring for single tasks:
  - Child package Timers with tagged type Timer
  - Set with protected handler – *expires*

NTNU – Trondheim
Norwegian University of
Science and Technology

Ada 2005 execution time control

— Execution time measurement for tasks:

- Package Ada.Execution_Time
- Type CPU_Time and function Clock

— Execution time monitoring for single tasks:

- Child package Timers with tagged type Timer
- Set with protected handler – *expires*

— Execution time budgets for dynamic task groups:

- Child package Group_Budgets tagged type Group_Budget

Ada 2005 execution time control

- Execution time measurement for tasks:
  - Package Ada.Execution_Time
  - Type CPU_Time and function Clock
- Execution time monitoring for single tasks:
  - Child package Timers with tagged type Timer
  - Set with protected handler – *expires*
- Execution time budgets for dynamic task groups:
  - Child package Group_Budgets tagged type Group_Budget
- Ravenscar – no timers or group budgets

Ada 2005 execution time control

- Tasks execution time defined as:
  - Time spent executing that task. . .
  - including services on behalf of task

NTNU – Trondheim
Norwegian University of
Science and Technology

Ada 2005 execution time control

- Tasks execution time defined as:
  - Time spent executing that task...
  - including services on behalf of task
- Execution time of interrupt handlers:
  - Implementation defined which task is charged
  - Implementations charge interrupted task
  - Inaccuracy to execution time measurement for tasks
  - Raised as an issue...

Ada 2005 execution time control

— Tasks execution time defined as:

- Time spent executing that task…
- including services on behalf of task

— Execution time of interrupt handlers:

- Implementation defined which task is charged
- Implementations charge interrupted task
- Inaccuracy to execution time measurement for tasks
- Raised as an issue…

— Also apply to other languages, POSIX…

NTNU – Trondheim
Norwegian University of
Science and Technology

Interrupt execution time control

— Is it right to charge interrupted task?

NTNU – Trondheim
Norwegian University of
Science and Technology

Interrupt execution time control

- Is it right to charge interrupted task?
- Separate execution time measurement:
  - Improves accuracy for tasks
  - Allows tighter task budgets
  - Testing and diagnostics. . .

NTNU – Trondheim
Norwegian University of
Science and Technology

Interrupt execution time control

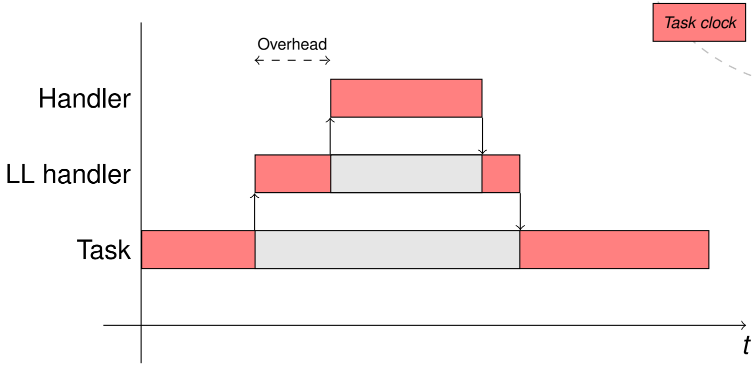
- Is it right to charge interrupted task?
- Separate execution time measurement:
  - Improves accuracy for tasks
  - Allows tighter task budgets
  - Testing and diagnostics. . .
- Full execution time control for interrupts:
  - Provide interrupt timers
  - Unexpected high interrupt rate
  - Bursts due to error. . .
  - Design and usage errors. . .

Interrupt execution time control

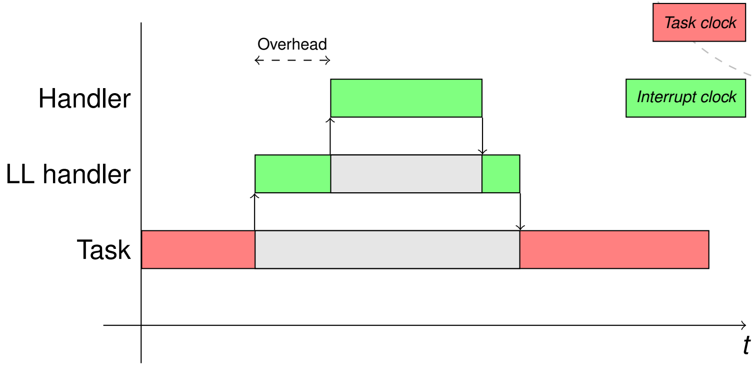
- Is it right to charge interrupted task?
- Separate execution time measurement:
  - Improves accuracy for tasks
  - Allows tighter task budgets
  - Testing and diagnostics. . .
- Full execution time control for interrupts:
  - Provide interrupt timers
  - Unexpected high interrupt rate
  - Bursts due to error. . .
  - Design and usage errors. . .
- Important with low overhead!

NTNU – Trondheim
Norwegian University of
Science and Technology

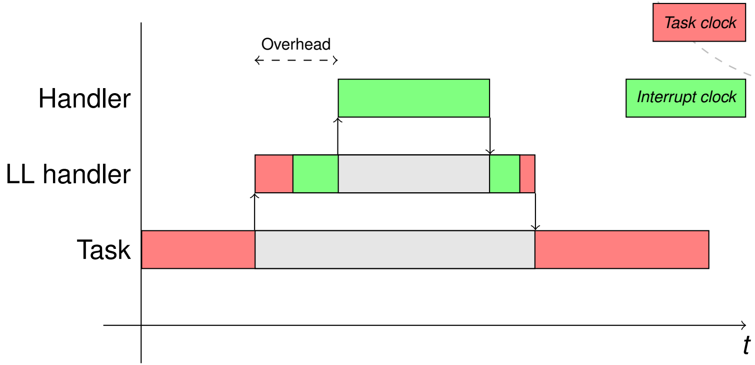
Interrupt handling



Interrupt handling – ideal



Interrupt handling – reality



NTNU – Trondheim
Norwegian University of
Science and Technology

Ada 2012 execution time control

- Discussed at IRTAW-14 in Portovenere, autumn 2009

NTNU – Trondheim
Norwegian University of
Science and Technology

Ada 2012 execution time control

- Discussed at IRTAW-14 in Portovenere, autumn 2009
- Total execution time for interrupt handling:
  - Rivas and Gonzalẽs Harbour
  - Implemented for MaRTE OS

NTNU – Trondheim
Norwegian University of
Science and Technology

Ada 2012 execution time control

— Discussed at IRTAW-14 in Portovenere, autumn 2009

— Total execution time for interrupt handling:

- Rivas and Gonzalẽs Harbour
- Implemented for MaRTE OS

— Separate execution time measurement for interrupts:

- Gregertsen and Skavhaug
- Implemented on GNATforAVR32
- Initially used interrupt *priorities*
- Updated to Interrupt_Id after discussion

NTNU – Trondheim
Norwegian University of
Science and Technology

Ada 2012 execution time control

— Discussed at IRTAW-14 in Portovenere, autumn 2009

— Total execution time for interrupt handling:

- Rivas and Gonzalẽs Harbour
- Implemented for MaRTE OS

— Separate execution time measurement for interrupts:

- Gregertsen and Skavhaug
- Implemented on GNATforAVR32
- Initially used interrupt *priorities*
- Updated to Interrupt_Id after discussion

— Workshop forwarded both proposals

NTNU – Trondheim
Norwegian University of
Science and Technology

Ada 2012 execution time control

- Discussed at IRTAW-14 in Portovenere, autumn 2009
- Total execution time for interrupt handling:
  - Rivas and Gonzalẽs Harbour
  - Implemented for MaRTE OS
- Separate execution time measurement for interrupts:
  - Gregertsen and Skavhaug
  - Implemented on GNATforAVR32
  - Initially used interrupt *priorities*
  - Updated to `Interrupt_Id` after discussion
- Workshop forwarded both proposals
- Now in draft for ISO-standard **Ada 2012**!

NTNU – Trondheim
Norwegian University of
Science and Technology

Ada 2012 execution time control

```
package Ada.Execution_Time is

    ...

    Interrupt_Clocks_Supported : constant Boolean :=
        implementation−defined;

    Separate_Interrupt_Clocks_Supported : constant Boolean :=
        implementation−defined;

    function Clock_For_Interrupts return CPU_Time;

private
     ...
end Ada.Execution_Time;
```

Ada 2012 execution time control

```
with Ada.Interrupts;

package Ada.Execution_Time.Interrupts is

    function Clock (Interrupt  :  Ada.Interrupts. Interrupt_Id )
         return CPU_Time;

    function Supported (Interrupt :  Ada.Interrupts. Interrupt_Id )
         return Boolean;

end Ada.Execution_Time.Interrupts;
```

Interrupt timer proposal

```
with Ada.Execution_Time.Timers;

package Ada.Execution_Time.Interrupts.Timers is

   type Interrupt_Timer (I  :  Ada.Interrupts. Interrupt_Id )
      is  new Ada.Execution_Time.Timers.Timer
     (Ada.Task_Identification.Null_Task_Id'Access)
     with private;

private
    ...
end Ada.Execution_Time.Interrupts.Timers;
```

— Implemented in GNATforAVR32

NTNU – Trondheim
Norwegian University of
Science and Technology

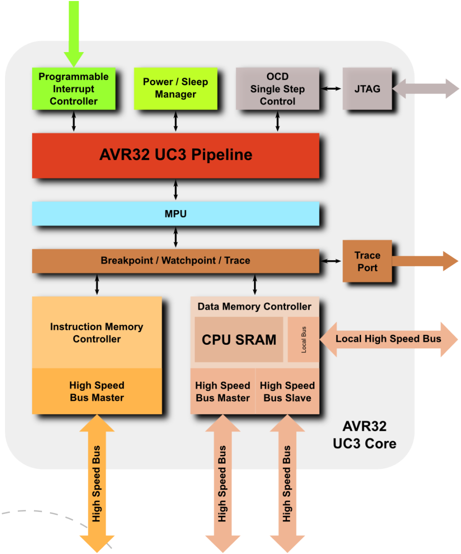
Interrupt timer proposal

```
with Ada.Execution_Time.Timers;

package Ada.Execution_Time.Interrupts.Timers is

   type Interrupt_Timer (I  :  Ada.Interrupts. Interrupt_Id )
      is  new Ada.Execution_Time.Timers.Timer
     (Ada.Task_Identification.Null_Task_Id'Access)
     with private;

private
    ...
end Ada.Execution_Time.Interrupts.Timers;
```

— Implemented in GNATforAVR32

— Not to be included in Ada 2012…

Atmel AVR32 UC3 series

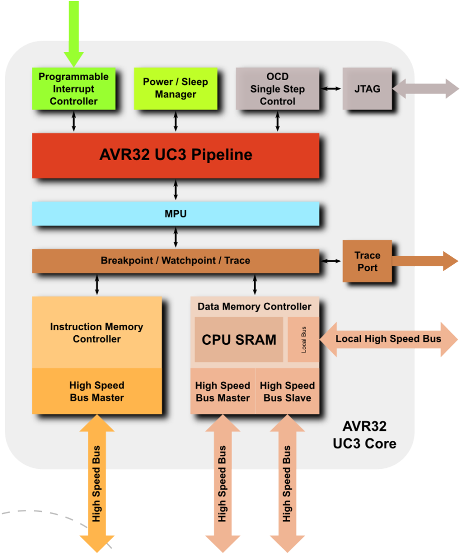


— Atmel AVR32 architecture:

- 32-bit RISC
- Efficient ISA
- 4 interrupt levels
- Atmel Norway

NTNU – Trondheim
Norwegian University of
Science and Technology

Atmel AVR32 UC3 series



- Atmel AVR32 architecture:
  - 32-bit RISC
  - Efficient ISA
  - 4 interrupt levels
  - Atmel Norway
- UC3 microcontroller series:
  - Second implementation
  - Embedded control apps.
  - Integrated SRAM
  - 16 to 64 KB SRAM
  - Up to 60 MHz

GNATforAVR32

— GNU Ada Compiler (GNAT) for AVR32 architecture:

- GNU Compiler Collection (GCC)
- GNAT front-end → AVR32 back-end

GNATforAVR32

- GNU Ada Compiler (GNAT) for AVR32 architecture:
  - GNU Compiler Collection (GCC)
  - GNAT front-end → AVR32 back-end
- Bare-board Ravenscar run-time environment:
  - Open Ravenscar Kernel by UPM
  - Used by ESA’s LEON space application processor
  - Real-time kernel integrated with GNARL
  - Ported to UC3 microcontroller series

GNATforAVR32

- GNU Ada Compiler (GNAT) for AVR32 architecture:
  - GNU Compiler Collection (GCC)
  - GNAT front-end → AVR32 back-end
- Bare-board Ravenscar run-time environment:
  - Open Ravenscar Kernel by UPM
  - Used by ESA's LEON space application processor
  - Real-time kernel integrated with GNARL
  - Ported to UC3 microcontroller series
- Small code size – low memory requirements

NTNU – Trondheim
Norwegian University of
Science and Technology

Ada 2012 implementation

— Similarities between RTC and execution time clocks:

- Same clock and alarm abstraction
- Use the COUNT / COMPARE timer for both clocks
- Reset and reprogram on clock change
- Tick-less clocks

NTNU – Trondheim
Norwegian University of
Science and Technology

Ada 2012 implementation

— Similarities between RTC and execution time clocks:

- Same clock and alarm abstraction
- Use the COUNT / COMPARE timer for both clocks
- Reset and reprogram on clock change
- Tick-less clocks

— Interrupt handling:

- Handler registered – allocated clock from pool
- Change clock before calling handler
- Store interrupted clock on stack

NTNU – Trondheim
Norwegian University of
Science and Technology

Ada 2012 implementation

— Similarities between RTC and execution time clocks:

- Same clock and alarm abstraction
- Use the COUNT / COMPARE timer for both clocks
- Reset and reprogram on clock change
- Tick-less clocks

— Interrupt handling:

- Handler registered – allocated clock from pool
- Change clock before calling handler
- Store interrupted clock on stack

— Low overhead – *can it be further reduced?*

NTNU – Trondheim
Norwegian University of
Science and Technology

Time Management Unit (TMU)

— HW timer specialized for execution time control:

- 64-bit COUNT / COMPARE registers
- Interrupt line asserted when COUNT $\geq$ COMPARE
- Atomic swapping of COUNT / COMPARE values
- Triggered by write to final swap register

Time Management Unit (TMU)

— HW timer specialized for execution time control:

- 64-bit COUNT / COMPARE registers
- Interrupt line asserted when COUNT $\geq$ COMPARE
- Atomic swapping of COUNT / COMPARE values
- Triggered by write to final swap register

— Memory-mapped interface:

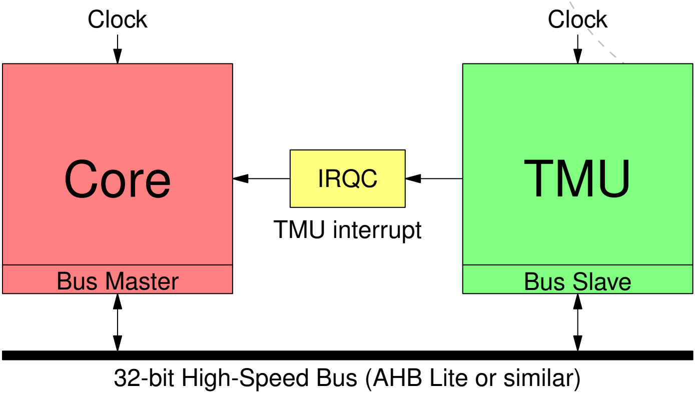
- Portable to different architectures
- Easy to use, no special instructions

Time Management Unit (TMU)

- HW timer specialized for execution time control:
  - 64-bit COUNT / COMPARE registers
  - Interrupt line asserted when COUNT $\geq$ COMPARE
  - Atomic swapping of COUNT / COMPARE values
  - Triggered by write to final swap register
- Memory-mapped interface:
  - Portable to different architectures
  - Easy to use, no special instructions
- Functional specification in SystemC

NTNU – Trondheim
Norwegian University of
Science and Technology

Overview



NTNU – Trondheim
Norwegian University of
Science and Technology

Memory map

Offset	Register	Reset state
0x00	TMU_COMPARE_HI	0xffffffff
0x04	TMU_COMPARE_LO	0xffffffff
0x08	TMU_COUNT_HI	0
0x0c	TMU_COUNT_LO	0
0x10	TMU_SWAP_COMPARE_HI	0xffffffff
0x14	TMU_SWAP_COMPARE_LO	0xffffffff
0x18	TMU_SWAP_COUNT_HI	0
0x1c	TMU_SWAP_COUNT_LO	0

TMU implementation for UC3

— Implemented for UC3 by master student:

- High-speed bus → peripheral bus
- Bound to peripheral bus clock for synchronous design
- Interface like other AVR32 peripherals
- Interrupt control registers
- Disabled by default

TMU implementation for UC3

- Implemented for UC3 by master student:
  - High-speed bus → peripheral bus
  - Bound to peripheral bus clock for synchronous design
  - Interface like other AVR32 peripherals
  - Interrupt control registers
  - Disabled by default
- Main change is move to peripheral bus:
  - Increased latency for access
  - Reduced predictability

TMU implementation for UC3

— Implemented for UC3 by master student:

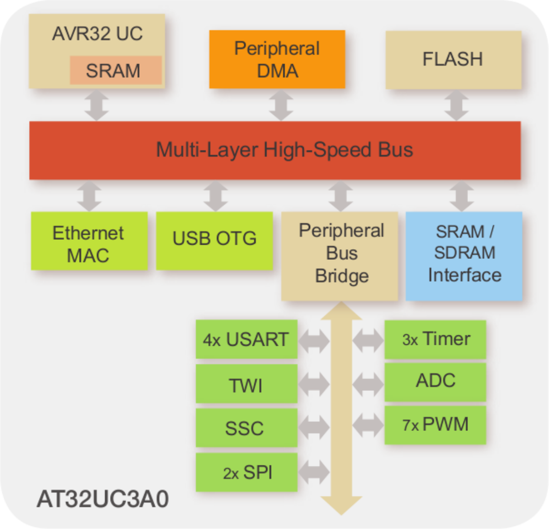
- High-speed bus → peripheral bus
- Bound to peripheral bus clock for synchronous design
- Interface like other AVR32 peripherals
- Interrupt control registers
- Disabled by default

— Main change is move to peripheral bus:

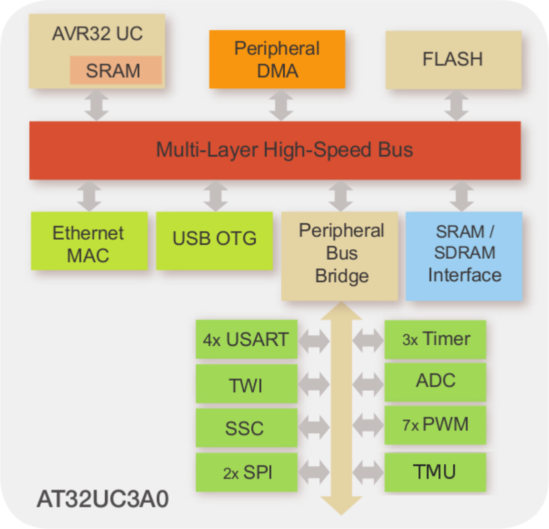
- Increased latency for access
- Reduced predictability

— Possible to use local CPU bus

NTNU – Trondheim
Norwegian University of
Science and Technology



NTNU – Trondheim
Norwegian University of
Science and Technology



NTNU – Trondheim
Norwegian University of
Science and Technology

Ada 2012 implementation with TMU

- Take advantage of powerful AVR32 instructions:
  - Load / store 64-bit values
  - Atomic access to COUNT / COMPARE
  - Load / store several registers
  - Efficient swap operation

NTNU – Trondheim
Norwegian University of
Science and Technology

Ada 2012 implementation with TMU

— Take advantage of powerful AVR32 instructions:

- Load / store 64-bit values
- Atomic access to COUNT / COMPARE
- Load / store several registers
- Efficient swap operation

— Only few changes needed in run-time environment:

- Interface to TMU
- Clock interface → two HW clocks
- Context switch

Ada 2012 implementation with TMU

- Take advantage of powerful AVR32 instructions:
  - Load / store 64-bit values
  - Atomic access to COUNT / COMPARE
  - Load / store several registers
  - Efficient swap operation
- Only few changes needed in run-time environment:
  - Interface to TMU
  - Clock interface → two HW clocks
  - Context switch
- Tested with synthesizable UC3 code

NTNU – Trondheim
Norwegian University of
Science and Technology

Performance improvements

Test	Improvement	
	CPU cycles	Reduction (%)
Context switch	65	54
Interrupt handler	30	25
Timing event	4	4
Interruption cost	42	21

— Compared to implementation without TMU

NTNU – Trondheim
Norwegian University of
Science and Technology

Performance improvements

Test	Improvement	
	CPU cycles	Reduction (%)
Context switch	65	54
Interrupt handler	30	25
Timing event	4	4
Interruption cost	42	21

- Compared to implementation without TMU
- Significant overhead reductions

Conclusion

- Execution time control for interrupts in Ada 2012:
  - Total and separate execution time measurement
  - Important with low overhead!

NTNU – Trondheim
Norwegian University of
Science and Technology

Conclusion

— Execution time control for interrupts in Ada 2012:

- Total and separate execution time measurement
- Important with low overhead!

— Implementation on GNATforAVR32:

- 32-bit timer – tick-less measurement
- Non-standard interrupt timer
- Acceptable overhead – could be reduced...

Conclusion

— Execution time control for interrupts in Ada 2012:

- Total and separate execution time measurement
- Important with low overhead!

— Implementation on GNATforAVR32:

- 32-bit timer – tick-less measurement
- Non-standard interrupt timer
- Acceptable overhead – could be reduced. . .

— Time Management Unit:

- Specialized 64-bit timer for execution time control
- Implemented and tested with AVR32 UC3
- Significantly reduces overhead

NTNU – Trondheim
Norwegian University of
Science and Technology