Timing instructions for RISC-V based hard real time edge devices

Nithin Ravani Nanjundaswamy (nithin.ravaninanjundaswamy@dlr.de)

DLR

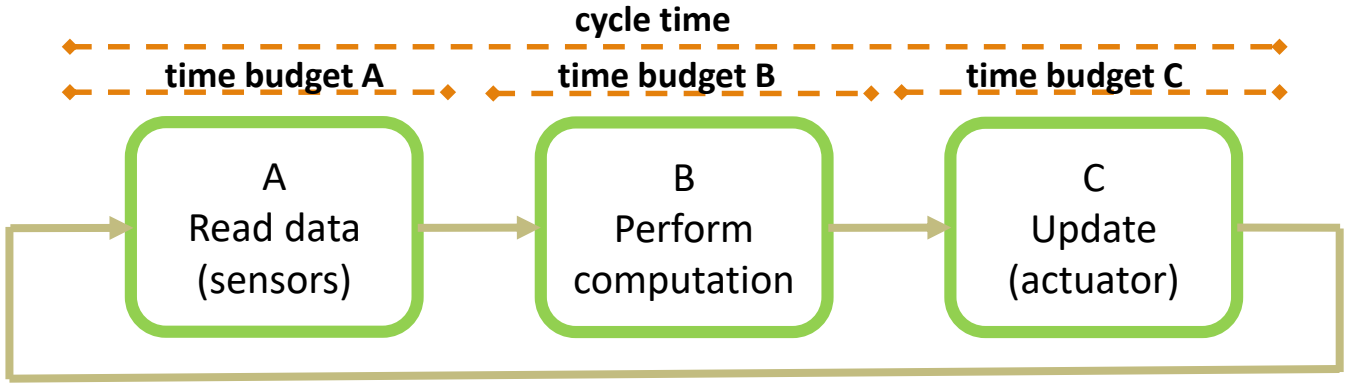
This work has been developed in the ZuSE project Scale4Edge. Scale4Edge is funded by the German ministry of education and research (BMBF) (reference numbers: 16ME0122K-16ME0140). The authors are responsible for the content of this publication.

SPONSORED BY THE

Federal Ministry of Education and Research

Motivation

- Most real-time edge devices follow cyclic execution and require precise timing and deterministic behavior



cycle time >= time budget A + time budget B + time budget C

- Achieving precise timing is usually realized through software that relies on HW timers/counters, interrupts and interrupt service routines
- Hypothesis: Dedicated RISC-V custom timing instructions enable low-overhead (number of instructions) and highly precise timing (number of clock cycles) measurement and control
- Proof-of-concept: Three new instructions (measure, smonit and emonit) are implemented on Murax SoC which is based on VexRiscV core.

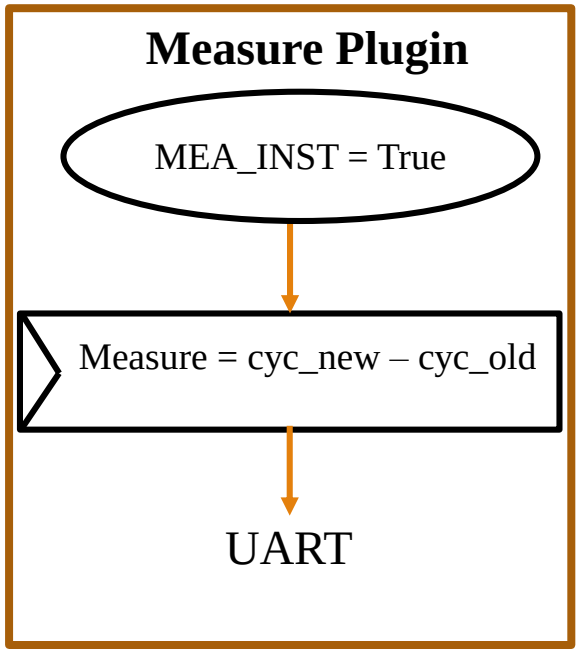
Timing Measure Instruction

- Helps obtaining software execution time
- Counts clock cycles between successive measure instructions
- Measure plugin implemented in SCALA and added to Murax SoC
- measure instruction is of R-type

```
measure;

//  software to obtain
// temporal behavior
instruction 1;
…….
instruction n;

measure;
```



0000000	00000	00000	000	00000	0001011
funct7	**rs2**	**rs1**	**funct3**	**rd**	**opcode**

Timing Control Instructions

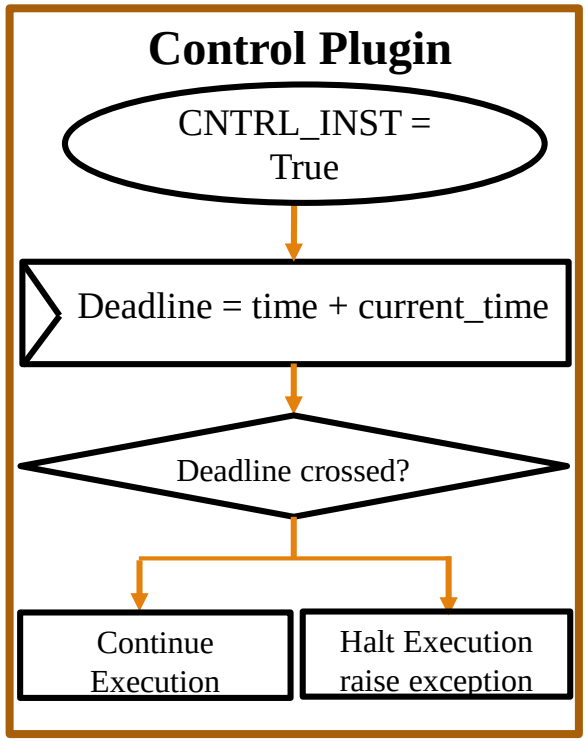
- Enforces a specific timing behavior
- smonit instruction (R-type)- start monitoring
- emonit instruction (R-type)- end monitoring

```
smonit(Block_id, time);

//  software to be monitored
instruction 1;
…….
instruction n;

emonit(Block_id);
```

- Exception raised if deadline crossed
- Nesting of control blocks is possible



0000000	00000	00001	010	00000	0001011
funct7	**rs2**	**rs1**	**funct3**	**rd**	**opcode**

0000000	00011	00001	011	00000	0001011
funct7	**rs2**	**rs1**	**funct3**	**rd**	**opcode**

Control Blocks Nesting

Scale4Edge

Nesting without overlap

```
smonit(1, t1);
instruction 1;
instruction 2;
        smonit(2, t2);
        instruction 3;
        instruction 4;
        emonit(2);
.....
instruction n;
emonit(1);
```

Nesting with overlap

```
smonit(1, t1);
instruction 1;
instruction 2;
        smonit(2, t2);
        instruction 3;
        instruction 4;
emonit(1);
        .....
        instruction n;
        emonit(2);
```

- Control instructions support nesting with & without overlap

Reference Software Solutions

Scale4Edge

- Software solutions rely on programmable timers/counters and interrupts
- Libbla (https://github.com/offis/libbla)
  - C++ based library originally developed for an ARM processor
  - Ported to RISC-V using counter register
  - Provides timing annotations to obtain temporal behavior
  - Estimate Execution Time (EET) – software counterpart for measure instruction
  - Forced Execution Time (FET) – software counterpart for control instructions
- C based optimized
  - Mere counter to count clock cycles without OOPs concepts

```
EET{
//  software to obtain
// temporal behavior
}
```

FET-Block structure

```
While(true){
  BLOCK_FET(100_ms){
     BLOCK_FET(5_ms){
        …
     }
     BLOCK_FET(5_ms){
        …
     }
     BLOCK_FET(10_ms){
        …
     }
  }
}
```

Results, Conclusion & Outlook

Hardware vs Libbla SW solution

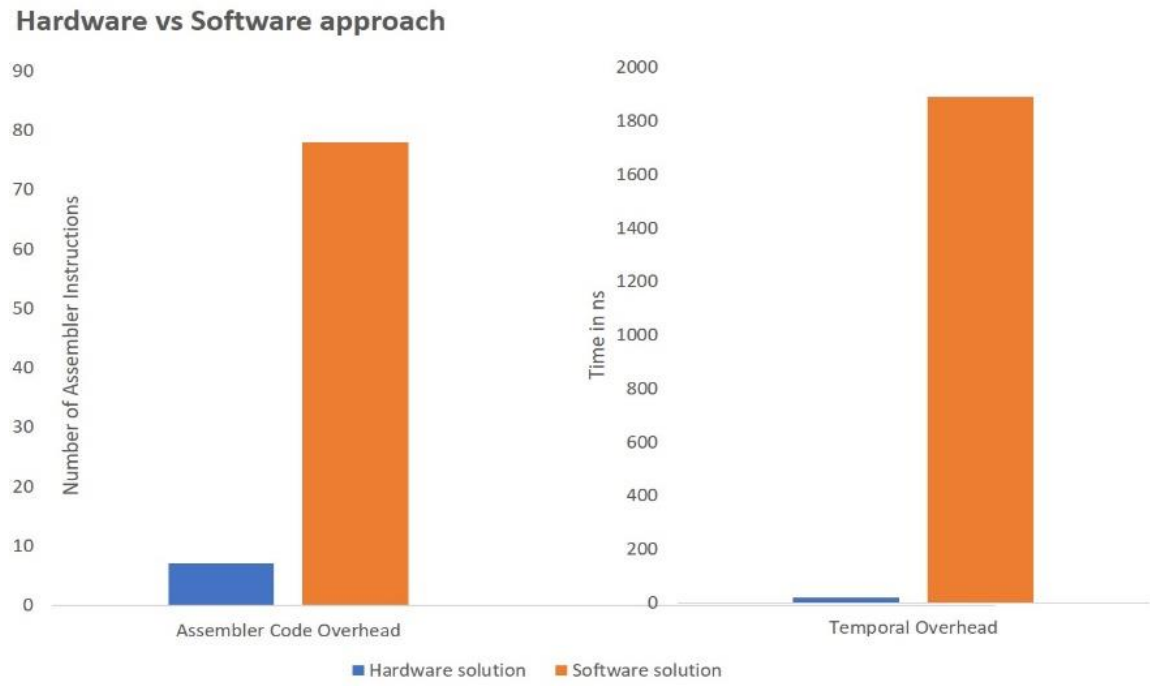
Measurement block	Assembler code overhead	Temporal overhead
Hardware solution	7 (0.5%)	20 ns
Libbla solution	1169 (101.5%)	3.002 ms

Hardware vs optimized C SW solution

Measurement block	Assembler code overhead	Temporal overhead
Hardware solution	7	20 ns
Software solution	78	1.89 µs

Hardware Utilisation

	LUTs	Registers
Measure	189 (11.5%)	200 (11.97%)
Control	276 (17.18%)	225 (13.47%)



Conclusion

- Precise temporal behavior with low overhead is achieved with timing instructions as compared to software approach
- ISAX comes at a cost of hardware overhead

Outlook

- Can be combined with power management (slack time)
- Interface to safety and security monitoring