

Improving Flexibility and Responsiveness in FTT-CAN with a Scheduling Coprocessor

E. Martins, J. Fonseca

**DET – IEETA
University of Aveiro
Aveiro - Portugal**

4th International Conference on Fieldbus Systems and their Applications
Nancy, France, 15-16 November, 2001

Contents

1- The Problem: Real-Time Traffic Scheduling in CAN

2- The Solution: Custom Coprocessor

2.1- Interface

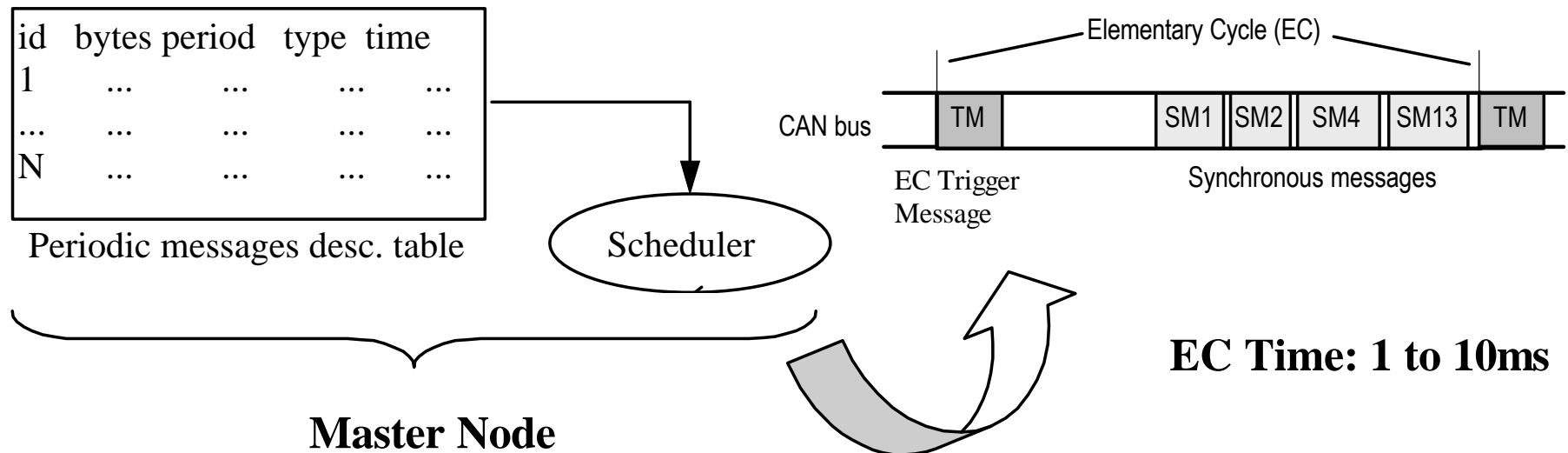
2.2- Architecture

2.3- Performance assessment

1- The Problem: Traffic Scheduling in CAN

The FTT-CAN protocol

- Support of flexible time-triggered communication on CAN
- Discrete time - multiples of an elementary cycle (EC)
- Centralised scheduling with distributed arbitration



1- The Problem: Traffic Scheduling in CAN

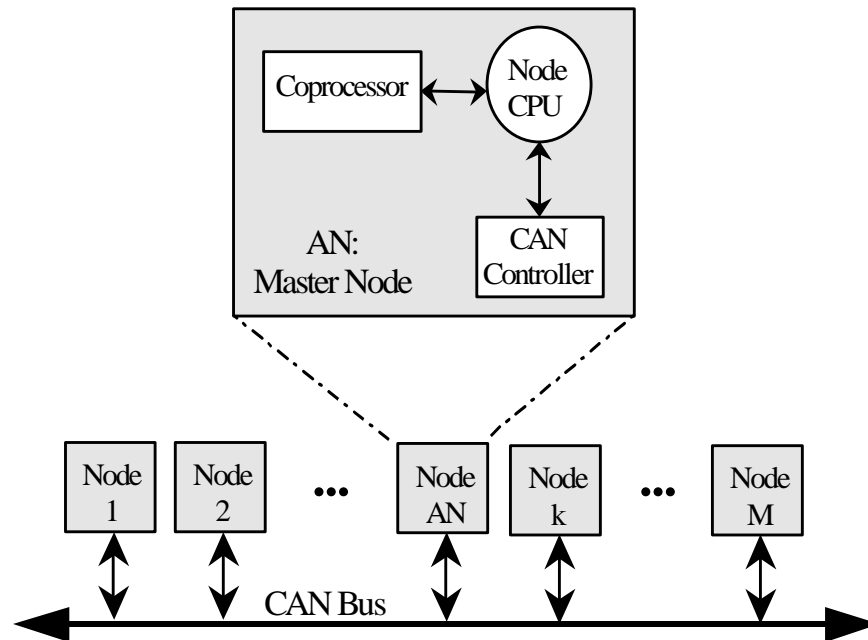
The Scheduler

- **Must meet time constraints and support some level of operational flexibility**

- **Invoking the Scheduler every EC interval**
 - => Great run-time overhead for a low *hp* mC**
 - => Prevents efficient utilization of bus bandwidth**

- **A solution: The Planning Scheduler**
 - **Limitations: Response time, Schedulability analysis**

2- The Solution: A Custom Coprocessor Scheduling Coprocessor



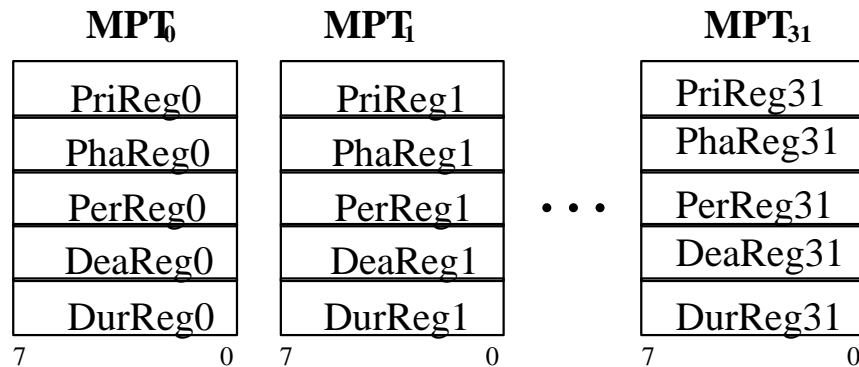
Main Goals:

- **Scheduling on an EC-basis**
- **Allow on-line changes in the message set**
- **Execute schedulability analysis**

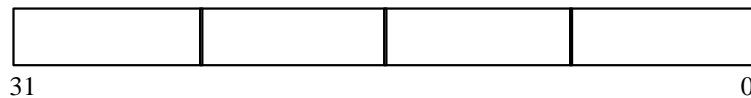
2- The Solution: A Custom Coprocessor

Coprocessor Interface

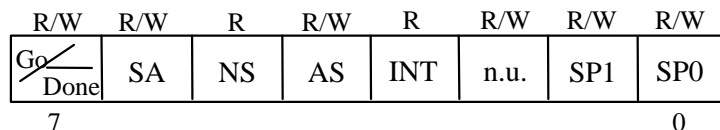
Message's Parameters Register Slots (R/W)



EC-Schedule Register (R)



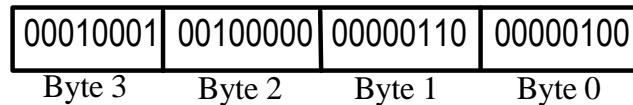
Control / Status Register



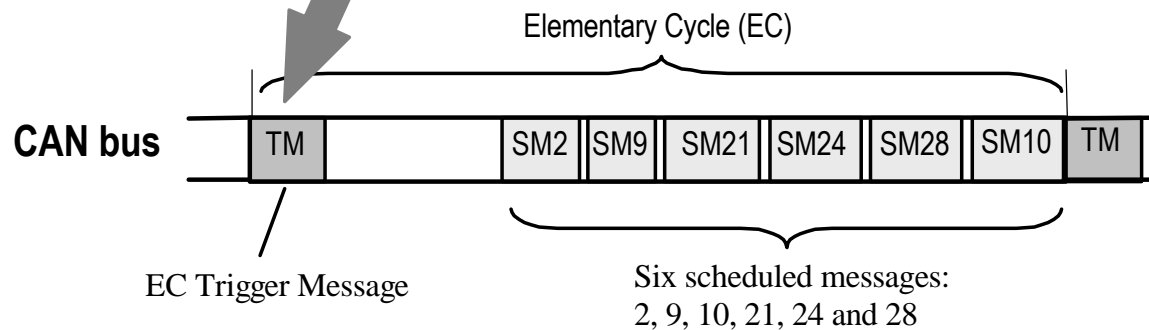
- **32 messages**
- **5, 8-bit parameters/message**
- **Scheduling:**
 - **Rate Monotonic;**
 - **Deadline Monotonic;**
 - **Priority-based.**
- **32-bit EC-Schedule output**
- **Standard interrupt-driven I/F**
- **Scheduler / Analyser Modes**

2- The Solution: A Custom Coprocessor Coprocessor Interface

EC-Schedule Register



To TM data field



EC-Schedule Output

- **FTT-CAN Trigger Message format to minimize dispatching overhead**

2- The Solution: A Custom Coprocessor

Coprocessor Architecture

ECSB

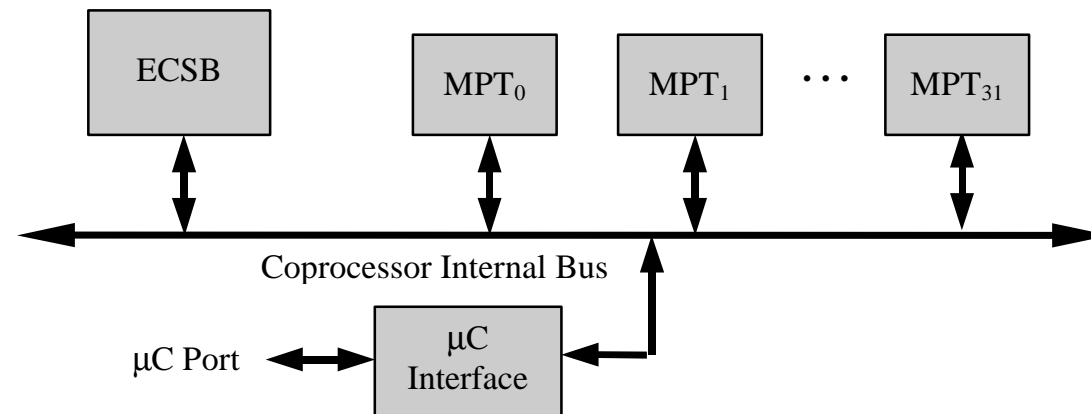
(EC-Schedule Builder):

- Allocates transactions in the ECs;
- Controls schedulability analysis.

MPT

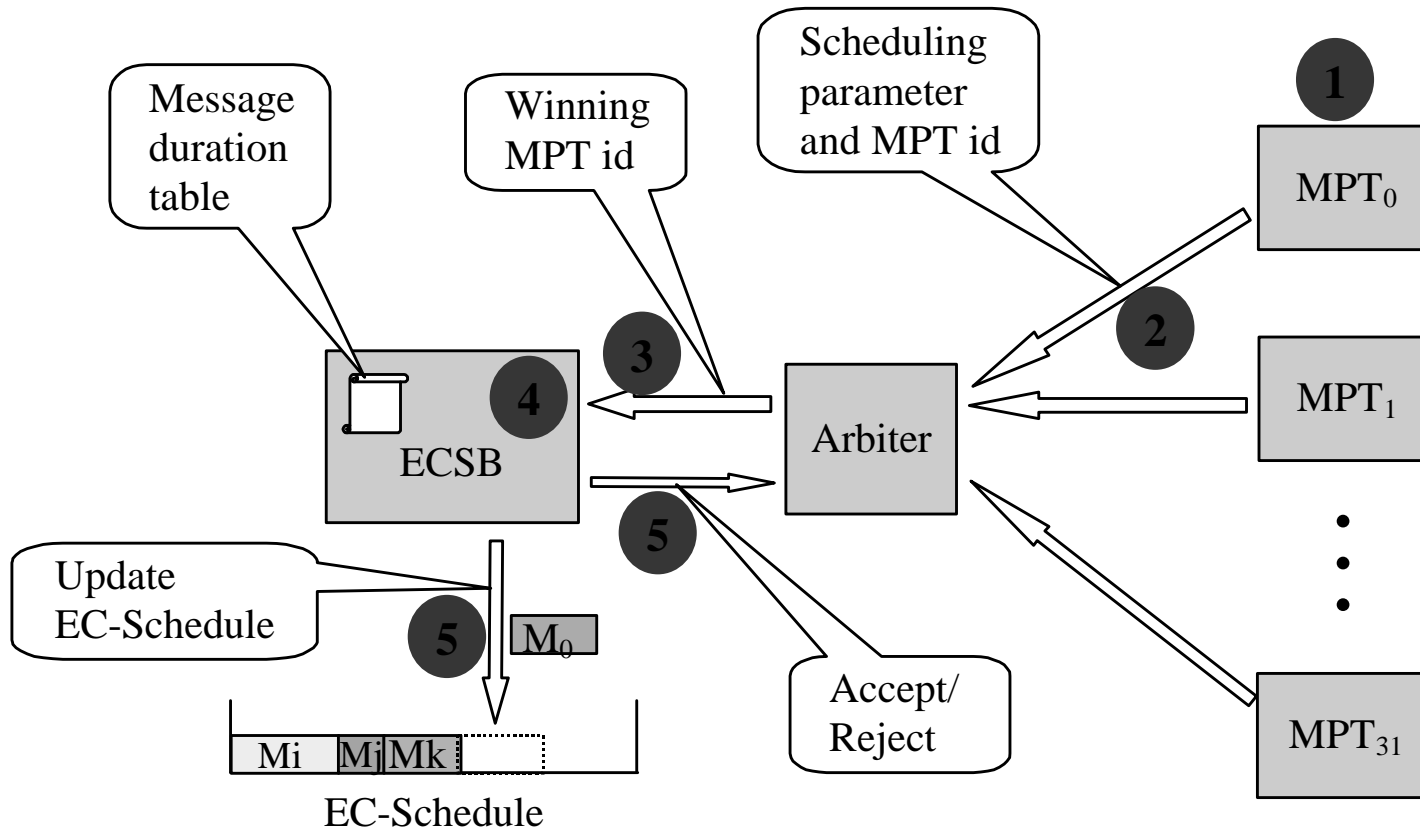
(Message Production Timer):

- Keeps track of each message release time;
- Monitors time-to-deadline during schedulability analysis.



2- The Solution: A Custom Coprocessor

Scheduling operation



2- The Solution: A Custom Coprocessor

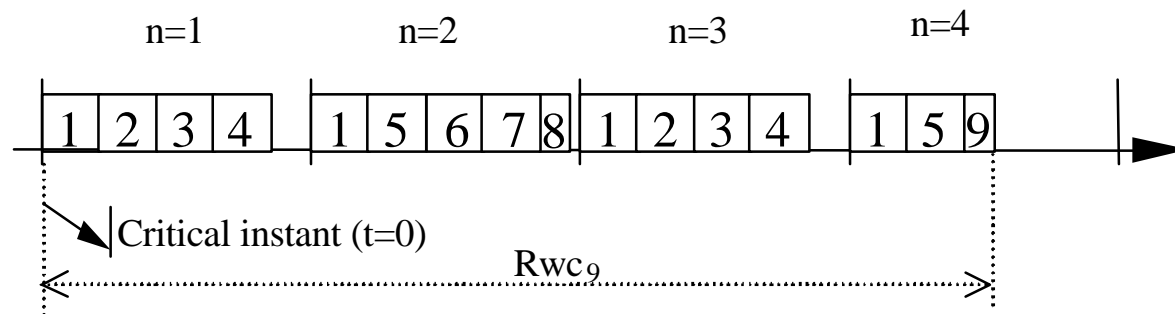
Schedulability analysis

A set of messages ($D_i=P_i$,
 $P_h=0$, $Pri=i$)

i	C_i (ms)	P_i (ms)
1	0.21	1
2	0.21	2
3	0.2	2
4	0.2	2
5	0.2	2
6	0.2	4
7	0.2	4
8	0.14	4
9	0.14	4

➤ Timeline analysis:

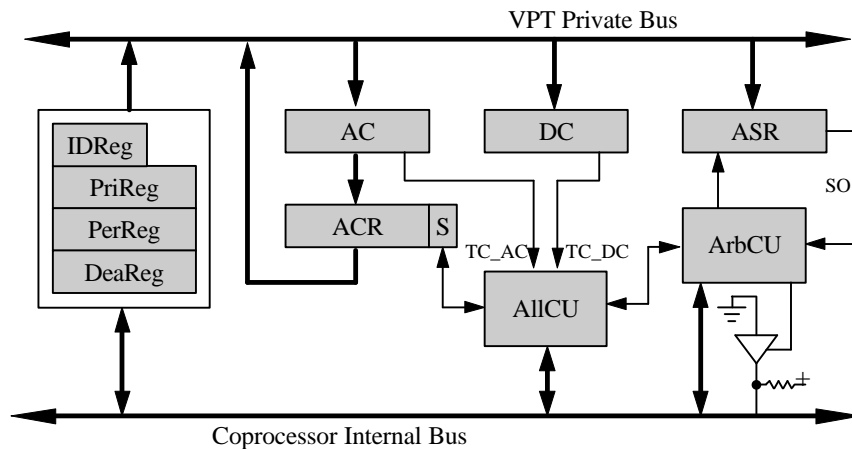
- Construction of the actual schedule from the critical instant ($t=0$) on;
- Verification if, for all messages, Response Time \leq Deadline.



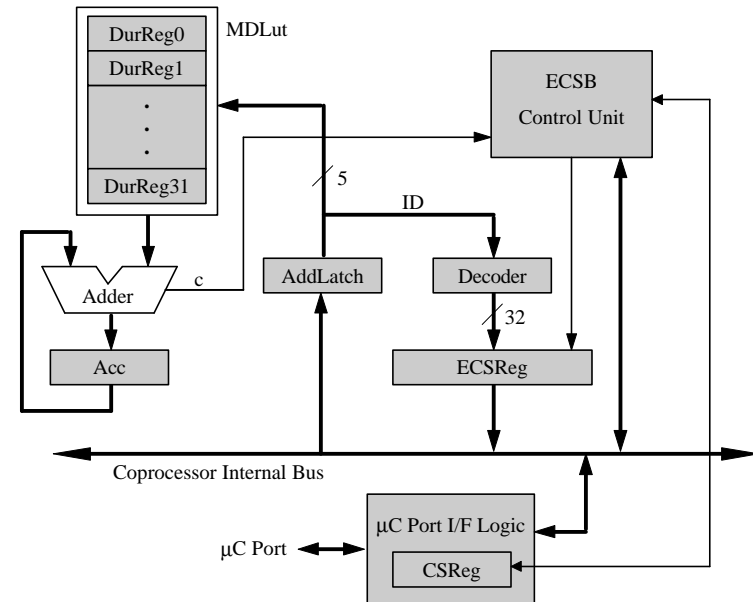
2- The Solution: A Custom Coprocessor

Coprocessor internal architecture

MPT- Message Production Timer



ECSB - EC-Schedule Builder



2- The Solution: A Custom Coprocessor Performance assessment

Requirements:

**Determinism – Scheduling and Schedulability
Analysis (SA) made within EC time.**

**Speed – Scheduling: Maximum availability for SA.
SA: Maximum number of SA tests
(flexibility).**

2- The Solution: A Custom Coprocessor

Performance assessment using SAE

Scheduling:

SAE Benchmark message set

Type	P (ms)	D (ms)	N° of Msgs
A	5	5	8
B	10	10	2
C	50	5	1
D	50	20	30
E	100	100	6
F	1000	1000	6

EC time set to 5ms

- Scheduling time: $t_{sch} = 3 + 16 \cdot N_v$
- Worst-case scenario: All 32 messages allocated in the same EC
 - $t_{sch} = 26\text{ms}$ (@ 20MHz)
- Scheduling done in 0,5% of EC time
=> 99,5 % available for SA

2- The Solution: A Custom Coprocessor

Performance assessment using SAE

Scheduling:

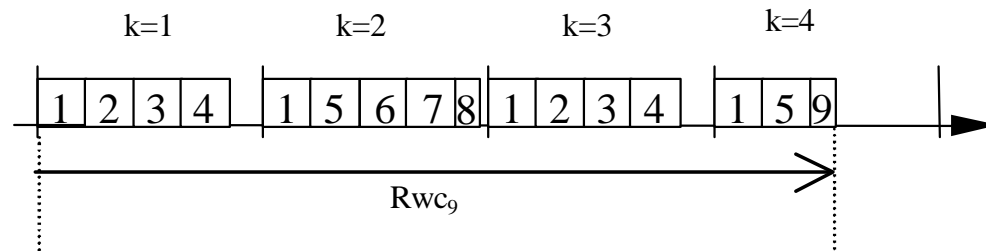
- **For a 1ms EC and assuming we could schedule all 53 SAE messages:**
 - **$t_{\text{sch}} = 42\text{ms}$ (@ 20MHz) => 4,2% of EC time**

- **80C592 vs Coprocessor (9-message set, @ 11MHz)**
 - **80C592: 7800 ms**
 - **Coprocessor: 13,4 ms**

2- The Solution: A Custom Coprocessor

Performance assessment using SAE

Schedulability Analysis (SA):



SA execution time:

$$t_{sa} = 5 \cdot k + 16 \cdot \sum_{i=1}^k Nv(EC_i)$$

- **Worst t_{SA} : Coprocessor generates largest number of fully populated ECs**
 - => **It happens if we try to add a 1000ms-deadline message with all ECs fully occupied (set becomes non-schedulable)**
- **$t_{SA} = 1,65\text{ms}$ (@ 20MHz)**
- **Coprocessor can do at least 3 SAs per EC**

Conclusions

- **Custom scheduling coprocessor supporting operational flexibility with low response time.**
- **Dynamic EC-based scheduler; on-line changes in the message set; timeline-based schedulability analysis.**
- **Additional features: 32 messages, 3 scheduling policies, universal CPU interface, can be used in other centralised-scheduling fieldbuses.**
- **Fast scheduling: utilisation less than 5%**
- **SA: analysis using SAE guarantees 3 tests per EC.**
Determinism: CPU can determine a priori WCET of SA