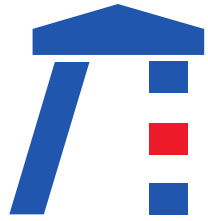
Integration of FlexRay into the SDL-Model-Driven Development Approach

Tobias Braun, Matthias Wiebel, Reinhard Gotzhein
{tbraun,m_wiebel,gotzhein}@cs.uni-kl.de
http://vs.cs.uni-kl.de

Networked Systems Group



TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

04.10.2010

Outline

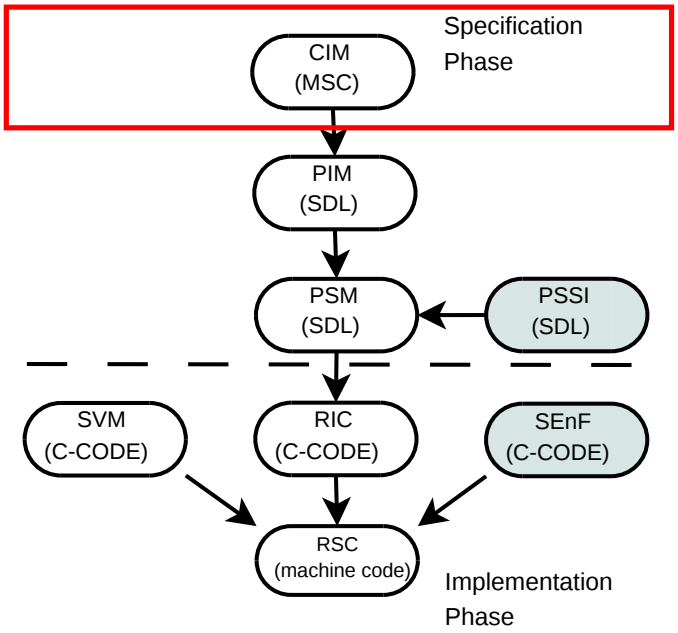
Motivation – Why Integrate FlexRay?

- FlexRay is a recent field-bus technology
  - Realtime-capable with deterministic guarantees
  - Complex protocol
  - Highly configurable

- The SDL-MDD supports abstraction from hardware platforms
  - Abstract signal interfaces on design level
  - Driver support on implementation level

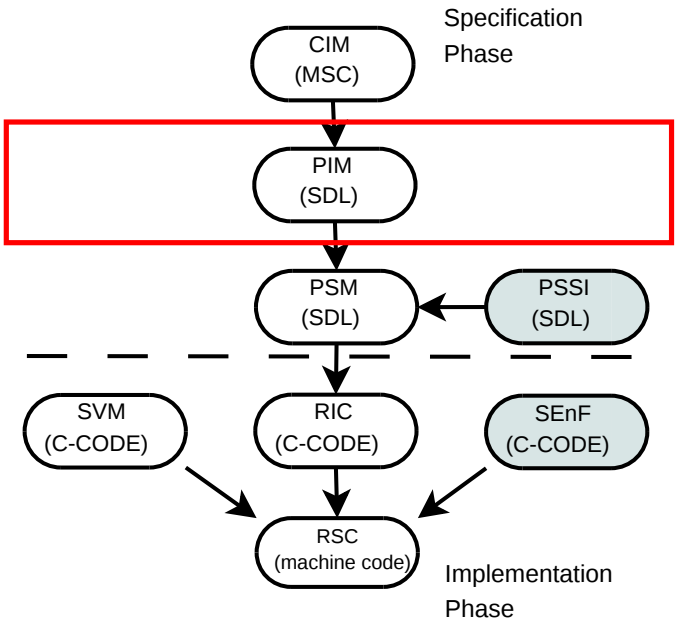
$\longrightarrow$ Integration of FlexRay in the SDL-MDD

SDL-MDD – Model-Driven Development with SDL



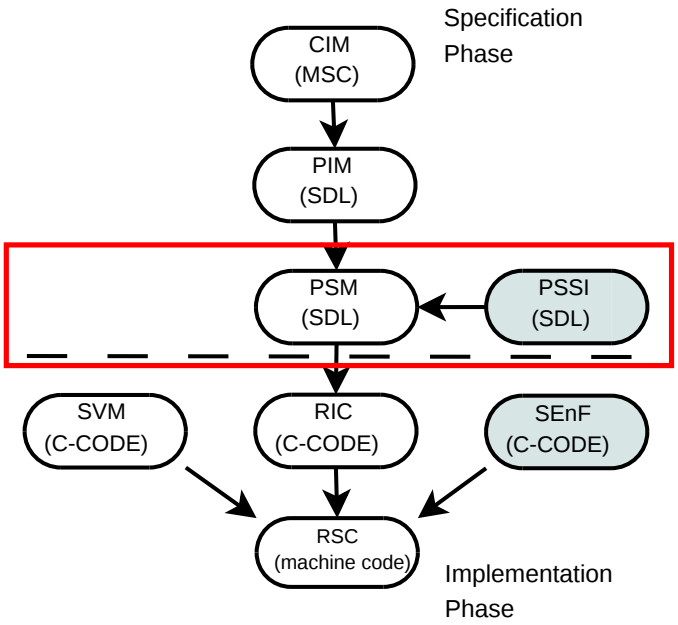
- **C**omputation **I**ndependent **M**odel
- **P**latform **I**ndependent **M**odel
- **P**latform **S**pecific **S**ignal **I**nterface
- **P**latform **S**pecific **M**odel
- **R**untime **I**ndependent **C**ode
- **S**DL **V**irtual **M**achine
- **S**DL **E**nvironment **F**ramework
- **R**untime **S**pecific **C**ode

SDL-MDD – Model-Driven Development with SDL



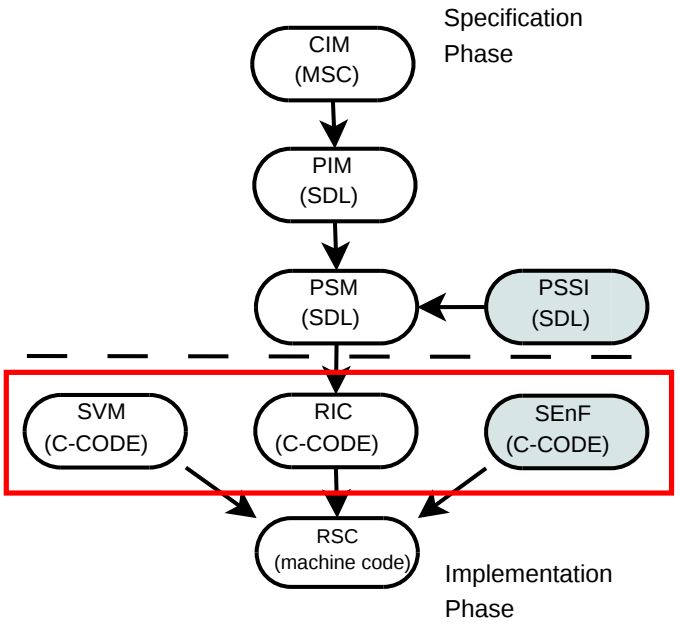
- Computation Independent Model
- **Platform Independent Model**
- Platform Specific Signal Interface
- Platform Specific Model
- Runtime Independent Code
- SDL Virtual Machine
- SDL Environment Framework
- Runtime Specific Code

SDL-MDD – Model-Driven Development with SDL



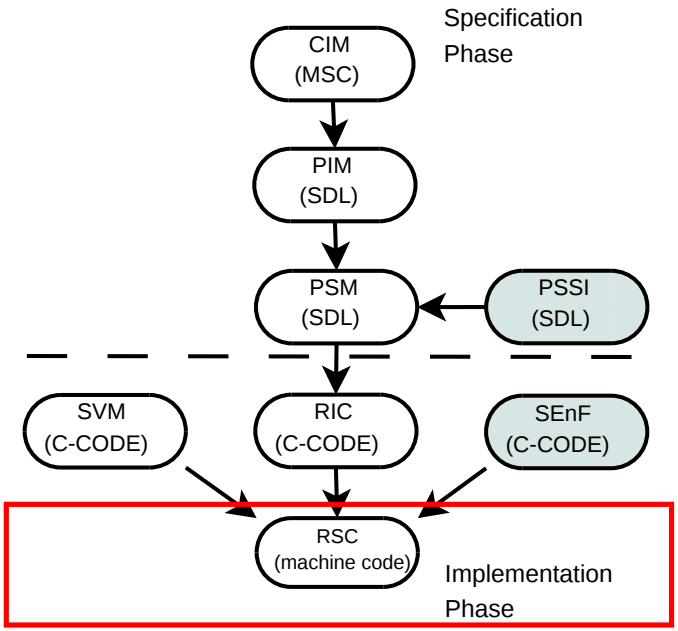
- Computation Independent Model
- Platform Independent Model
- **Platform Specific Signal Interface**
- **Platform Specific Model**
- Runtime Independent Code
- SDL Virtual Machine
- SDL Environment Framework
- Runtime Specific Code

SDL-MDD – Model-Driven Development with SDL



- Computation Independent Model
- Platform Independent Model
- Platform Specific Signal Interface
- Platform Specific Model
- **R**untime **I**ndependent **C**ode
- **S**DL **V**irtual **M**achine
- **S**DL **E**nvironment **F**ramework
- Runtime Specific Code

SDL-MDD – Model-Driven Development with SDL



- Computation Independent Model
- Platform Independent Model
- Platform Specific Signal Interface
- Platform Specific Model
- Runtime Independent Code
- SDL Virtual Machine
- SDL Environment Framework
- **Runtime Specific Code**

FlexRay – Introduction

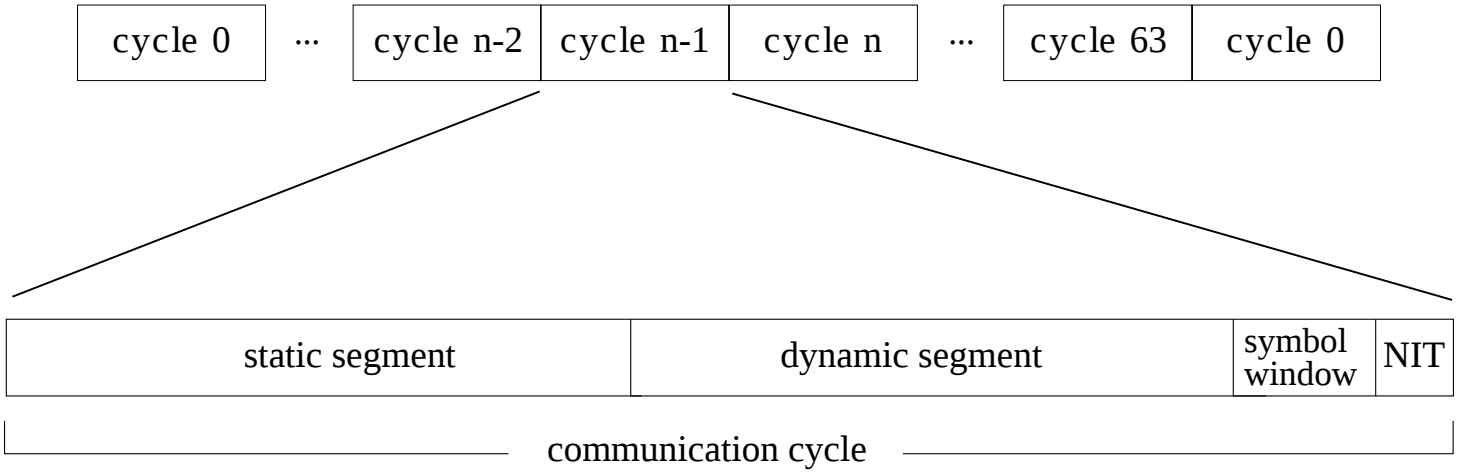
- Recent Field Bus technology in the Automotive Domain
- Developed by the FlexRay Consortium from 2000-2009
  - Release of Version 2.1A in 2005
- Partners of the Consortium:
  - BMW AG, DaimlerChrysler AG
  - Freescale, Bosch, etc.

FlexRay – Introduction cont.

- Two independent physical channels
  - Up to 10 Mbit/s of bandwidth each
  - Redundant transmission possible

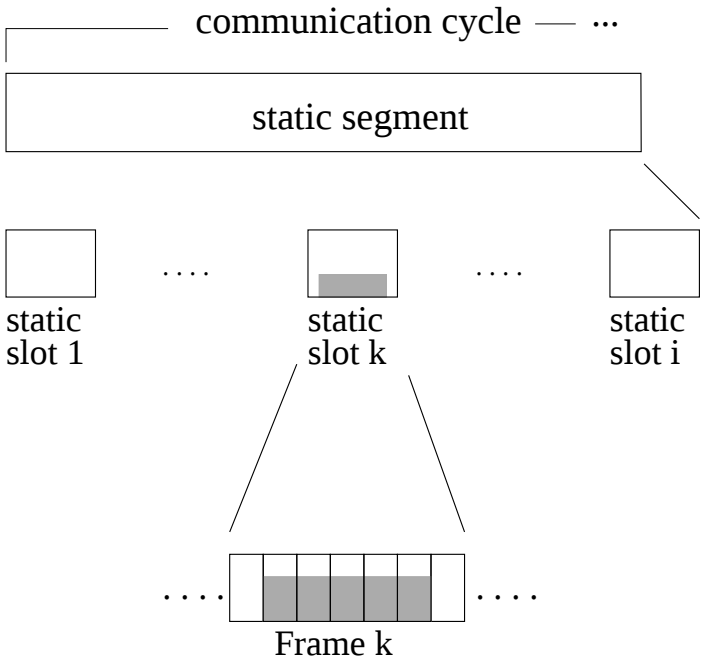
- Time Division Multiple Access (TDMA)
  - Deterministic guarantees concerning frame latency and jitter
  - Real-time capable

FlexRay – Communication Cycle



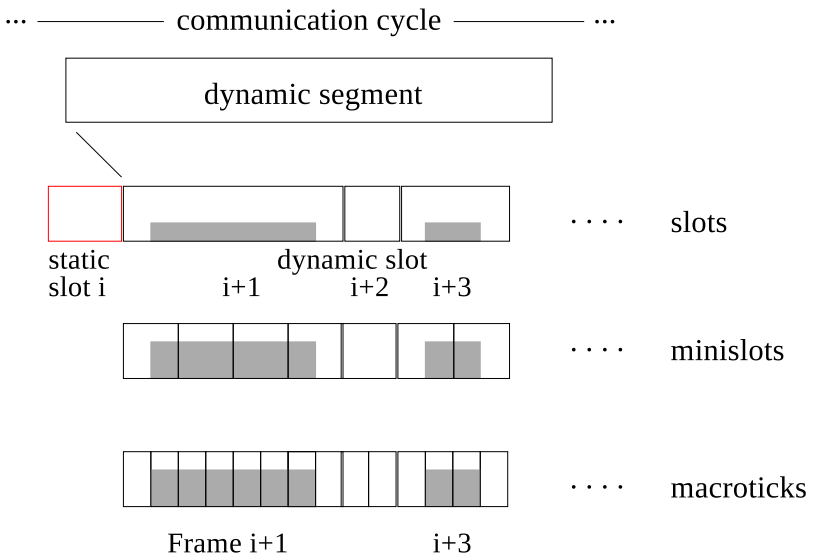
- Static segment allows time guarantees
- Contenion-based dynamic segment

FlexRay – Static Segment



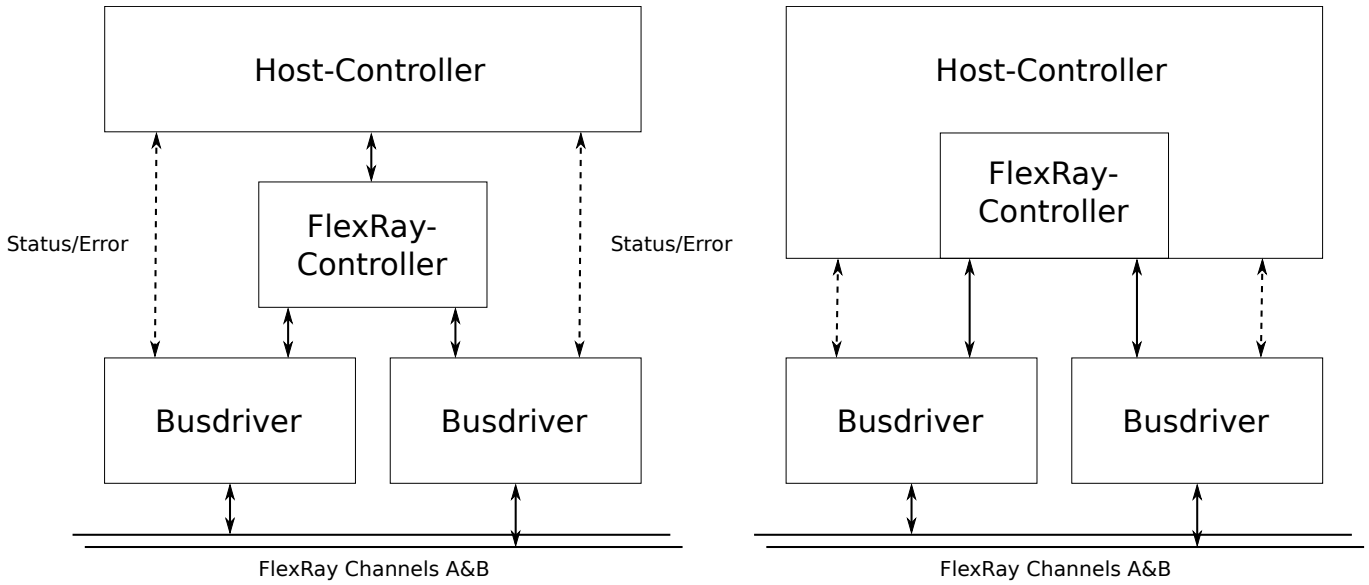
- Subdivded into a configurable, but fixed number of static slots
  - Length is configured in macroticks (cluster-wide time unit)
  - Slots and Frames have a unique ID
  - Slots are assigned to at most one node
- Each node has a key slot
  - Used for startup by coldstart nodes
  - Used for sync-messages by sync nodes

FlexRay – Dynamic Segment



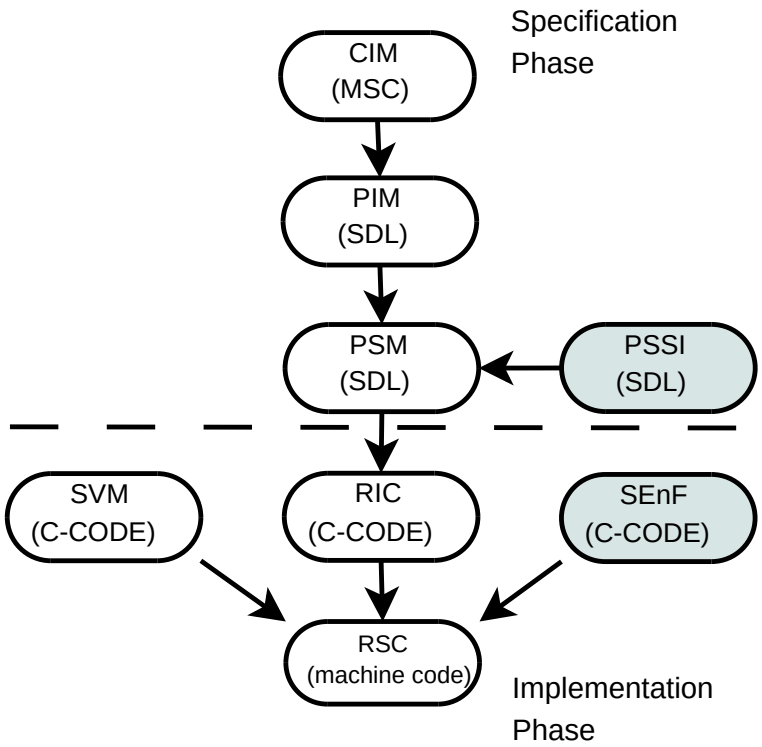
- Contention-based segment
- Length in **minislots**
- Minislots build up **dynamic slots** at runtime
- Medium arbitration follows a minislotting mechanism

FlexRay – Setup of the Nodes



- CC provides free configurable message RAM
  - Send/receive buffers
  - Communication Cycle multiplexing

SDL-FlexRay Interface – Overview



- SDL-FlexRay PSSI
- SDL-FlexRay Driver of the SEnF

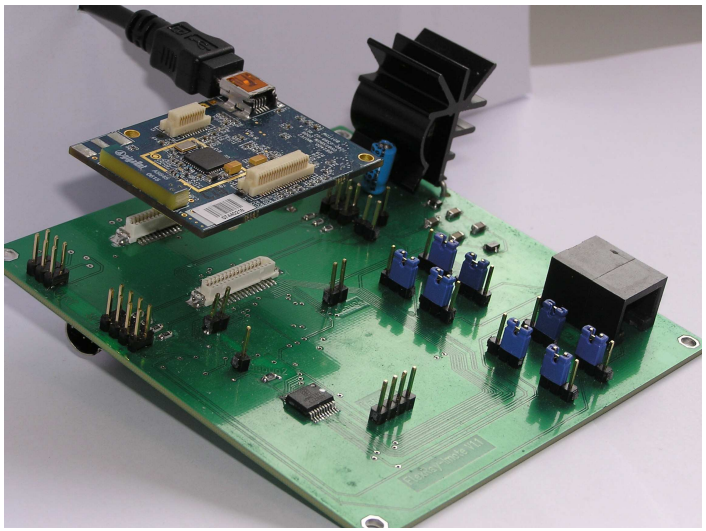
SDL-FlexRay Interface – PSSI

- Startup signals
  - `FLEXRAY_startup`
  - `FLEXRAY_startupFinished`
- Send signals
  - `FLEXRAY_send`
  - `FLEXRAY_ppSend`
  - `FLEXRAY_sendFinished`
- Receive signals
  - `FLEXRAY_recv`
- Buffer configuration
  - `FLEXRAY_cfgBuffer`
  - `FLEXRAY_cfgFinished`

SDL-FlexRay Interface – SEnF Part

- Consists of functions to handle the communication
  - Incoming and outgoing SDL signals
  - Received interrupts
  - Interaction with the CC
- Buffering data for the SDL-System on the host and for the CC

Application Scenario – Hardware setup

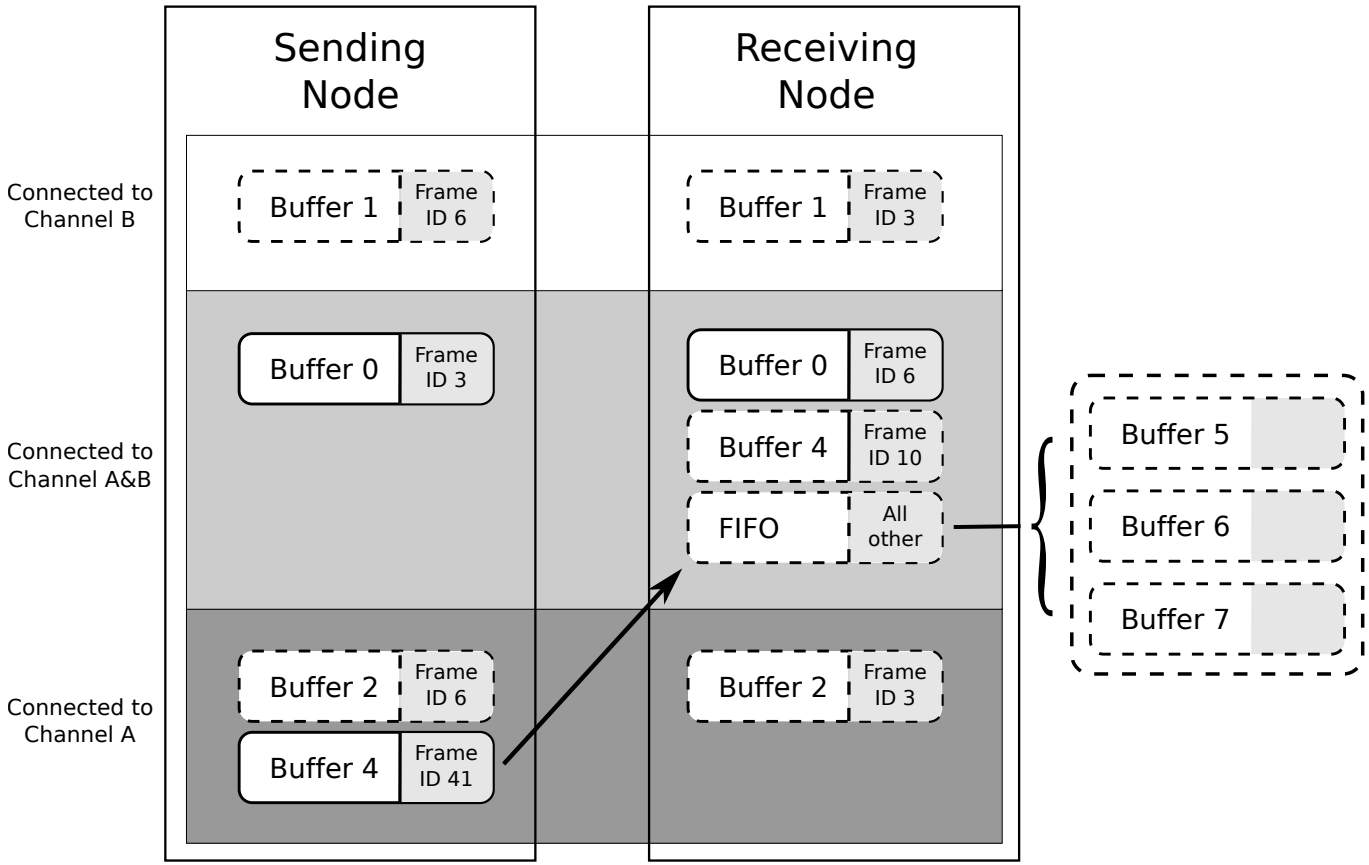


- Host: Imote2
- CC: Fujitsu Siemens MB88121C based on Bosch E-Ray IP-Module
- Bus drivers: AS8221D by Austriamicrosystems

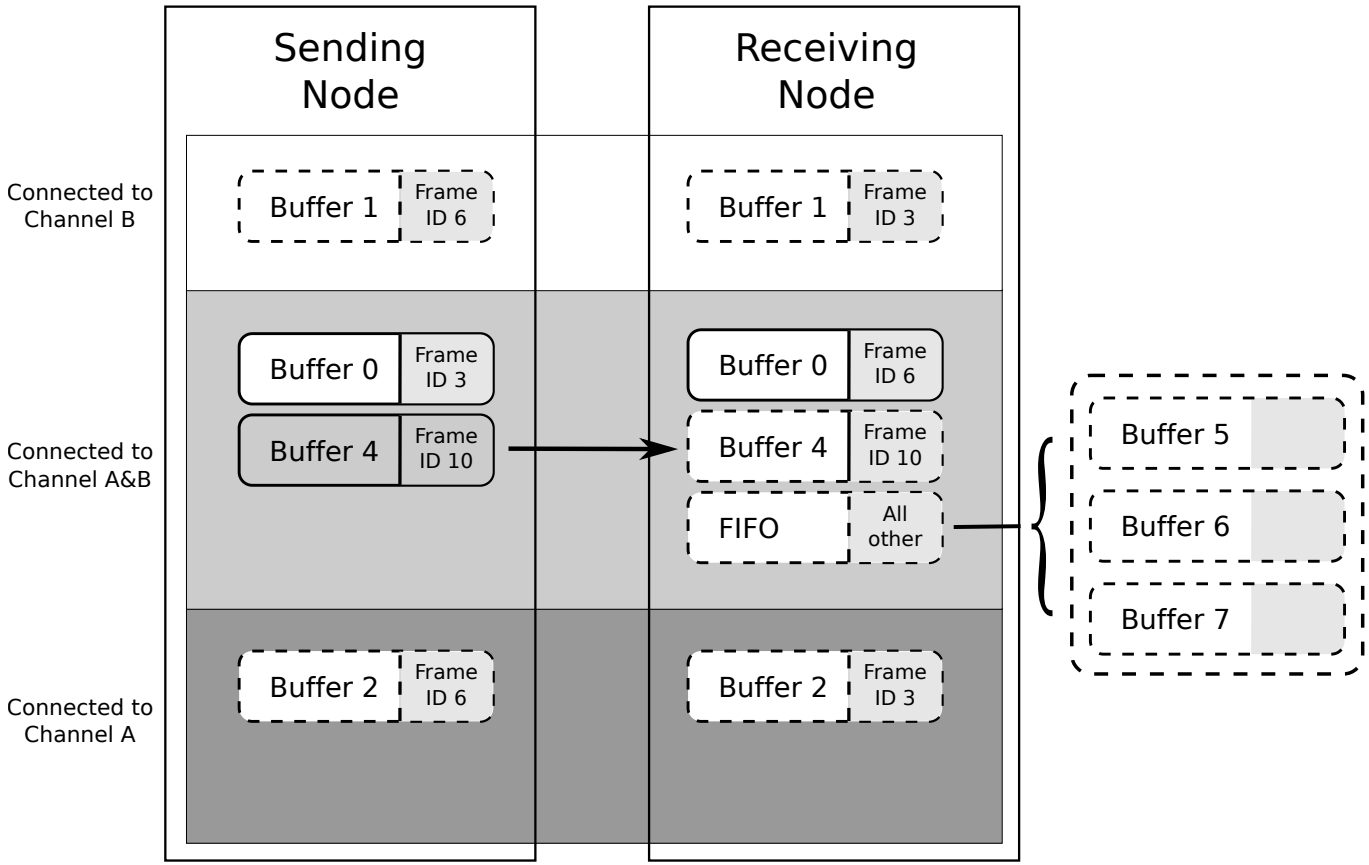
Application Scenario – Experiment Flow

- Cluster consists of two FlexRay nodes
- Sending node
  - 50 ping messages with ID 41 (dynamic segment)
  - Another 50 messages with ID 10 (static segment)
- Receiving node
  - Receives the first 50 messages over the FIFO
  - Receives the second 50 messages over buffer 4
- Scenario is run twice
  - First run: normal configuration
  - Second run: channel *A* detached

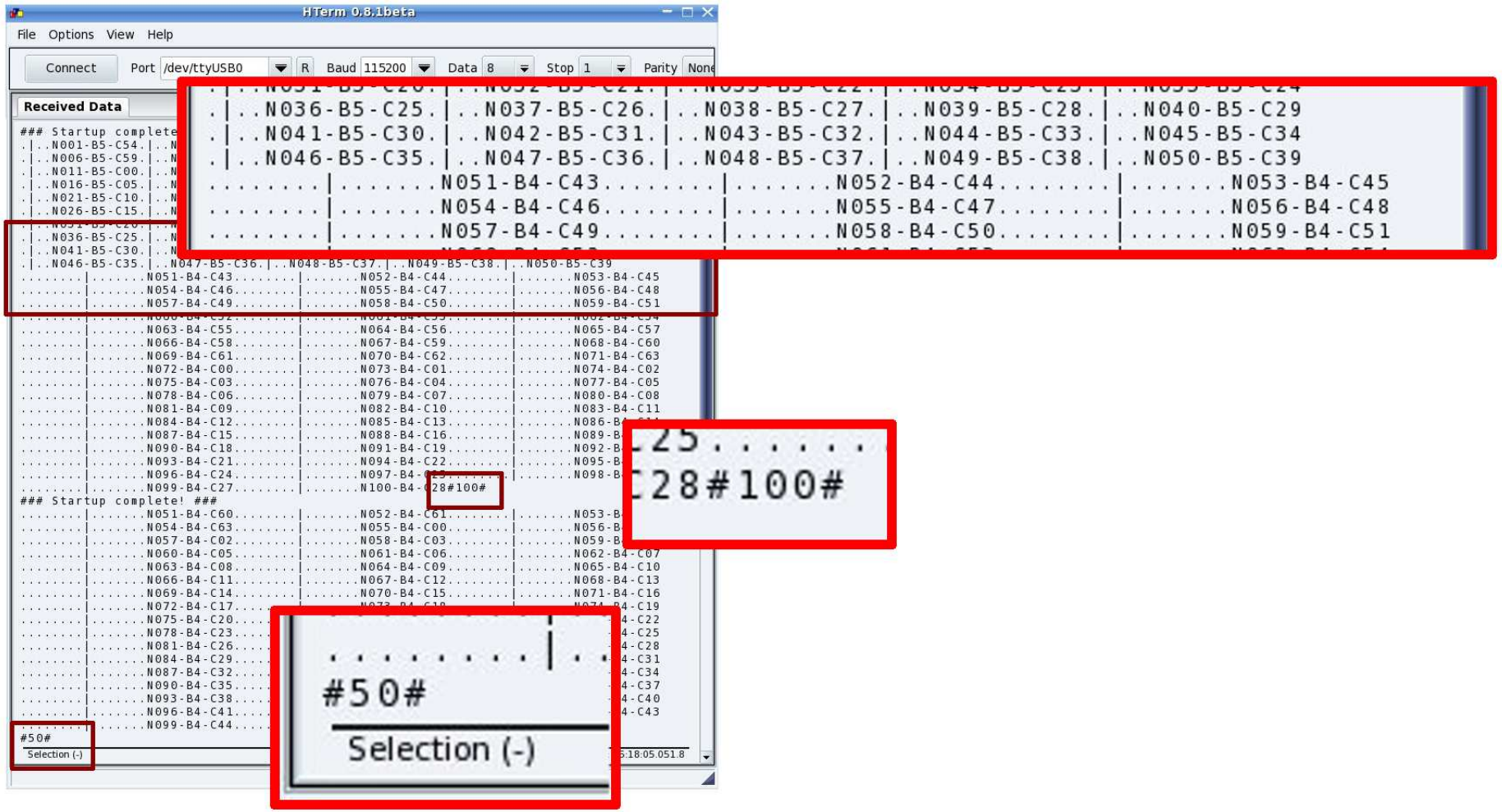
Application Scenario – Message Buffer Configuration



Application Scenario – Message Buffer Configuration



Application Scenario – Log Messages of Receiving Node



Conclusion

- Integration of FlexRay into SDL-MDD
- Abstract signal interface on design level (PSSI)
- Extension of our toolchain with a FlexRay driver for SEnF
- Demonstration of the integrated functionalities with an application scenario

Conclusion – Future Work

- Integration of further hardware support into the SEnF driver
  - PC card for debugging purposes
- Research on timing constraints
  - Real-time guarantees
  - Signal transmission delay
- Using the FlexRay PSSI for the generation of simulation code