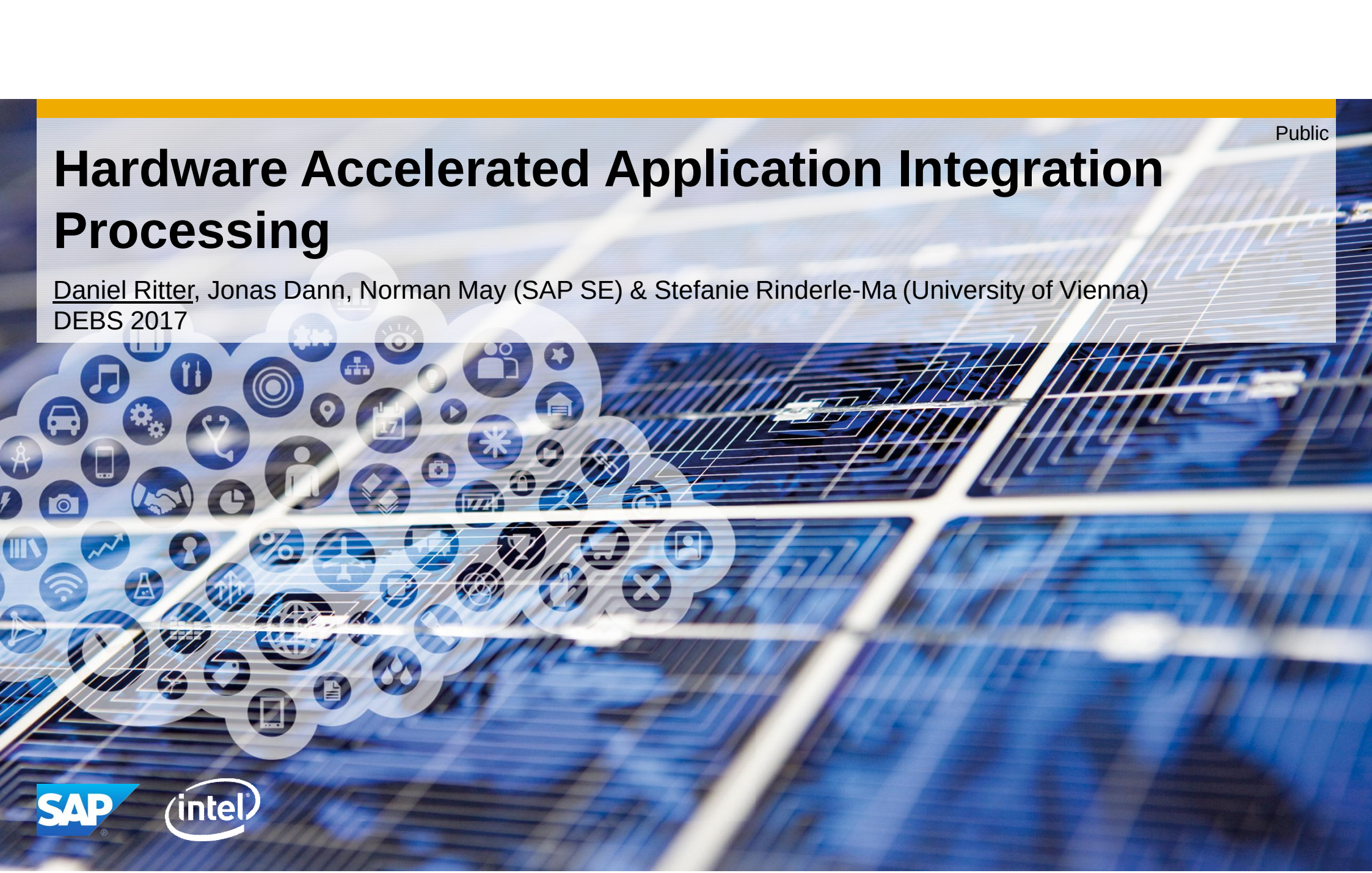
Public

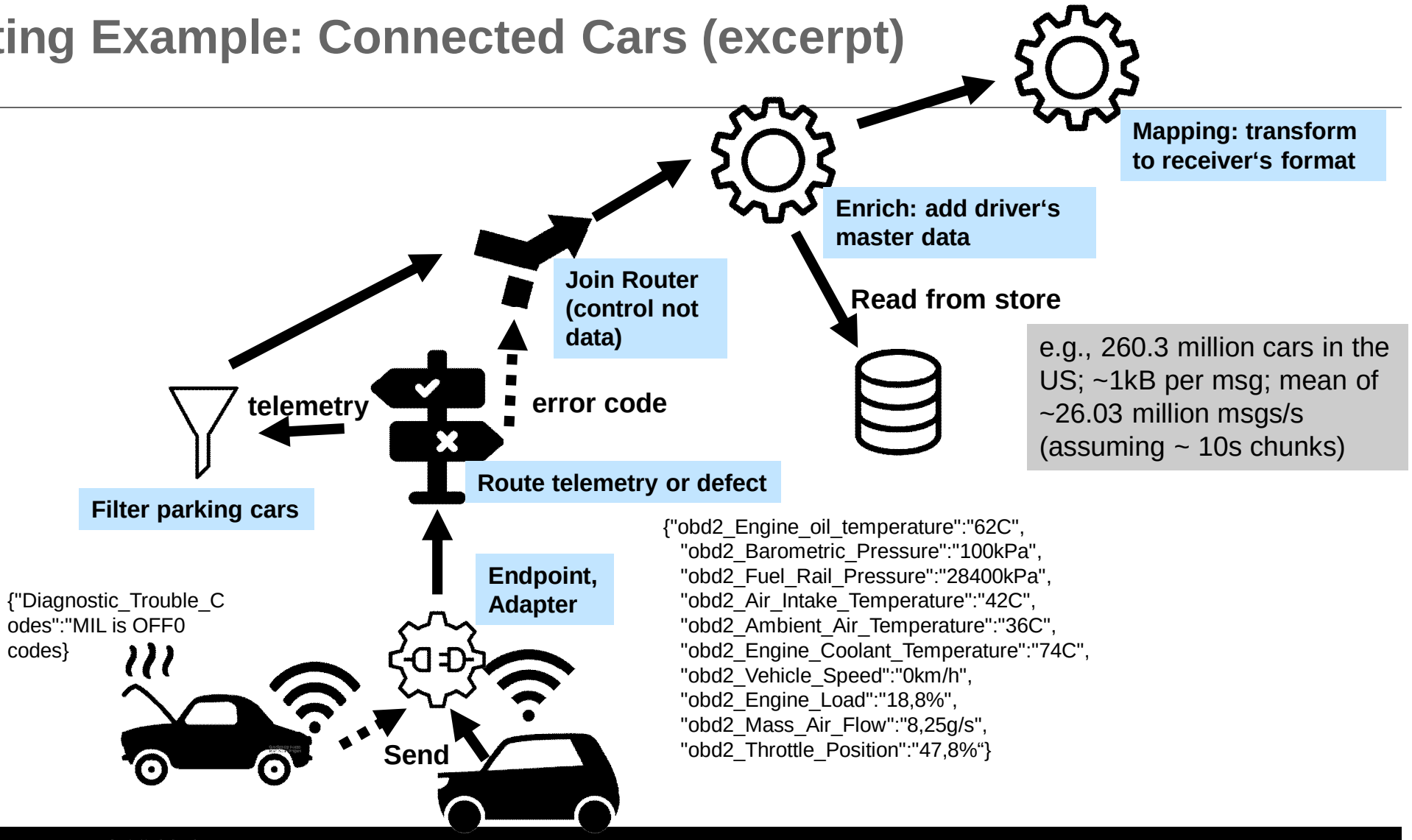
Hardware Accelerated Application Integration Processing

Daniel Ritter, Jonas Dann, Norman May (SAP SE) & Stefanie Rinderle-Ma (University of Vienna)

DEBS 2017



Motivating Example: Connected Cars (excerpt)



Problem Statements and Solution Sketch

P1: Missing design of a fully ``hardware-based integration system“ as message stream pipeline (hardware == FPGA)

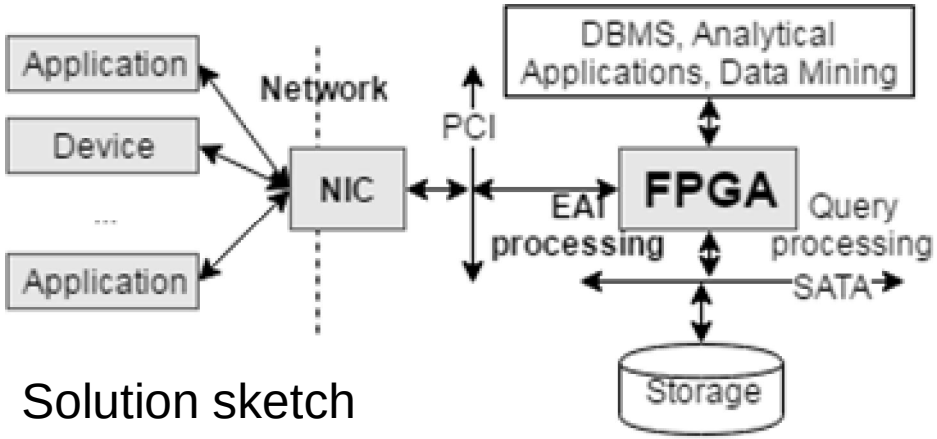
P2: Missing streaming definitions [Zimmermann2016] of Enterpriese Integration Patterns (EIP) [Hohpe2003] (and not on hardware)

P3: Support of condition and expression evaluation on hierarchical data model

P4: Throughput – EIPBench benchmark [DEBS2016b] showed critical impacts on software implementations for

- Branching, evaluation of complex conditions (ie, CBR, Message Filter)
- Threading (i.e., Load Balancer, Parallel Processing)
- Big messages

(P5: Energy efficiency in Data Centers)



Solution sketch

Related Work

Query processing

- Industrial solutions: Netezza, Kickfire for specific data warehousing workloads; immutable at runtime
- Glacier database query to hardware compiler, **asynchronous design** with configurable clock frequencies [Müller2009a, Müller2009b, Müller2010]

Complex Event Processing

- Event detection, regular expression evaluation, sorting network, e.g., [Woods2010]
- XPath evaluation using **finite state automata**, e.g., [Agrawal2008]

Publish/Subscribe and Queuing Systems

- Industrial solution: Solace Message Broker (no CBR)
- Content-based Routing (CBR) Publish/Subscribe System based on FPGA XPath evaluation, e.g., [ElHassan2010]

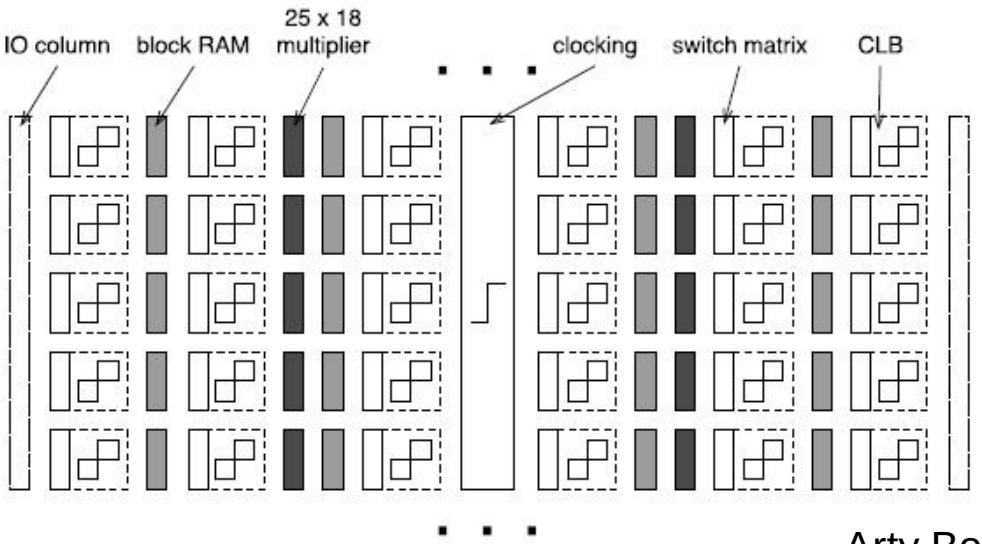
Hardware accelerated EAI processing:

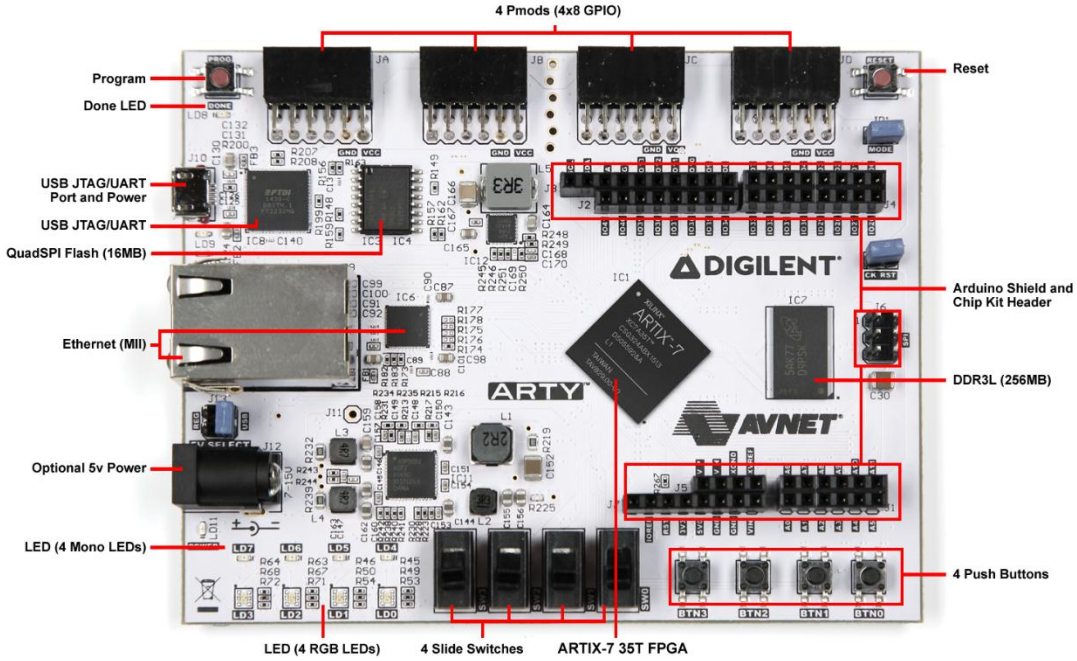
- **Systolic ``pipeline-chain'' parallelism** for higher throughput through good scalability (even across chips); not **MISD** trees with longer signal paths [Caspi2005]
- **Synchronous design**, since configurable clocks are only needed for varying frequencies in integration adapters (e.g., TCP, UDP)
- Use FIFO buffers for **flow control** and backpressure and not for asynchronous design [Caspi2005]
- P1: Design hardware **message processing**
- P2: Define **EIP [Hohpe2003] streaming semantics** and transfers the to FPGA-hardware
- P3: Define hierarchical message format processing for **resource constraint hardware**
- (P4, P5: Evaluate tradeoffs for message throughput and data sizes, parallelism and resource consumption, as well as optimizations for the connected car example)

Integration Systems on FPGAs

System Design and Integration Patterns on Circuits

FPGAs in a Nutshell



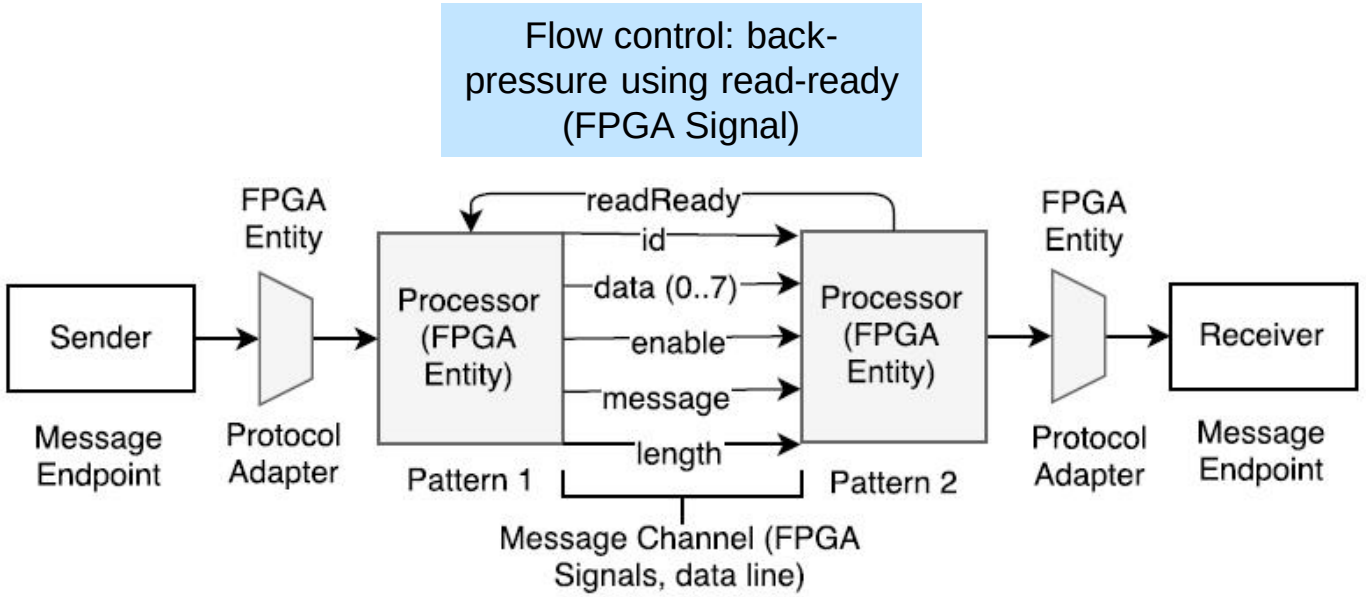


From:
https://www.xilinx.com/content/xilinx/en/products/boards-and-kits/arty/_jcr_content/mainParsys/xilinxtabs2/tab-hardware/xilinxtabs2_6909/tab-board/xilinximage_a593.img.png/1443135118904.png

Arty Board:
Xilinx Artix-7 FPGA
100 MHz clocking
36 kB BRAM blocks (L1-cache)
CLBs: 20,800 LUTs, 41,600 FFs

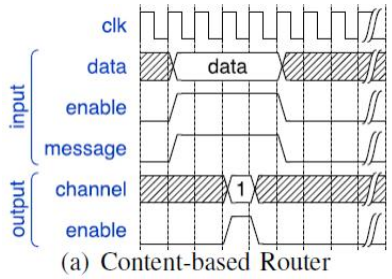
Basic Integration Semantics on Hardware

- Message Endpoint (Sender, Receiver)
- Protocol Adapter (e.g., UDP, TCP)
- Message, Message Channel
- Message Processor

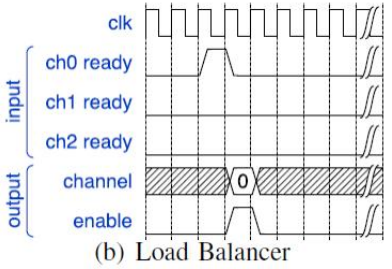


Message Processor Definitions in a Nutshell

- Sliding windows do not fit due to fixed time intervals or tuples vs message boundaries
- We use data-dependent windows similar to Frames [DEBS2016a] for messages
- Pattern representation idea:
  - Identify common characteristics according to the streaming and on-chip definitions: ``Message to Channel“, ``Message to Message“
  - Combine with type of user interaction: ``with user interaction“, ``no user interaction“

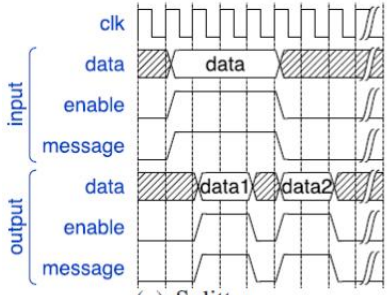


(a) Content-based Router

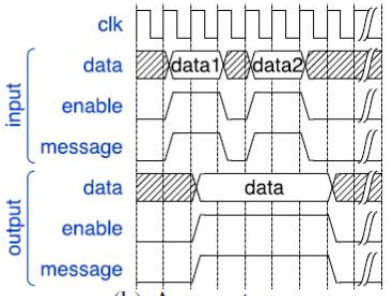


(b) Load Balancer

``Message to Channel“

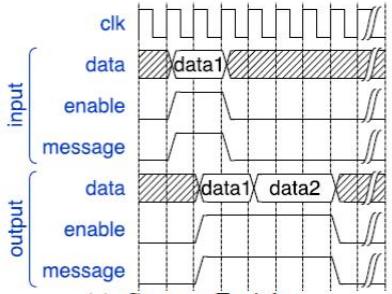


(a) Splitter

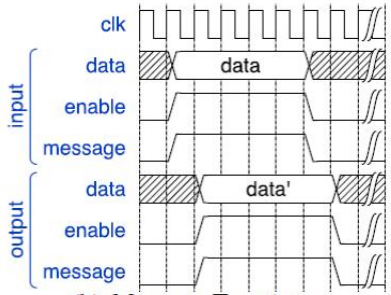


(b) Aggregator

``Message to Message“



(a) Content Enricher



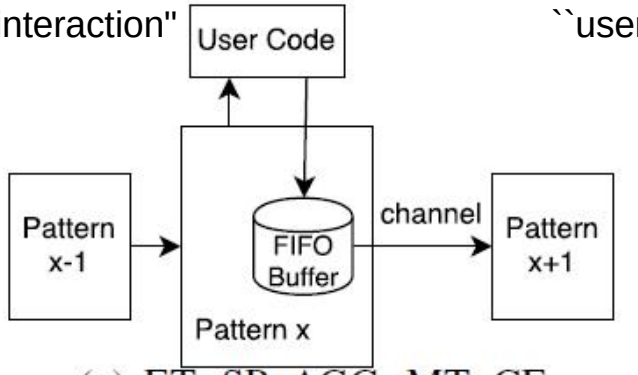
(b) Message Translator

Pattern Templates and Format Conversion

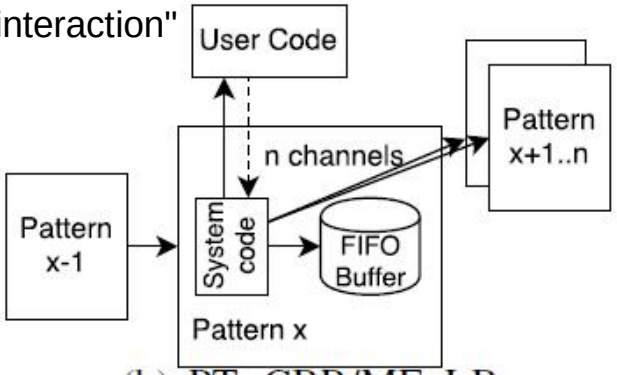
Expression Template (ET)
``Message to Message“,
``user interaction"

Predicate Template (PT)
``Message to Channel“,
``user interaction"

FIFOs for Flow control: back-pressure using read-ready signal

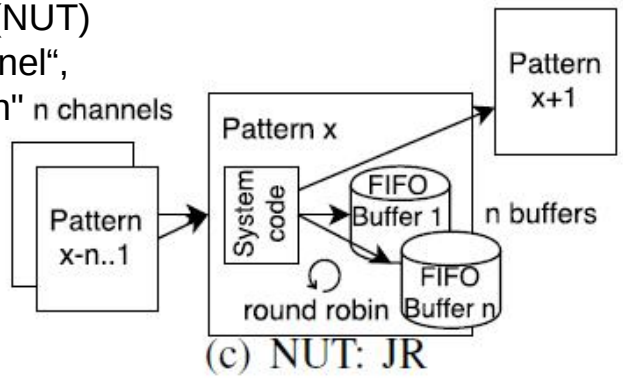


(a) ET: SP, AGG, MT, CE

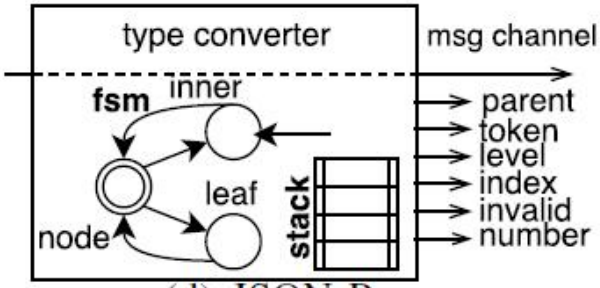


(b) PT: CBR/MF, LB

No User Template (NUT)
``Message to Channel“,
``no user interaction"



(c) NUT: JR



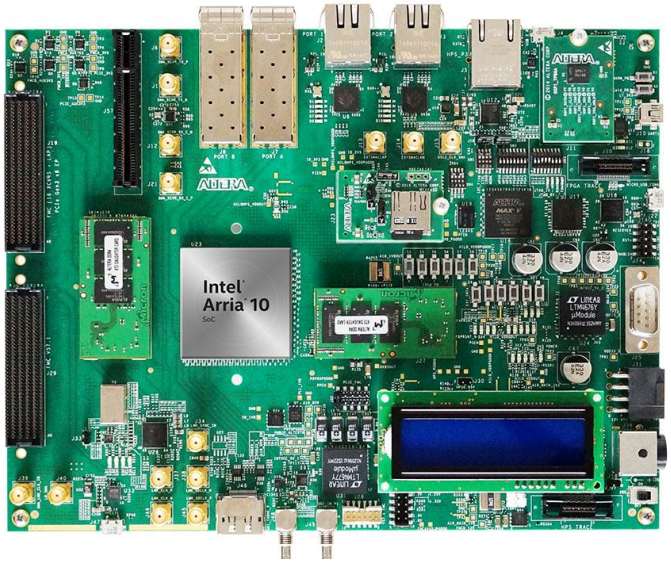
(d) JSON Parser

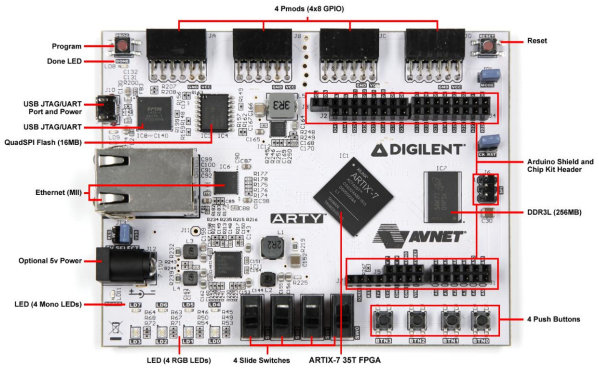
Hiearchical format handling (pushdown-automaton): e.g., signals for parent type {object, array}

Experiments

Message throughput, Message Sizes and CCT Scenario

System Setups & Benchmark



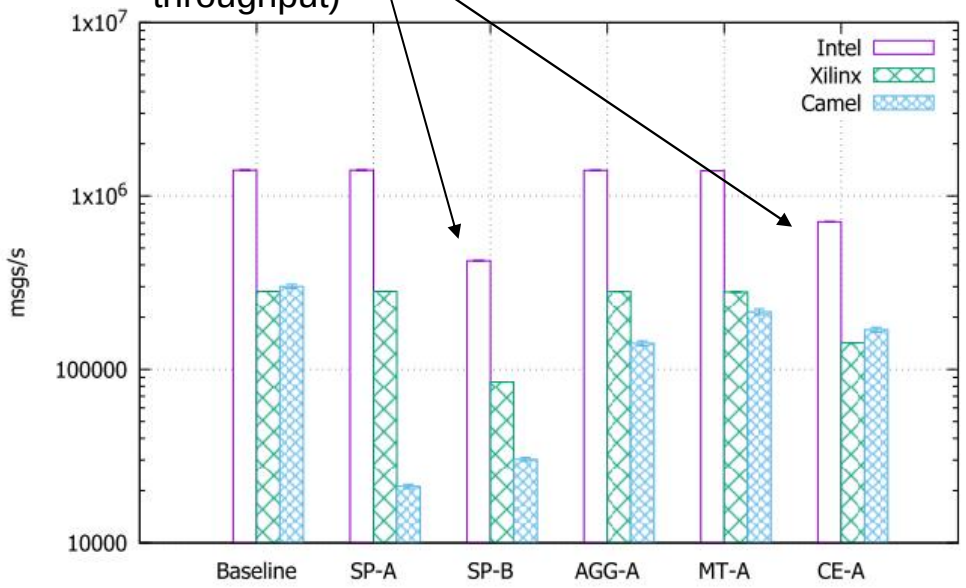


Characteristics	Xilinx XC7A35T	Arria SOC 10	Z600
Clock rate	100 MHz	500 MHz	2.67 GHz
Ethernet speed	10/100 Mbit/s	2x 10 Gbit/s	10/100 Gbit/s
On-chip RAM / on-board DRAM	1,800 kB / 256 MB	39 MB / 1 GB	- / 24 GB

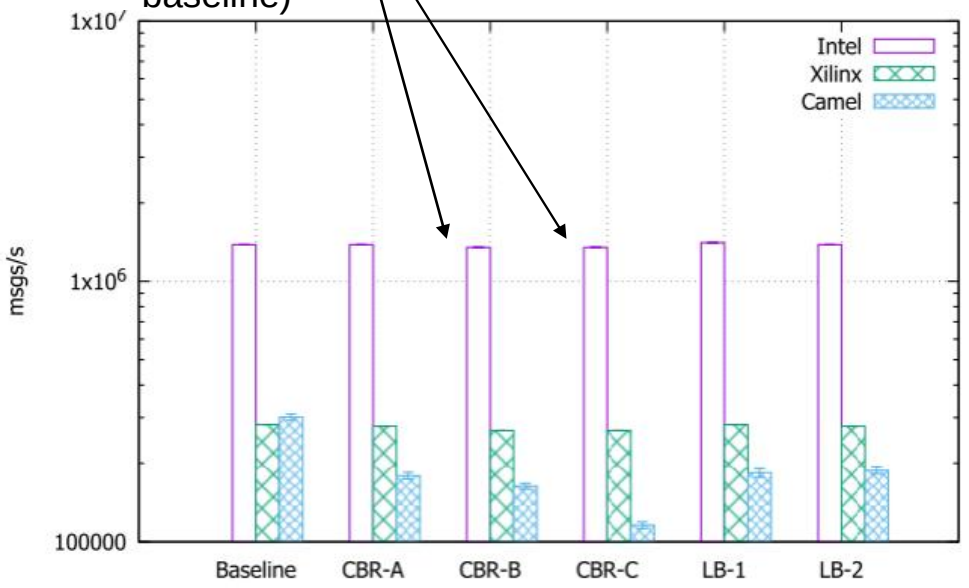
Benchmark: EIPBench [DEBS2016b] for benchmarking integration pattern implementations
Software Integration System: Apache Camel v2.7

Predicate and Expression Template Throughput

ET: The FPGA data generating patterns (e.g.,SP-B, CE) lead to degrading throughput (à message size has a big impact on the throughput)

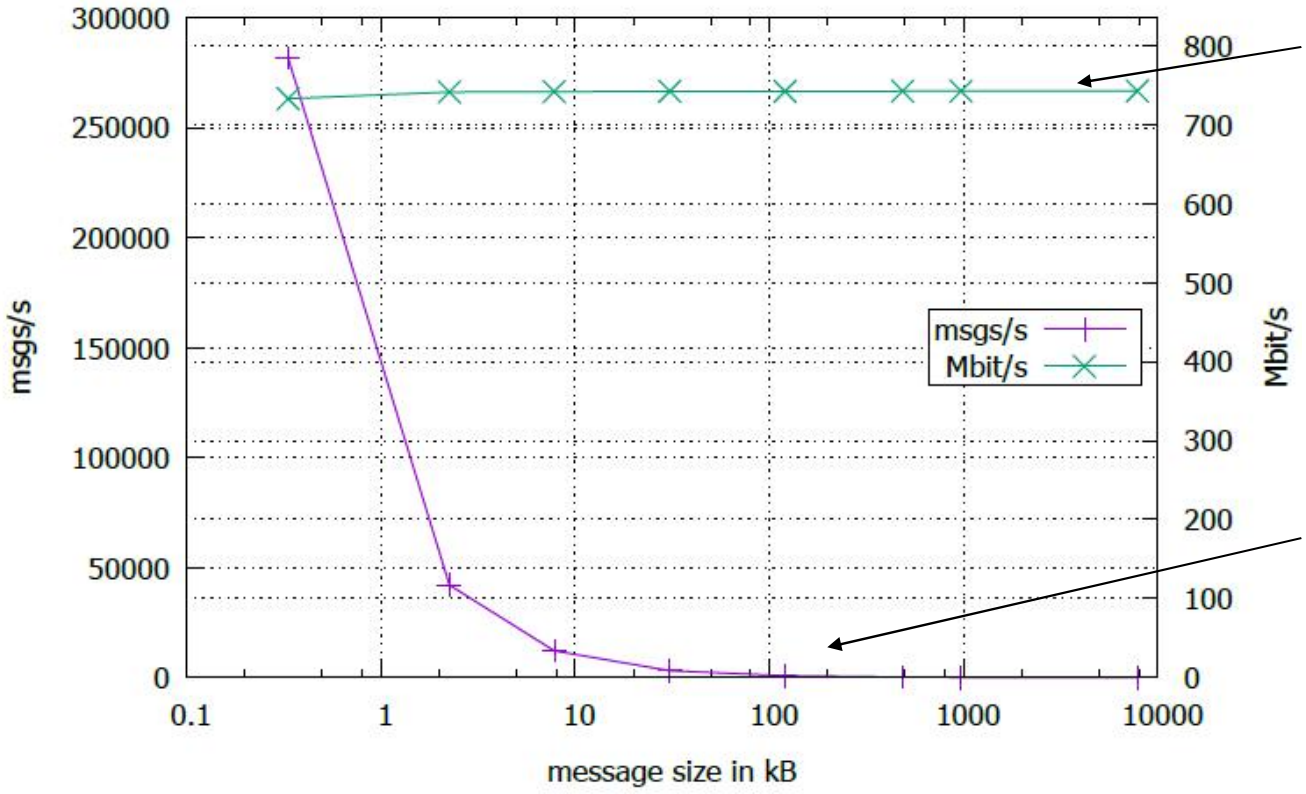


PT: The throughput is invariant to multiple conditions and route branches (e.g, CBR-B+C, LB and join router (not shown) perform near baseline)



Throughput scales factor five with more hardware resources due to factor five higher clocking (cf. small Xilinx with 100MHz vs more production ready Intel FPGA with 500MHz).

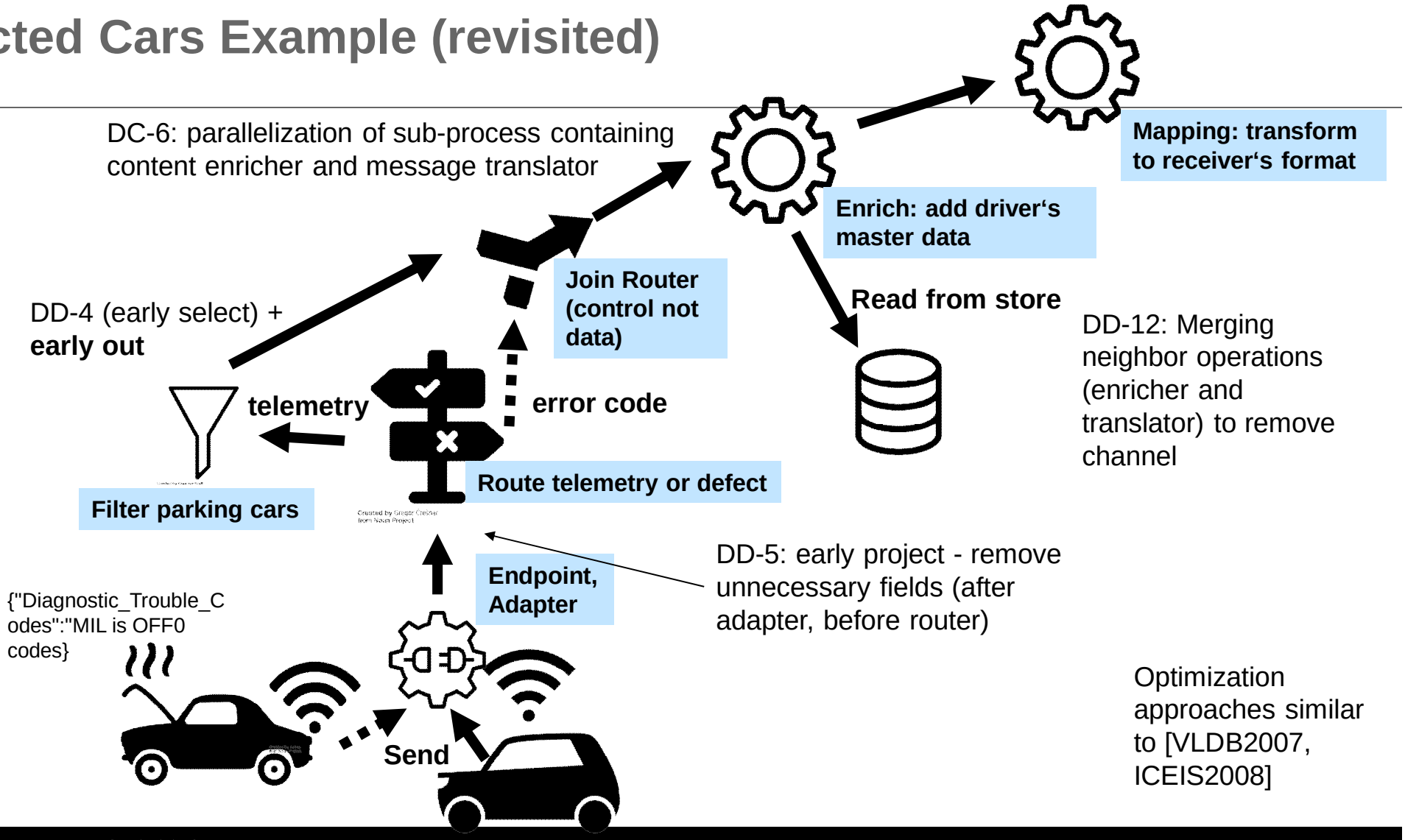
Message size scaling CBR-A (single instance) on Xilinx



The throughput is physically limited by the capacity of the hardware (cf. on-chip memory).

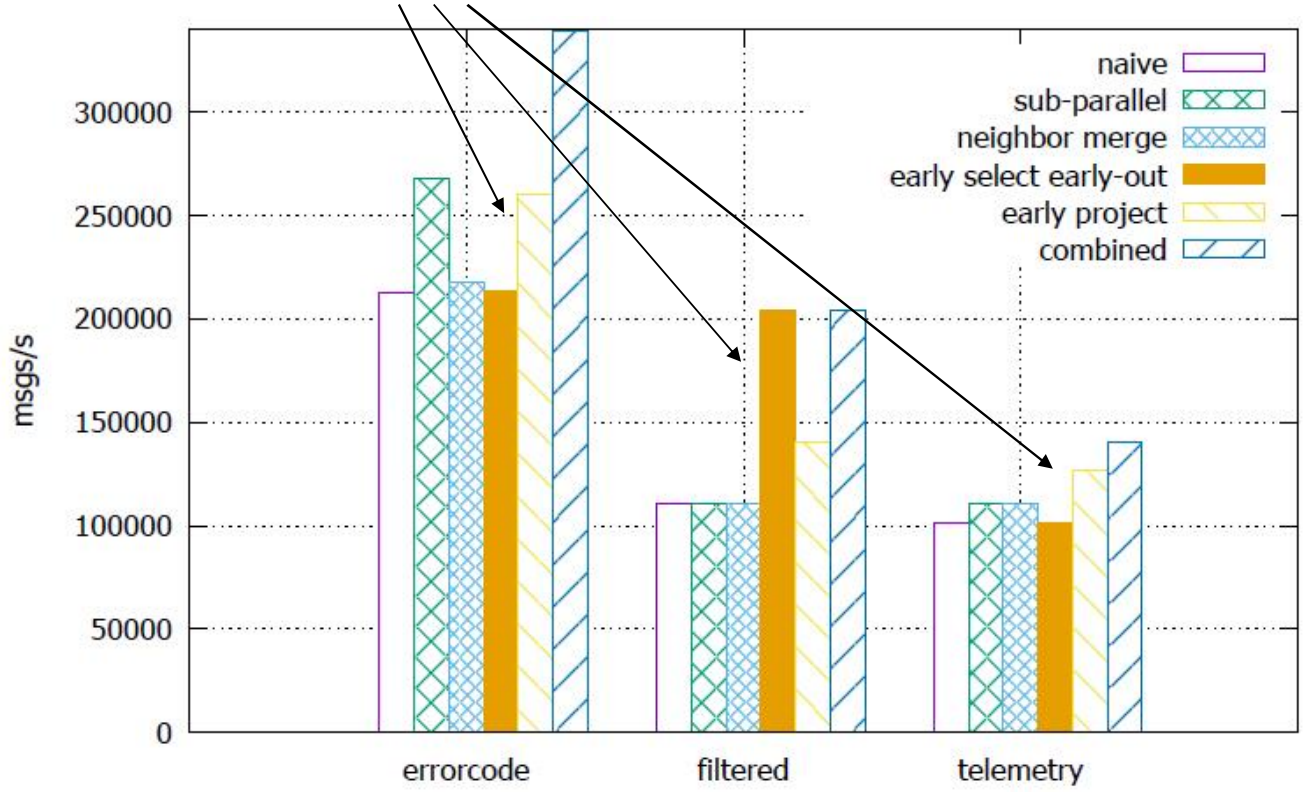
The throughput can be increased by using secondary memory (on-board DRAM), however, trading for more selective data operations.

Connected Cars Example (revisited)



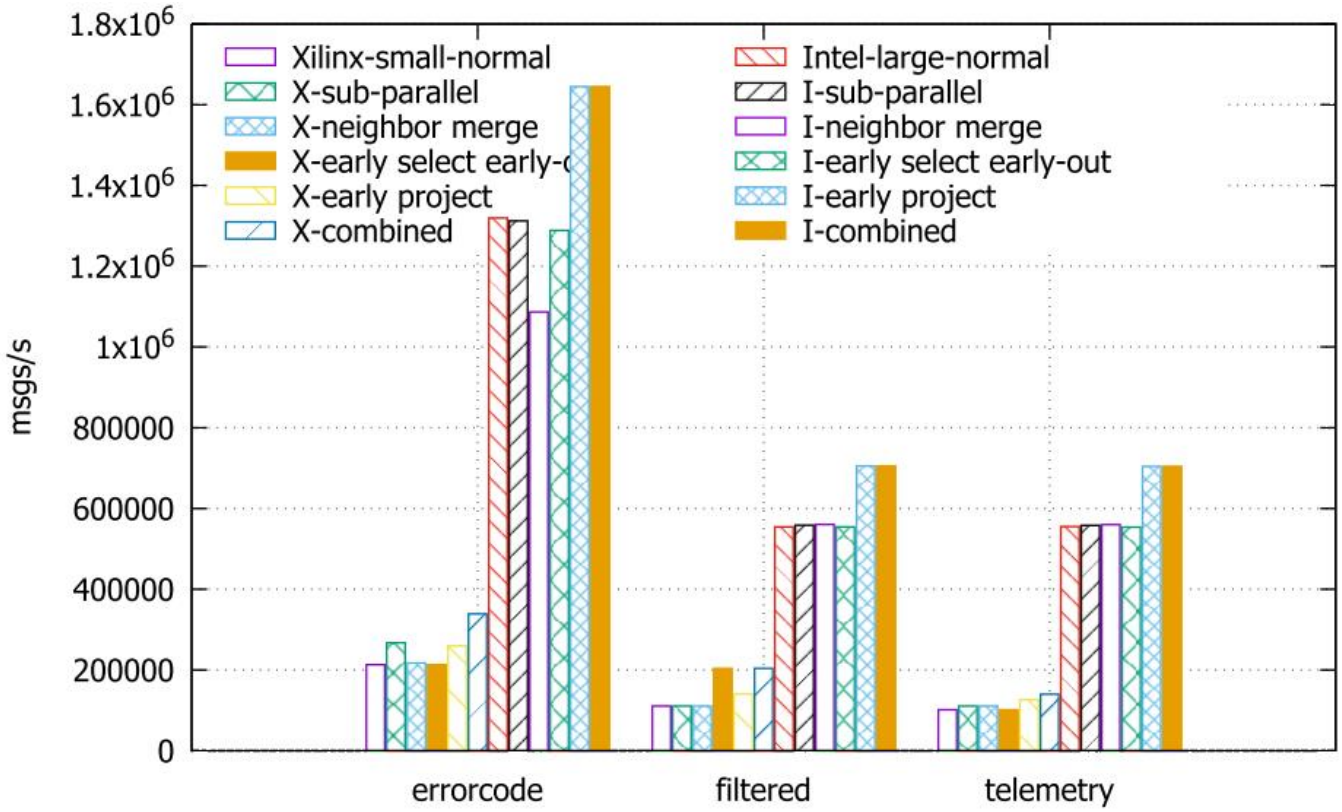
Connected Car Scenario: Message Throughput (single instance)

General: not all optimizations in software processes are applicable to hardware processes (not shown). Especially data flow optimizations that reduce the message sizes increase the throughput



Optimization	Result (relative to naive)
DC-6 (Sub-parallel)	+ (error codes), 0 (telemetry data), and more space
DD-4 (early select) + early out	+ (less data processed)
DD-5 (early project)	+ (content filter; reduces data)
DD-12 (Neighbor (operation) merge)	0 (not more instances, similar throughput)

Connected Car Scenario: Message Throughput (single instance)



Scenario can be deployed ~twice on Xilinx and ~50+ times to Intel Arria FPGA (à one FPGA Integration System for CCT)

Discussion and Outlook

Some Results and Future Work

Contributions

- Simple EIP streaming semantics [Zimmermann2017]
- On circuit represenation of the streaming pattern (basics and more advanced)
- Architecture proposal for the usage of FPGAs as integration system (similar to [Caspi2005,Müller2010] in other domains)
- Some results:
  - throughput physically limited for on-chip streaming approach, but scales with more product ready FPGA
  - branching and condition evaluation hardly any impact compared to software implementations
  - less energy consumption[1] of FPGA processing: ~153,061.22 msgs/watt vs 11,052.32 msgs/watt on CPU
- Analysis of common optimization techniques from other domains (e.g., [PVLDB2007, ICEIS2008])

Further Evaluations in the Paper:

- Parallelism: Space Management
- Parallelism: Performance
- Instance Parallelization

Next Steps

- Compare with an in-memory message indexing approach
- Study further optimization techniques
- Adapt to more advanced integration semantics: security, error handling [IJCIS2017], monitoring
- User-interaction: programming model and language support

[1]Power consumption: $P \propto U^2 \times f$, with voltage U, frequency f. For the **FPGA**, a power analyzer provided by Xilinx reports an estimated consumption of 1.0 W, and for the **CPU** the consumption lies between the *Extended HALT Power* and the *Thermal Design Power* around 95 Watt.

References

[Hohpe2003] G. Hohpe and B. Woolf. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley Longman Publishing Co., Inc., 2003.

[Caspi2005] E. Caspi. Design Automation for Streaming Systems. PhD thesis, UC Berkley, Berkeley, CA, USA, 2005.

[PVLDB2007] M. Vrhovnik, H. Schwarz, O. Suhre, B. Mitschang, V. Markl, A. Maier, and T. Kraft. An approach to optimize data processing in business processes. In PVLDB, pages 615–626, 2007.

[ICEIS2008] M. Böhm, U. Wloka, D. Habich, and W. Lehner. Model-driven generation and optimization of complex integration processes. In ICEIS (1), pages 131–136, 2008.

[Agrawal2008] J. Agrawal, Y. Diao, D. Gyllstrom, and N. Immerman. Efficient pattern matching over event streams. In ACM SIGMOD, pages 147–160, 2008.

[Müller2009a] R. Müller, J. Teubner, and G. Alonso. Data processing on FPGAs. PVLDB, 2(1):910–921, 2009.

[Müller2009b] R. Müller, J. Teubner, and G. Alonso. Streams on wires - A query compiler for FPGAs. PVLDB, 2(1):229–240, 2009.

References cont‘d

[Woods2010] L. Woods, J. Teubner, and G. Alonso. Complex event detection at wire speed with FPGAs. PVLDB, 3(1):660–669, 2010.

[Müller2010] R. Müller and J. Teubner. FPGAs: a new point in the database design space. In EDBT, pages 721–723, 2010.

[ElHassan2010] Fadi El-Hassan, Dan Ionescu: A Hardware Architecture of an XML/XPath Broker for Content-Based Publish/Subscribe Systems. ReConFig 2010: 138-143

[IJCIS2017] D. Ritter and J. Sosulski. Exception handling in message-based integration systems and modeling using bpmn. International Journal of Cooperative Information Systems, 0(0):1650004, 0.

[Zimmermann2016] O. Zimmermann, C. Pautasso, G. Hohpe, and B. Woolf. A decade of enterprise integration patterns: A conversation with the authors. IEEE Software, 33(1):13–19, 2016.

[DEBS2017a] M. Grossniklaus, D. Maier, J. Miller, S. Moorthy, and K. Tufte. Frames: data-driven windows. In ACM DEBS, pages 13–24, 2017.

[DEBS2017b] D. Ritter, N. May, K. Sachs, and S. Rinderle-Ma. Benchmarking integration pattern implementations. In ACM DEBS, pages 125–136, 2017.

Thank you

Contact information:

Daniel Ritter
daniel.ritter@sap.com