Microelectronic System Design Research Group
University Kaiserslautern
www.eit.uni-kl.de/wehn

SoC-Network for Interleaving in Wireless Communications

Norbert Wehn
wehn@eit.uni-kl.de

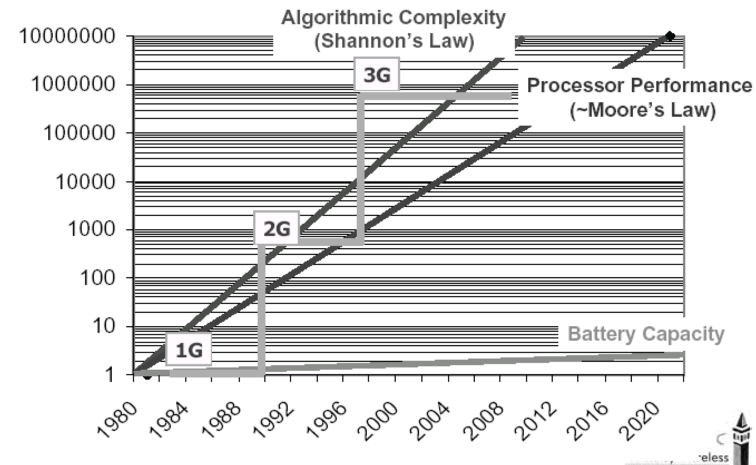
MPSoC'03
7-11 July 2003, Chamonix, France

Outline

- Motivation
- Outer Modem Algorithms
  - Channel Coding
  - Interleaving (Turbo-Codes)
- Application Specific Processing Node
- Application Specific Communication Network
  - Network Structure
  - Network Analysis
- Results
- Conclusion

Wireless Implementation Challenges I

- DECT 10 MIPS, GSM 100 MIPS, UMTS x 1000 MIPS



Courtesy: Ravi Subramanian (Morphics)

MPSoC'03
N. Wehn

Wireless Implementation Challenges II

- Algorithmic Complexity
  - ⇨ "Shannon's Law beats Moore's Law"
- Programmability and Flexibility
  - ⇨ different QoS
  - ⇨ „multi-mode" support: different algorithms & standards
  - ⇨ „software radio"
  - ⇨ different throughput requirements
- Low Power/Low Energy
  - ⇨ BUT: „Energy-Flexibility Gap"
- Design Space
  - ⇨ algorithms, architecture

….

Motivation

MPSoC'03
N. Wehn

New architectures: AP-MPSoC

- *scalable, highly parallel, programmable, energy-efficient*
- *application-specific processor node running with low frequency*
- *application-specific communication network*

Wireless baseband algorithms

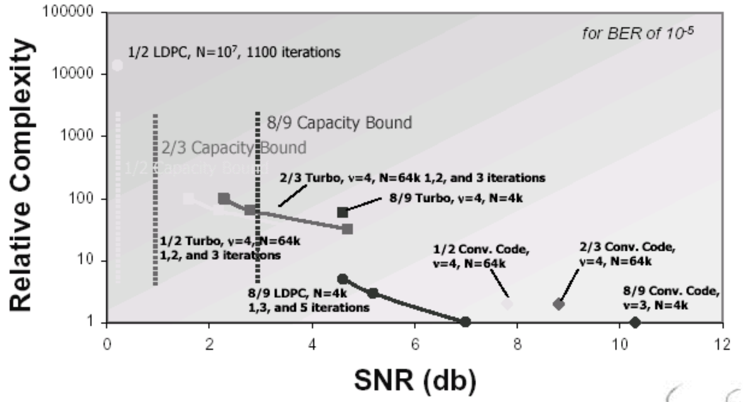
- Inner modem
  - signal processing based on matrix computations e.g. multi-user detection, interference cancellation, filtering, correlators
  - many publications on efficient multi-processor implementations of matrix computations e.g. systolic arrays
- *Outer Modem*
  - Channel coding, Interleaving, Data stream segmentation
  - efficient multi-processor implementation largely unexplored

5

Importance of Channel Coding

MPSoC'03
N. Wehn

Efficient channel coding is key for reliable communication



Courtesy Engling Yeo, UCB

Berkeley Wireless Research Center

High throughput: complexity is in data distribution and not in computation

6

Channel Coding Techniques

MPSoC'03
N. Wehn

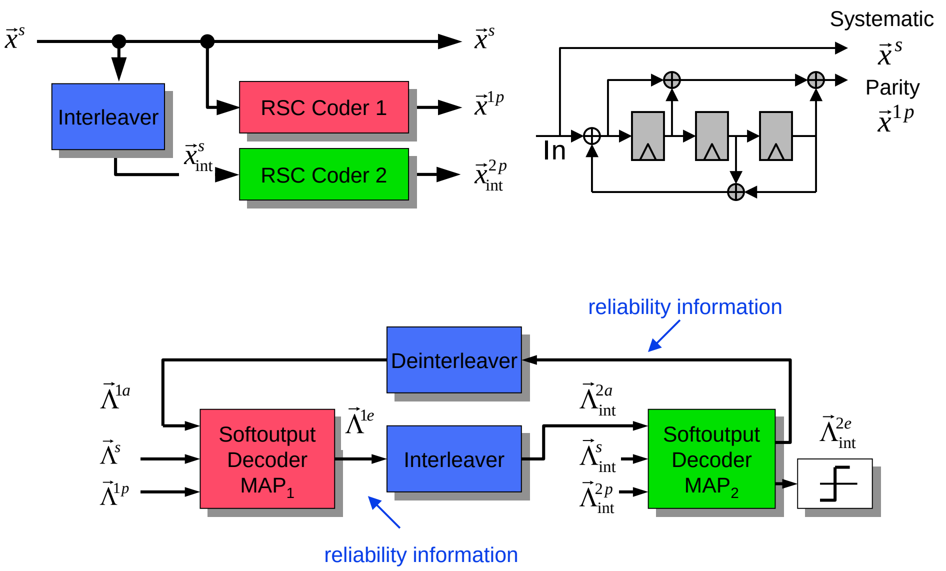
- Convolutional Codes
  - ⇨ Viterbi decoding algorithm
  - ⇨ intensively studied (HW/SW/DSP_extensions)
- Most efficient Codes: Turbo-Codes (1993), LDPC-Codes (1996)
  - ⇨ block-based
  - ⇨ iterative decoding techniques
  - ⇨ computational complexity increased by order of magnitude
  - ⇨ memory access and data transfers are very critical
- Turbo-Codes
  - ⇨ one of the big changes when moving from 2G to 3G
  - ⇨ part of many emerging standards e.g. WLAN, 4G
  - ⇨ Turbo-principle extended to modulation
- Very active research area in the communication community

Mapping of this type of algorithms onto programmable architectures largely unexplored

7

Turbo-En/Decoder Structure

MPSoC'03
N. Wehn



8

Turbo-Codes

- Iterative decoding process
  - ⇨ block-based 3GPP: 20-5114 bits, 3GPP2: 378-20730 bits
  - ⇨ DEC1, Interleaving, DEC2, Deinterleaving
  - ⇨ interleaved reliability information is exchanged between decoders
- Softoutput Decoder
  - ⇨ determine Log-Likelihood Ratio (LLR) of each bit being sent „0" or „1" (Viterbi determines only *most likely path* in trellis)
  - ⇨ three step algorithm: forward/backward recursion, LLR calculation
  - ⇨ ~2.5 x computational complexity of Viterbi algorithm
  - ⇨ memory complexity (size,access) >> Viterbi algorithm
- Interleaving/Deinterleaving
  - ⇨ important step on the physical layer
  - ⇨ scrambles data processing order to yield timing diversity
  - ⇨ minimizes burst errors

Implementation Challenges

- Programmability and Flexibility

  „...It is critical for next generation programmable DSP to adress the requirements of algorithms such as Turbo-Codes since these algorithms are essential for improved 2G and 3G wireless communication"

  (I. Verbauwhede „DSP's for wireless communications")

- High throughput requirements
  - ⇨ UMTS: 2 Mbit/s (terminal), >10Mbit/s (basestation)
  - ⇨ emerging standards >100 Mbit/s
- DSP performance (UMTS compliant based on Log-MAP algorithm)

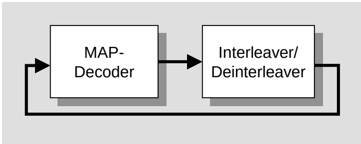
Processor	Architecture	Clock freq. [MHz]	cycles/ (bit*MAP)	Throughput @ 5 Iter.
MOT 56603	16-bit DSP	80	472	17 kbit/s
STM ST120	VLIW, 2 ALU	200	100	~ 200 kbit/s
SC140	VLIW, 4 ALU	300	50	600 kbit/s
ADI TS (1)	VLIW, 2 ALU	180	27	666 kbit/s

(1) With special ACS-instruction support

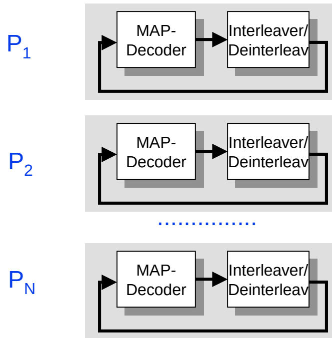
Multiprocessor Solution (Block Level)

Multiprocessor solution becomes mandatory

Single Processor



Simple MP solution



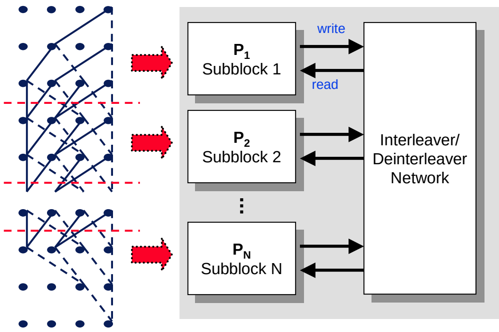
- Sequential processing of
  - MAP algorithm
  - two MAP component decoders
  - Interleaving and Deinterleaving

- N blocks are processed
- **Large latency**
- Low architectural efficiency
  - large area (memory!)
  - high energy

Optimized MPSoC (Sub-Block Level)

Better solution: *parallelization on algorithmic level (sub-block level)*

- MAP decoder parallelization (exploiting trellis windowing technique)
  - each processor can execute a sub-block of of the complete block independently
  - slight increase in computational complexity due to acquisition phase
  - allows distributed computing
- Iterative exchange of interleaved information yields only limited locality



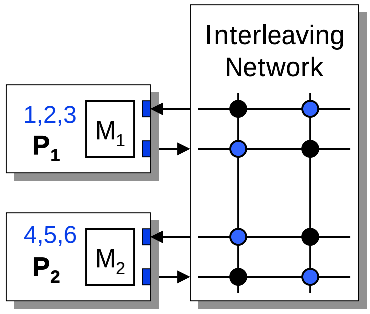
- Low Latency (decreases with N)
- Large architectural efficiency
- Computational locality but
  network-centric architecture

Interleaver Bottleneck

MPSoC'03
N. Wehn

- Data from ***N*** sources have to be „perfectly randomly“ distributed

BIT	P_I	Interl. position	P_I
1	1	**3**	1
2	1	**6**	2
3	1	**5**	2
4	2	**2**	1
5	2	**4**	2
6	2	**1**	1



⇨ Average : P_i sends & receives same amount of values/cycle
⇨ Peak : P_i can receive up to N-1 more values than average value

↷ Crossbar functionality, but with output blocking conflict

13

Interleaving Network Requirements

MPSoC'03
N. Wehn

- Flexibility and Scalability
  - ⇨ Interleaver scheme can change from decoding block to block
  - ⇨ e.g. ~ 5000 different interleaver tables in UMTS
  - ⇨ Different throughput requirements
- Global data distribution
  - ⇨ Good interleavers imply no locality
- 0-latency penalty
  - ⇨ data distribution should be completely done in parallel to data calculation
- Write conflicts i.e. different PEs write simultanously onto same target PE
  - ⇨ multi-port memories infeasable
  - ⇨ conflict-free interleaver design (e.g. IMEC approach), but lack of flexibility

14

Application Specific Processing Node

- Increased ILP by Tensilica Xtensa RISC core for MAP calculation
  - double add-compare-select operation (butterfly)

$$\alpha_k(2n) = \max^*(\alpha_{k-1}(n) + \Lambda in_k(I),\ \alpha_{k-1}(n+M/2) + \Lambda in_k(II))$$
$$\alpha_k(2n+1) = \max^*(\alpha_{k-1}(n) + \Lambda in_k(II),\ \alpha_{k-1}(n+M/2) + \Lambda in_k(I))$$

  - max* operation

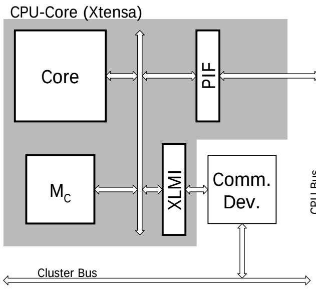
$$max^*(x_1, x_2) = max(x_1, x_2) + ln(1+exp(-|x_2-x_1|))$$

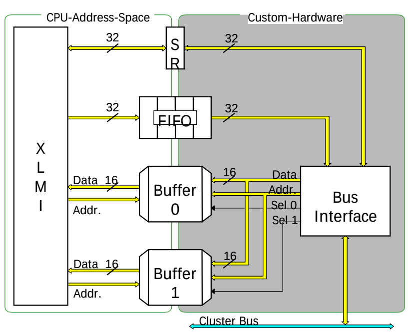
  - zero overhead data-transfers: memory operations parallel to butterfly operation
- 1.54mm^2 (0.18um techology), f=133 MHz

Processor	Clock freq. [MHz]	cycles/ (bit*MAP)	Throughput @ 5 Iter.
Xtensa	133	9	1,4 Mbit/s
STM ST120	200	100	~ 200 kbit/s
SC140	300	50	600 kbit/s
ADI TS	180	27	666 kbit/s

Processing Node Interface

- Fast single-cycle local data memory M_C
  - mapped into processors adress space
- XLMI single-cycle data interface for interprocessor communication
- Communication device for data distribution
  - message passing network (message=data + target addr.)
  - single cycle access



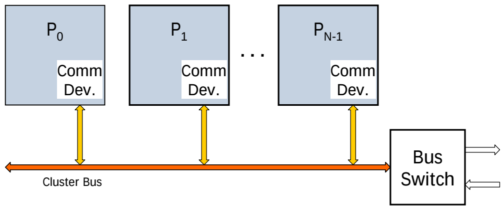


Message format

0	Node ID target Processor (7bit)	Local address in buffer (14bit)	Buffer ID (1bit)	0	Data (8bit)

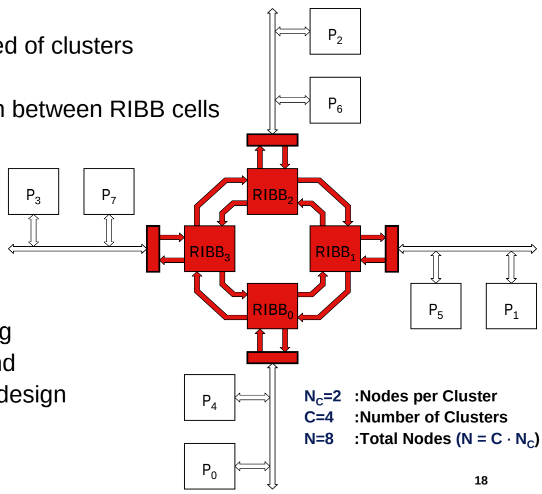
Network Structure

- **K** number of bits in a decoding block (e.g. 5114)
- **N** number of processing nodes
  - ⇨ each node processes **K/N** bits
- **R** average number of cycles per calculated data on a node processor
- Complete block processing needs **R*K/N** cycles
- Throughput requirement on communication network **N / R**
- **N / R ≤ 1** simple bus architecture sufficient



Heterogeneous Network

- Bus: limited scalability and throughput e.g. UMTS conditions
  - ⇨ N_{max}=5
  - ⇨ max throughput ~ 7 Mbit/s
- Hierarchical network composed of clusters
  - ⇨ ring topology
  - ⇨ point-to-point connection between RIBB cells
- RIBB cell
  - ⇨ crossbar switch
- Maximized locality
  - ⇨ minimized global routing
  - ⇨ only neighbouring routing
  - ⇨ scalable to a large extend
  - ⇨ allows synthesis-based design methodology
  - ⇨ does not limit t_{cycle}



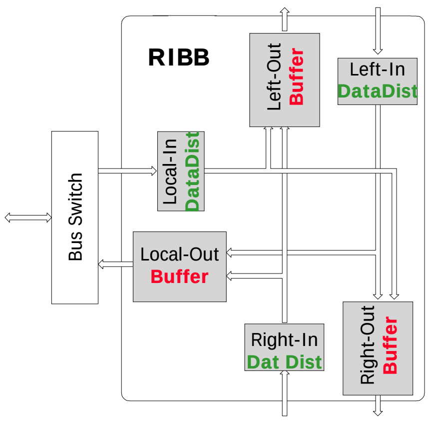
RIBB Cell

Data distributor
- routing decision unit
- determines target buffer
- nearest neighbour routing

Buffer (FIFO)
- multiple data in
- single data out
- buffer sizes determined by simulation at design time

Throughput
- 1 message / cycle per Link

Low complexity cell



Network Analysis

Necessary and sufficient conditions such that the throughput of the communication network does not degrade the AP-MPSoC throughput i.e. data distribution is completely done in parallel to computation

K	:	Interleaver size
N_C	:	Nodes per Cluster
R	:	Data production rate
C	:	Number of Clusters
N	:	Total Nodes

Perfect interleaver: P_{node_acess} = **1/N**

Internal Cluster traffic $N_C * \frac{1}{C} * \frac{K}{N} = \frac{1}{C^2} * K$

Traffic from/to cluster $N_C * \frac{C-1}{C} * \frac{K}{N} = \frac{C-1}{C^2} * K$

Cluster traffic must be completed within data calulation

$$\frac{1}{C^2} * K + 2 * \frac{C-1}{C^2} * K \leq R * \frac{K}{N}$$

Network Analysis

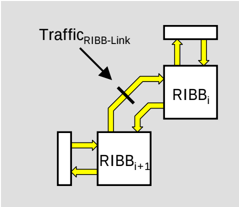
MPSoC'03
N. Wehn

- Traffic on the cluster bus determines number of nodes per cluster

$$N_C \le R \cdot \frac{C}{2 \cdot C - 1} \Rightarrow N_C \approx \frac{1}{2} R$$

- Scheduling Scheme:
  - Grant_{nodes} = C/(2C-1)
  - $\text{Grant}_{bus\_switch}$ = 1-C/(2C-1)

- Traffic on ring-network („nearest neighbour routing“)



$$\text{Traffic}_{\text{RIBB-Link}} = \sum_{i=0}^{\frac{C}{2}-1} \frac{1}{2} \frac{C-1}{C^2} \cdot K - i \cdot \frac{K}{C^2} = \frac{1}{8} K$$

Traffic must be completed within data calulation

$$\frac{1}{8} * K \le R * \frac{K}{N}$$

Network Analysis

MPSoC'03
N. Wehn

- Traffic on ring network determines total number of nodes

$$N \le 8 * R$$

- *Worst case RIBB capacity limit:* R_{max}=1 → N=8
  - Extended RIBB to chordal ring → N=22
  - Synthesis based results (0,18 um technology), UMTS conditions, average values

N	Buff_{left}	Buff^{*}_{local}	Buff_{right}	Buff_{chord}	RIBB [mm²]
4	4	34	4	-	0.16
6	6	29	7	-	0.14
8	17	19	17	-	0.21
16	17	16	15	4	0.25

*** Buffer has different bitwidth**

Results

- Synthesis-based, 0.18um technology, UMTS compliant (K=5114, 5 iterations), t_{cycle} =7.5ns, R=5, R_{LLR}=9

Total Nodes (N)	# of Clusters (C)	Cluster Nodes (N_C)	Throughp.* [Mbit/s]	Area Comm. [mm^2]	Area Total [mm^2]	Efficiency [$Mb/s*mm^2$]
1	1	1	1.48	NA	6.42	1
5	1	5	7.28	0.21	14.45	2.19
6	2	3	8.72	0.66	16.73	2.26
8	4	2	11.58	1.25	20.91	2.40
12	6	2	17.18	2.02	28.92	2.58
16	8	2	22.64	2.88	36.98	2.66
32	16	2	43.25	7.29	70.26	2.67
40	20	2	52.83	10.05	87.47	2.62

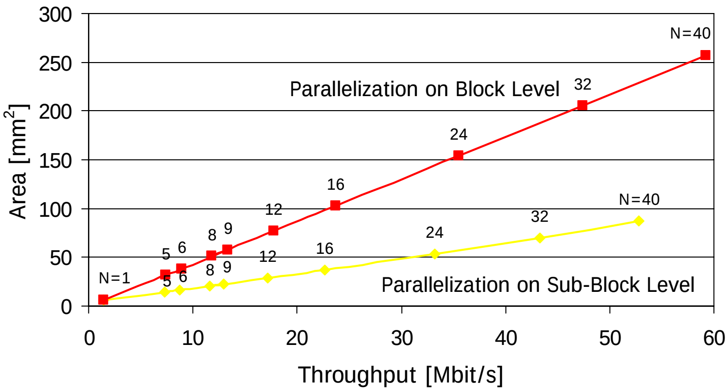
* Validated with Tensilica Xtensa API Interface, Tensilica ISS simulator

- Architecture efficiency increases with increasing parallelism
  - ⇨ memory dominated application
  - ⇨ application memory (interleaver, I/O data memories) size is constant
  - ⇨ communication network overhead < 10%

Results

MPSoC'03
N. Wehn

- Comparison block level versus sub-block level parallelism



- Sub-block level parallelism
  - ⇨ architecture efficiency superior
  - ⇨ **latency much shorter** (decreased ~ N)

Results dedicated Implementation

MPSoC'03
N. Wehn

- VHDL-Model of fully parameterizable scalable Turbo-Decoder
  - Log-MAP / Max-Log-MAP
  - Window- and Acquisition-Length
  - Maximum Blocklength
  - Number of SMAP Units
- Synthesis and Power-Characterization with Synopsys Design Compiler on a 0.18 µm Standard Cell Library
- Validated in UMTS environment
- 166 MHz Log-MAP Implementation with 6 Turbo Iterations

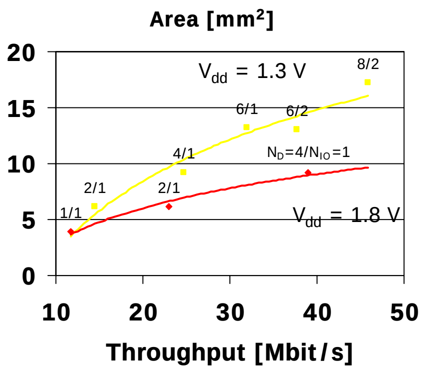
Parallel SMAP Units N_D Parallel I/O N_{IO}	1 1	4 1	6 1	6 2	6 con. I/O	8 1	8 2
Total Area [mm^2]	3.9	9.2	13.3	13.0	18.0	15.9	17.3
Fraction of Memory	85%	69%	69%	68%	77%	61%	64%
Energy per Block [mJ]	48.7	51.7	55.2	50.9	55.2	57.6	55.2
Throughput [MBit/s]	11.7	39.0	50.6	59.6	72.6	59.7	72.7
Efficiency (norm.)	1.00	1.32	1.12	1.47	1.19	1.05	1.24

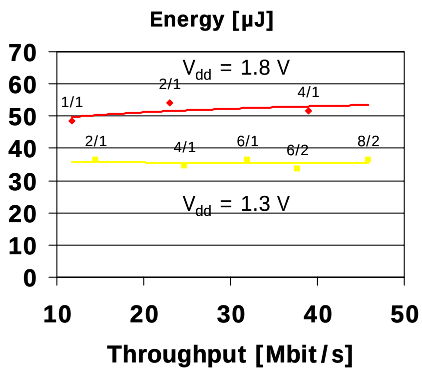
25

Dedicated Solution, VS

MPSoC'03
N. Wehn

- Area, throughput, and energy per decoded block (166 MHz clock frequency, 6 iterations)
- Different degrees of parallelization (N_D and N_{IO}) and different supply voltages (V_{dd})





26

Conclusion

- Channel coding is key for efficient wireless communication
  - Interleaving is a bottleneck for high-throughput iterativ block-based decoding/modulation algorithms
- AP-MPSoC for channel coding
  - parallelization on sub-block level for distributed computing
  - scalable from 1.5 to 52 Mbit/s
  - synthesis-based design methodology
  - application specific processing node
  - increased instruction level parallelism by XTENSA RISC core
- Application specific network for interleaving
  - *network also applicable to LDPC-codes*
  - allows scalable high-throughput architectures (dedicated and programmable) for emerging channel coding techniques
- Low Power
  - Switch –off processing units dependent on throughput
  - (D)VS

Thank you for listening!

For further information please visit

http://www.eit.uni-kl.de/wehn

You can download papers describing the techniques presented in this talk

Special thanks to my PhD students

Frank Gilbert, Gerd Kreiselmaier, Michael Thul, Timo Vogt