A Hardware Packet Re-Sequencer Unit for Network Processors

Michael Meitinger
Rainer Ohlendorf
Thomas Wild
Andreas Herkersdorf

Technische Universität München
Institute for Integrated Systems
Prof. Dr. Andreas Herkersdorf

Arcisstraße 21
80290 Munich, Germany

http://www.lis.ei.tum.de

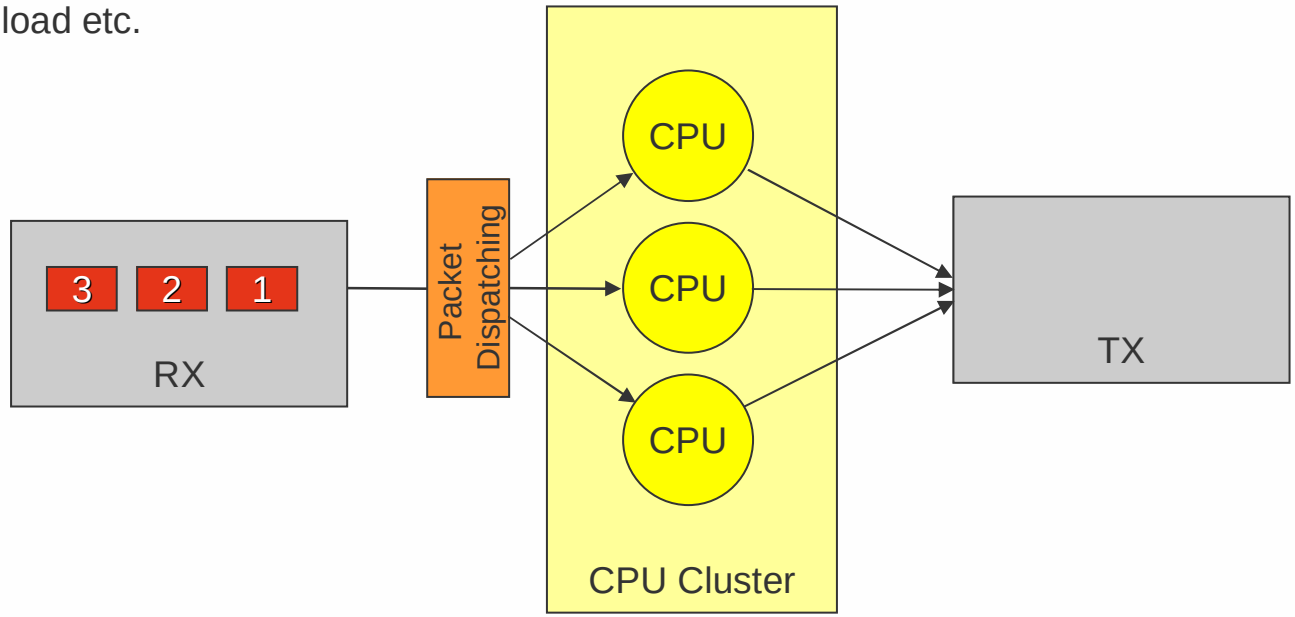
Agenda

- Motivation: Packet Reordering
  - Packet Reordering
  - Effects of Packet Reordering
- Hardware Packet Re-Sequencing Concept
  - Ingress Tagger
  - Egress Aggregation Unit
- SystemC Model
  - Simulation Setup
  - Simulation Results
- FPGA Implementation
- Conclusion

Motivation: Packet Reordering

- Parallel processing of packets
  - Parallel CPUs
  - Multithreading
- Processing latency differs
  - Access time
    - Memory
    - Hardware accelerators etc.
  - Bus load etc.

Packet order per flow not guaranteed
Flow: same IP 5-tuple (IP addresses, ports and protocol type)



Effects of Packet Reordering

- TCP protocol sensitive to Packet Reordering
  - Dominant protocol in internet: ~90%
  - Reordered packets may be defined as lost (if out of receive window)
- Effects
  - Retransmissions of packets
  - Congestion control (high retransmission rate)
- Research:
  - Measurement on IXP2400 10-hop network:
    Retransmissions rate of up to 10% due to packet reordering [1]
  - Significant network performance reduction of up to 60% [2]

Widely used solution:

- Packets of one flow are processed by the same CPU
- **But:** may lead to unbalanced CPU cluster load

► **Puts additional burden on Load Balancing**

[1] Govind, S., Govindarajan, R., Kuri, J.: Packet Reordering in Network Processors. In: IPDPS 2007 (May 2007)
[2] Laor, M., Gendel, L.: The Effect of Packet Reordering in a backbone Link on Application Throughput. IEEE Network (Sept./Oct. 2002)

Hardware Packet Re-Sequencing Concept

Basic Idea:

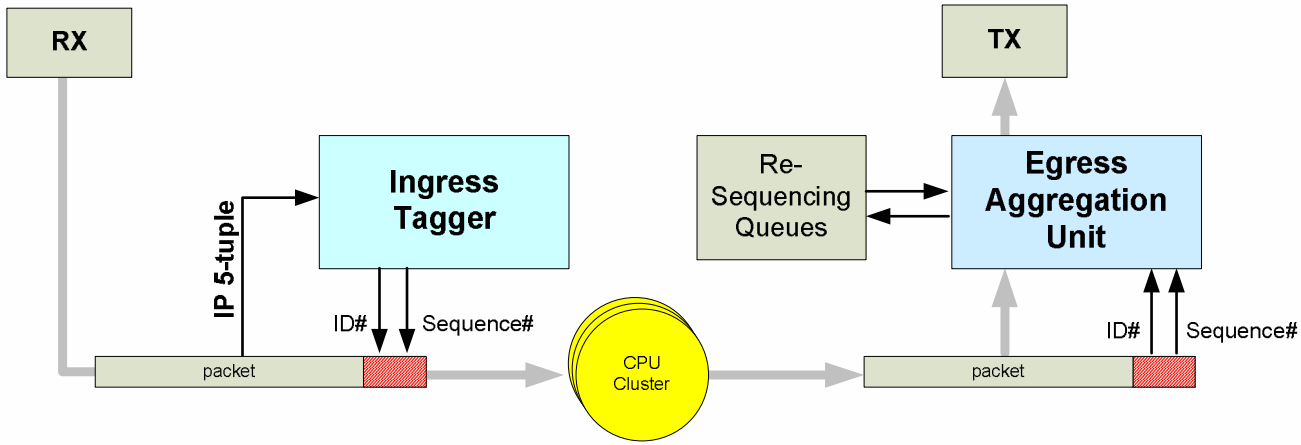
- Allow packet reordering in CPU cluster
- Packets will be re-sequenced after processing
- 2 independent hardware units

Ingress Tagger:

– Stamp each incoming packet with
  - ID indicating flow
  - Sequence number

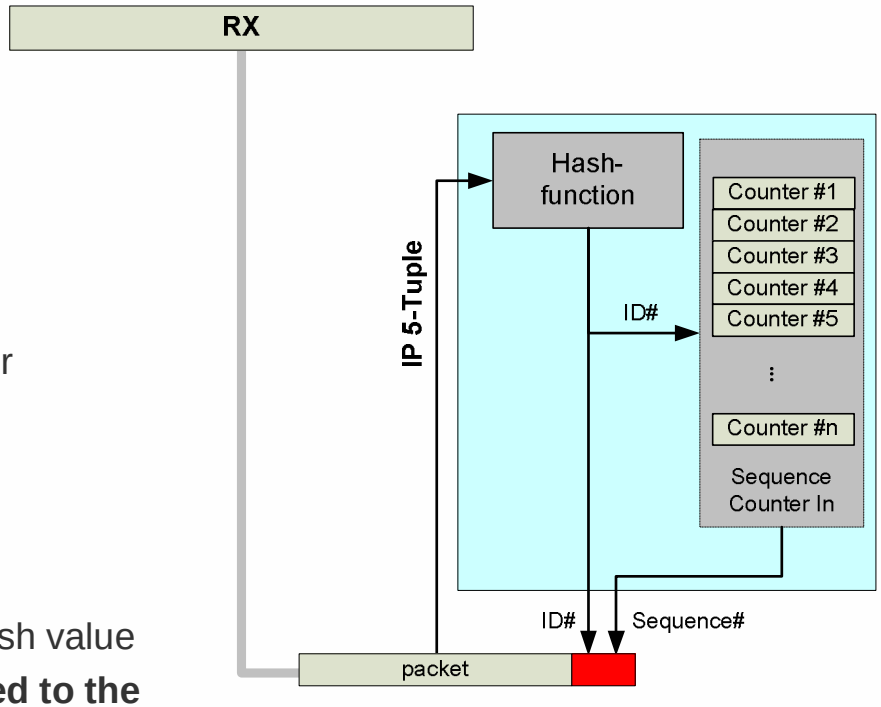
Egress Aggregation Unit:

– check packet order per flow at output
  - In-order-packets are sent out directly
  - Out-of-order packets are stored in buffer



Ingress Tagger

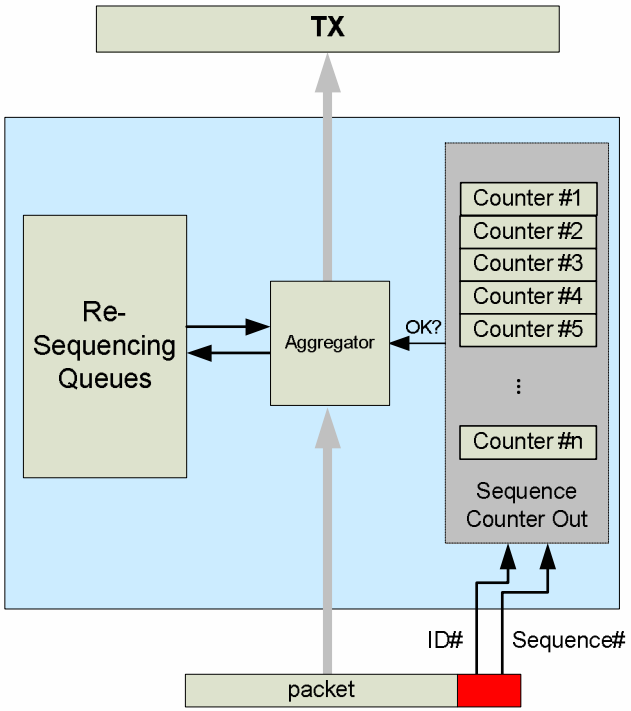
- Flow ID and Sequence number added to packet
  - e.g. as special header
- Flow ID
  - Packet IP 5-tupel will be hashed
  - CRC16: well balanced distribution [3]
  - Full or part of CRC may be taken as ID
- Sequence number
  - Counter per ID is incremented whenever a packet arrives
  - consecutive sequence number
  - May wrap-around
- Hash collision
  - Many flows are mapped to the same hash value
  - **Collision: two active flows are mapped to the same hash value**
  - Effects will be discussed later



[3] Cao, Z., Wang, Z., Zegura, E.: Performance of Hashing-Based Schemes for Internet Load Balancing. IEEE INFOCOM, Tel Aviv, Israel (March 2000)

Egress Aggregation Unit

- Counterpart of Ingress Tagger
- Again including one counter per ID
  - Indicating next expected sequence number
- Aggregator decides based on
  - Flow ID
  - Sequence number
  - Counter value
- Out-of-order-packets are buffered
  - Limited number of re-sequencing queues available
- In-Sequence packets are sent out directly
  - re-sequencing queues are checked for waiting packets
  - Increment counter value by number of sent out packets



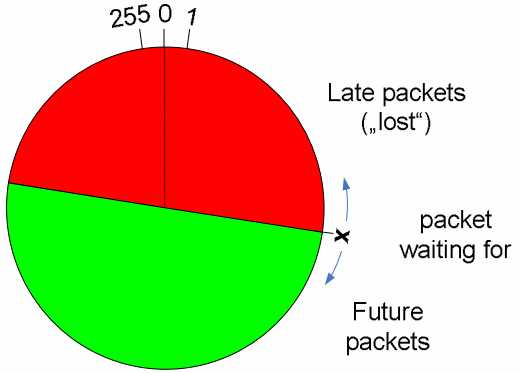
Problem: Re-Sequencing Queues

- In rare cases very large queues needed
  - High latency for single packets
  - Collisions
- Waiting for never arriving packets
  - Packets may be discarded by CPU due to
    - Checksum errors
    - Next-hop Look-up miss etc.

Solution

- Use of timers
  - After a certain time packets waiting for are defined to be lost
  - Packets arriving after timeout must be recognized as late (“lost”) packets
- Maximum queue size defined

► Waiting packets may be sent out without full re-sequencing



Arrangement of sequence numbers (8 bit example)

Problem: Hash collisions

- Hashing leads to collisions:
  several active flows are mapped to the same ID (= hash value)
- Sequence will be kept up for the combination of both flows
  - In most cases: will not have (significant) impact
  - But a mix of
    - High active flow with low processing time
    - Flow with very high processing time (e.g. IPSec, deep packet processing)

    will lead to
    - Unnecessary high latencies
    - High queue fill level
- What is the perfect bit width of hash?

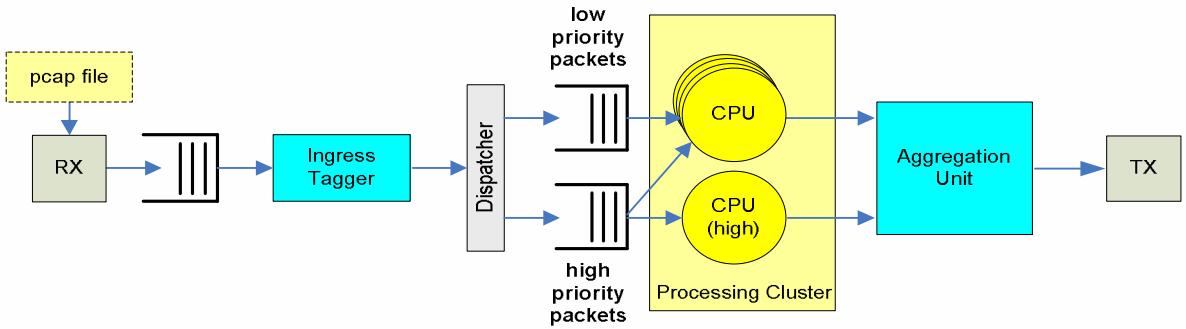
Large bit width: 👍 few collisions
👎 many counters, resource-intensive

Small bit width : 👍 few counters
👎 many collisions

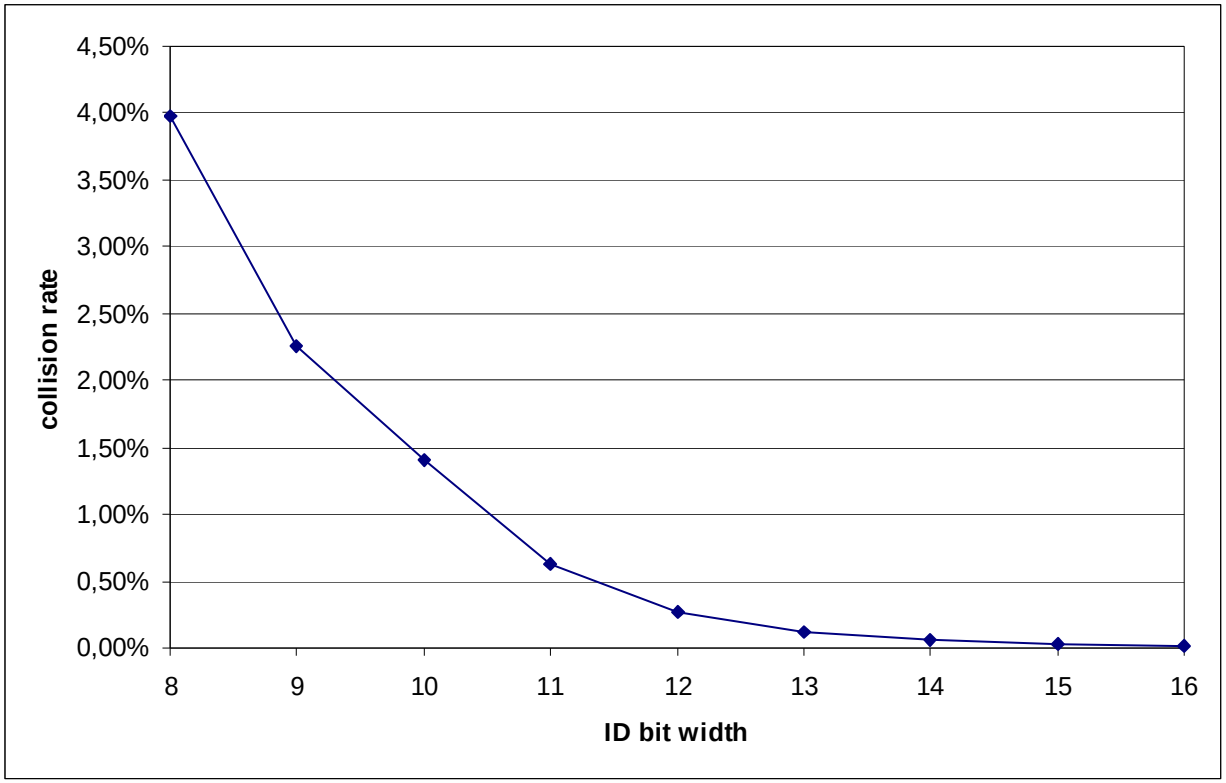
Need to investigate effects of collisions to find the ideal bit width

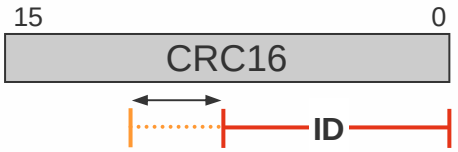
SystemC Model

- Simulation under real traffic conditions
  - Effects of collisions
  - Dimensioning (e.g. queue sizes)
- Packet Source: pcap-file
  - Use of real traffic traces
    - e.g. 100Mbps Backbone
    - Average data rate: 972 Mbps (speedup factor: 70)
- Processor Cluster
  - Variable number of CPUs (here: 8)
  - One dedicated high priority CPU
  - Processing Times and Jitter based on measurements in our FlexPath NP project



Results: Collision rate depending on ID bit width





ID bit width: Truncation of hash value to number of bits

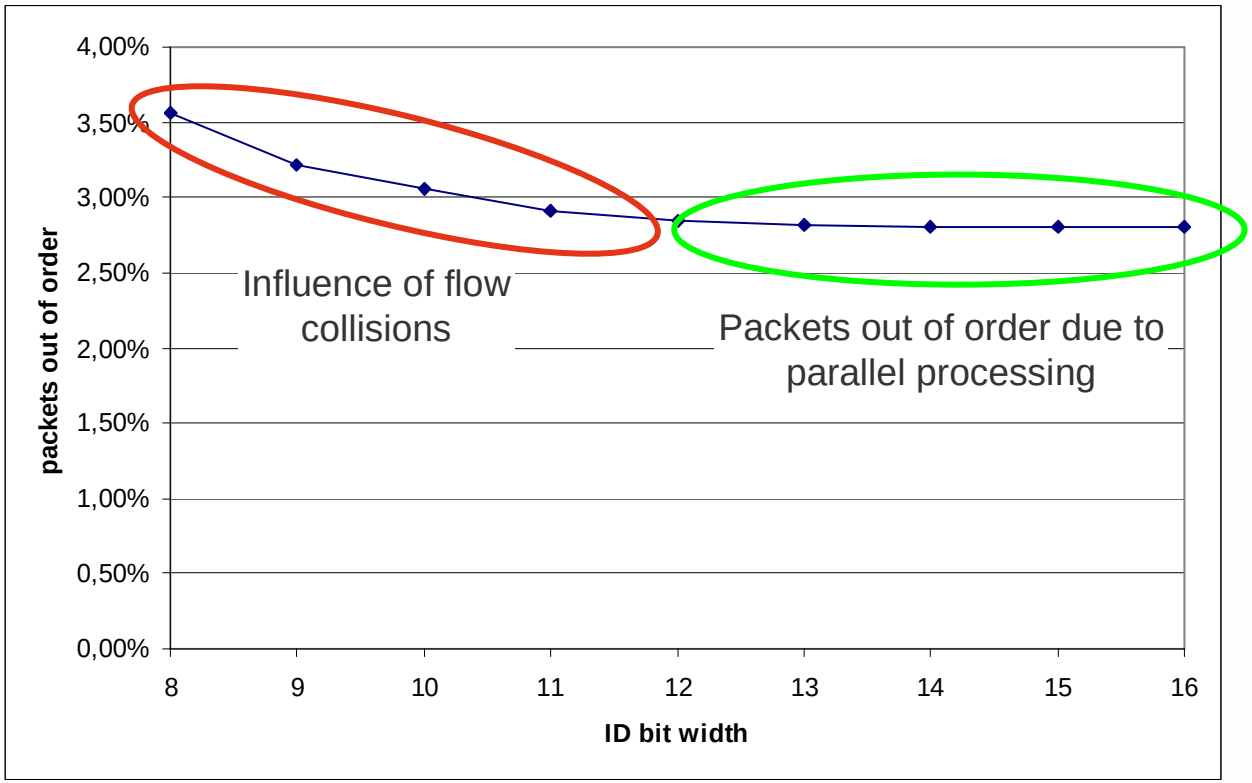
Collision rate: percentage of packets that share a hash value with at least one other active packet (= being currently processed by the system) of another flow

Results: Percentage of re-sequenced packets

- Large bit widths
  - Hardly any collisions (see slide before)
  - Re-sequencing only due to parallel processing
- Small bit widths
  - Influence of collisions can be seen

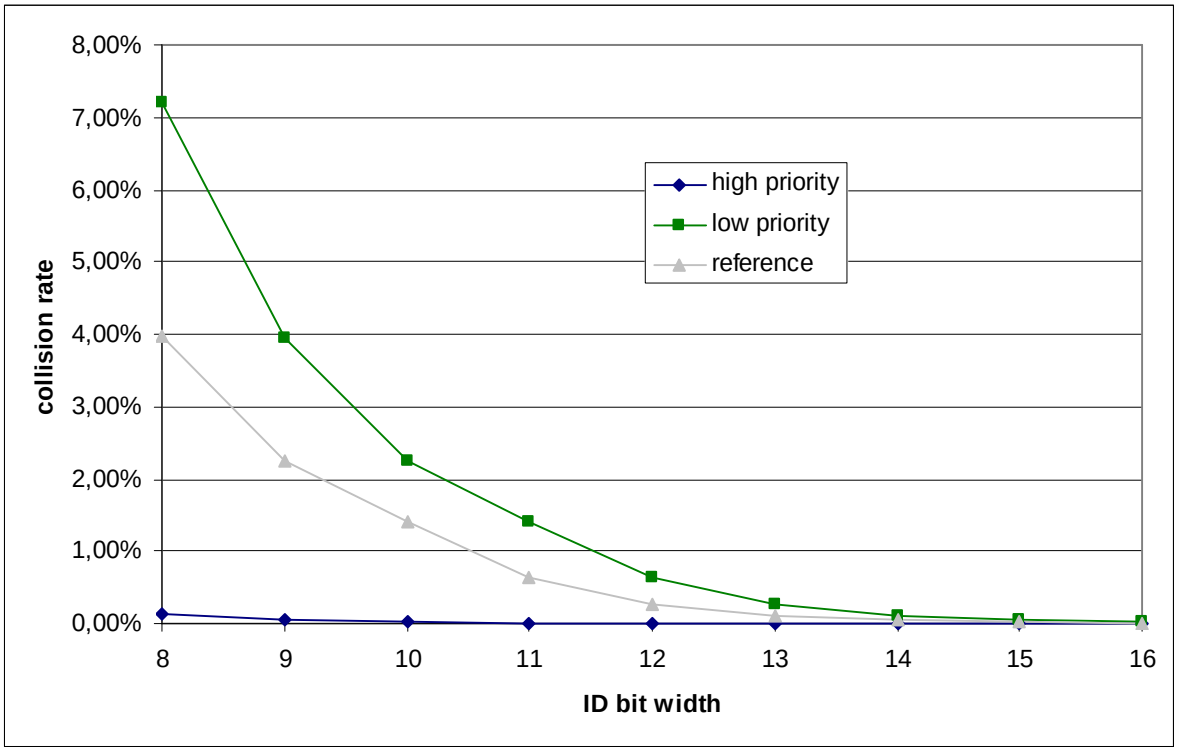
Influence of collisions negligible in practice if ID of 12 bit and more is used!

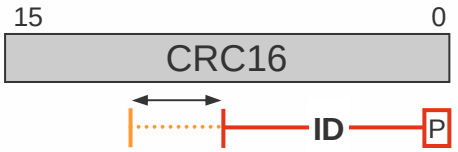
But: what about high priority packets?



Results: Collision rate depending on hash bit width

Using priority bit



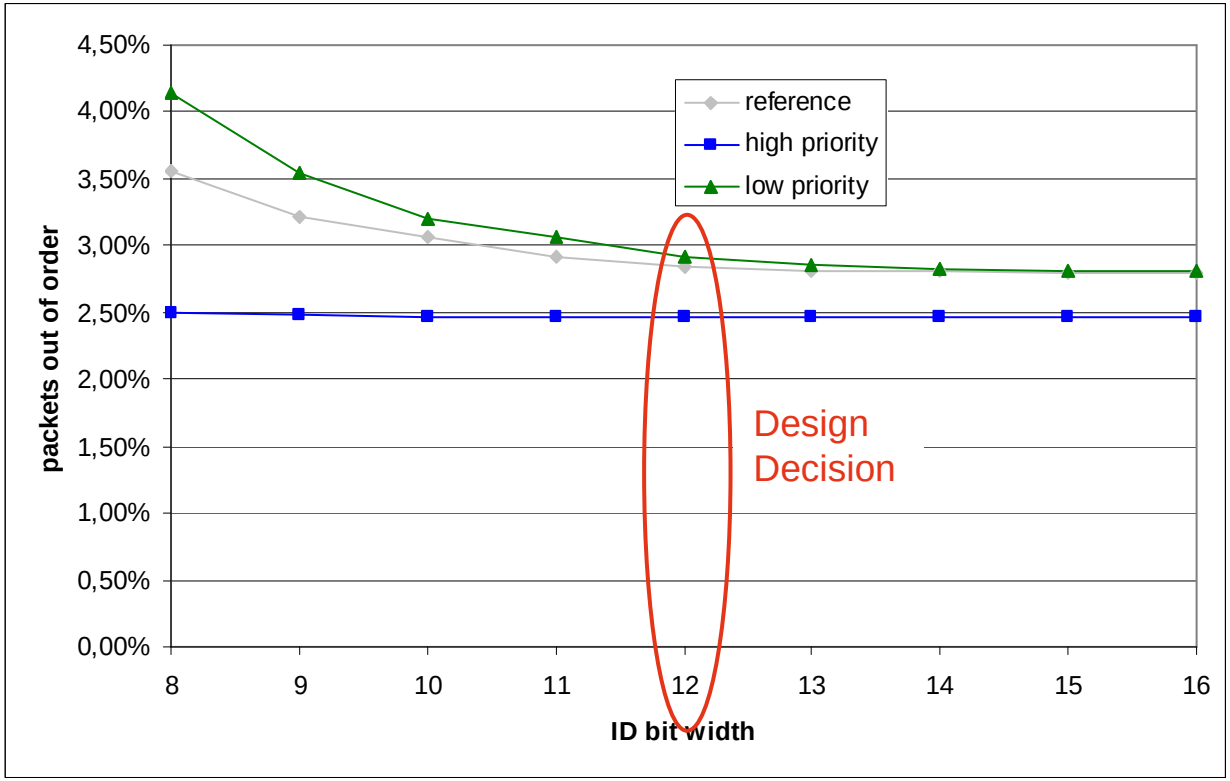


Priority Bit: One bit used to differ between low an high priority packets

Results: Percentage of re-sequenced packets

Using priority bit

- Low Priority packets
  - Influence of collisions increasing due to reduced effective hash size
- High Priority packets
  - Since only a small portion of packets has high priority, probability of collision low
  - Hardly any influence by collisions

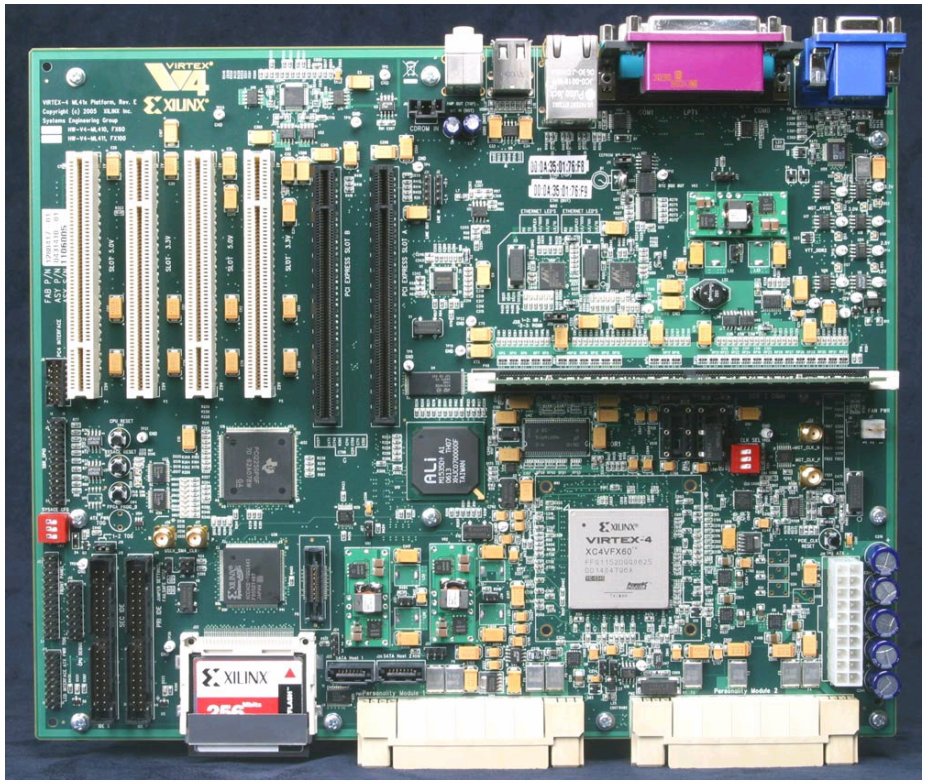


Further Results: ID bit width 12 bit, incl. priority bit

Trace file	1	2	3	4	5
Link	100 mbps	150 mbps	oc-48	oc-3	oc3
Type	Backbone	Backbone	Backbone	Border inbound	Border outbound
Orig. average data rate [Mbps]	13.89	118.07	121.31	19.26	30.54
Packets [million]	3.1	18.1	11.3	17.5	18.0
Orig. trace time [s]	900	900	300	3305	3308
Speedup factor	70	9	8	50	33
Sim. in data rate [Mbps]	972	1062	970	963	1008
Out data rate [Mbps]	906	1047	820	924	1007
Packet loss in PE Cluster[%]	9.3	1.8	14.4	4.6	0.05
coll. rate (low) [%]	0.62	0.45	0.73	0.58	0.42
coll. rate (high) [%]	0.002	0.006	0.038	0.21	0.20
Packets ooo after PE Cl. [%]	2.91	2.64	0.83	1.87	0.97
Max. buffer size (total)	10	12	12	12	9
Max. buffer size per queue	10	12	12	12	9
Max.# of active queues	3	4	3	5	4
Max.# of packets in System	28	25	29	27	27
Max.# of packets of one flow (ID) in System	19	20	18	17	20
Max.# of active flows in System	20	19	20	21	21

FPGA Implementation

- Implemented on Xilinx Virtex-4 FX60 FPGA
  - Ingress Tagger
    - 425 slices
    - 370 4-input LUTs
    - 3 Block SRAMs (18 kbit each)
  - Egress Aggregation Unit
    - 997 slices
    - 1941 4-input LUTs
    - 11 Block SRAMs (18 kbit each)
  - Operating Frequency: 100 MHz
- Validation and Measurements to be done next



Conclusion

- Complete freedom for load balancing
  - Packets of one flow can be distributed among all available CPUs
  - Packet re-sequencing after processing with flow granularity
- Quite simple and efficient solution to keep packet order
  - Full re-sequencing in all simulations
  - Hash collisions negligible in practice
  - Number and size of re-sequencing queues can be held small
    - Only depending on percentage of out-of-order packets
    - Most packets will pass CPU cluster in-sequence
- No 100% guarantee to keep packet order
- Good scaling expected
  - No need for resource-intensive Content Addressable Memory (CAM)
  - Encouraging results also for simulations with up to 64 Processors