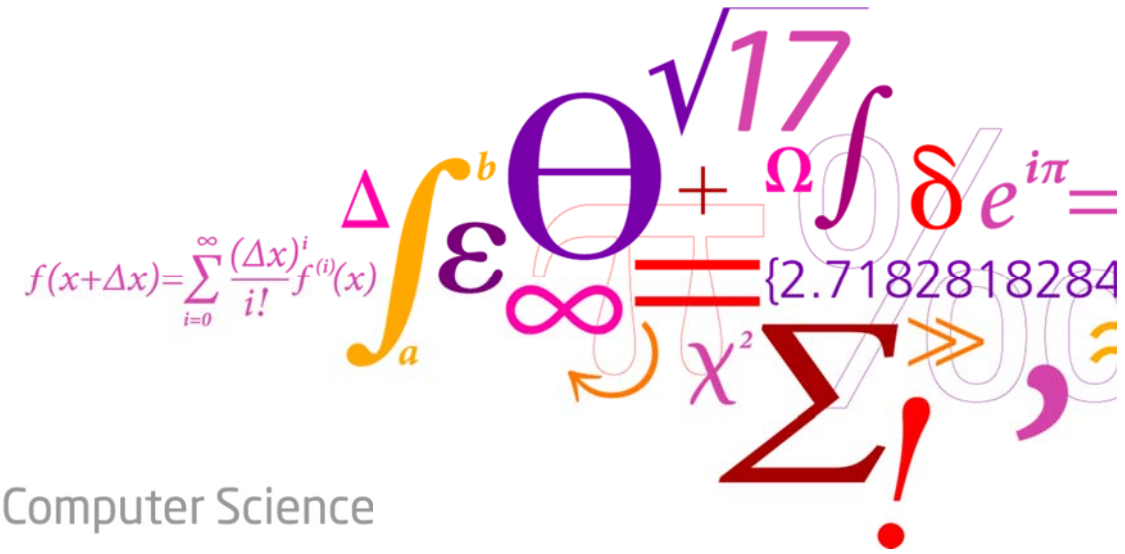
Multicore Message Passing with ARINC 653

Rasmus Bo Sørensen
rboso@dtu.dk



DTU Compute
Department of Applied Mathematics and Computer Science

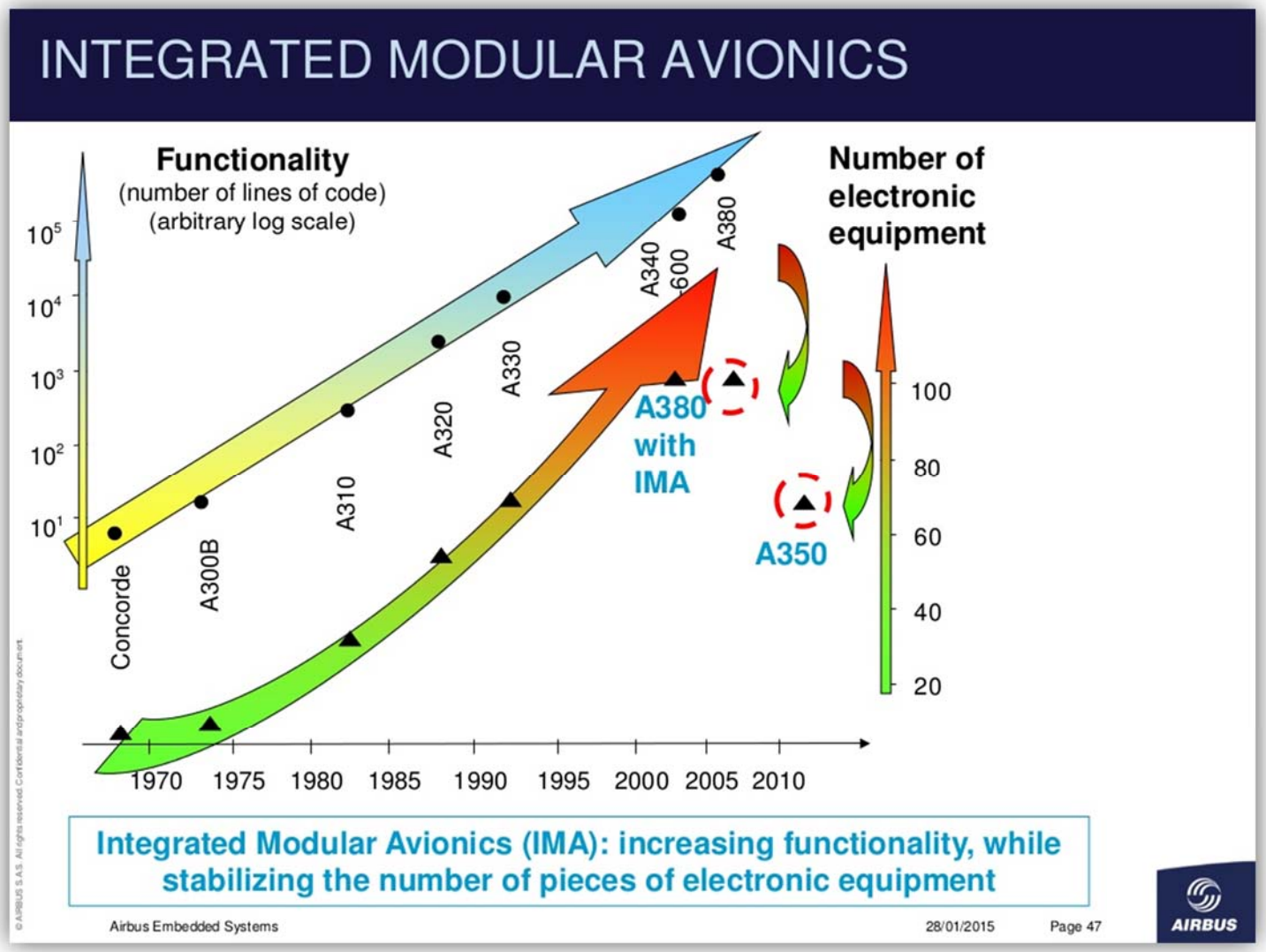
Outline

- Motivation
- Integrated modular avionics
- ARINC 653 communication
- Multicore platform
- Queuing message passing
- WCET analysis
- ARINC 653 queuing ports
- Conclusion
- Future work

Motivation

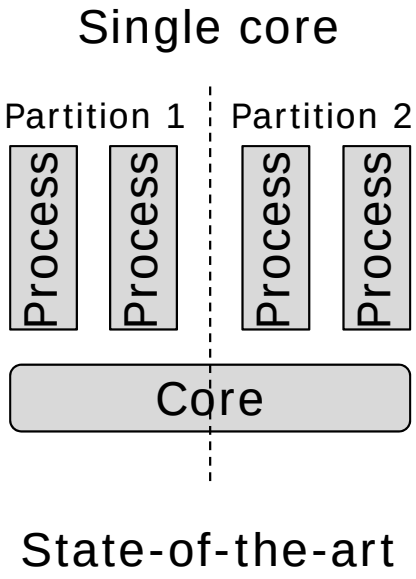
- Trends move towards Integrated Modular Avionics (IMA)
- The future IMA will employ multicore processors
- How does the existing ARINC 653 standard fit with multicore processors?

Integrated Modular Avionics



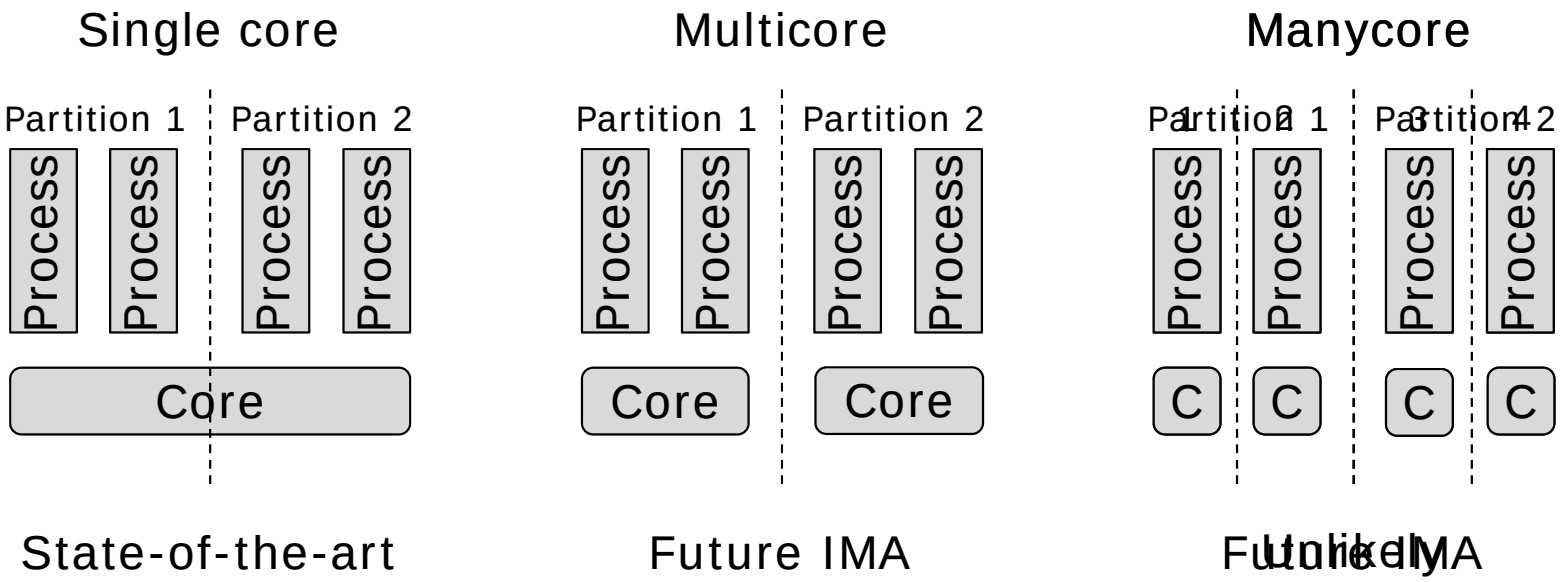
Integrated Modular Avionics

- Saves power and weight, increasing functionality
- Increasing utilization
- Parallel mixed criticality application
- Spatial and temporal separation



Integrated Modular Avionics

- The future will most likely bring partitions on different cores
- The future is unlikely to bring multicore partitions



ARINC 653 Communication

- Explicit message passing between processes/partitions
  - Intra partition
    - Blackboard
    - Buffer
  - Inter partition
    - Sampling port
    - Queuing port

ARINC 653 Communication

Intra partition communication

- Blackboard
  - Always reading the newest value
  - No backwards acknowledge
- Buffer
  - Flow control

ARINC 653 Communication

Inter partition communication

- Sampling port
  - Always reading the newest value
  - No backwards acknowledge
- Queueing port
  - Flow control

ARINC 653 Communication

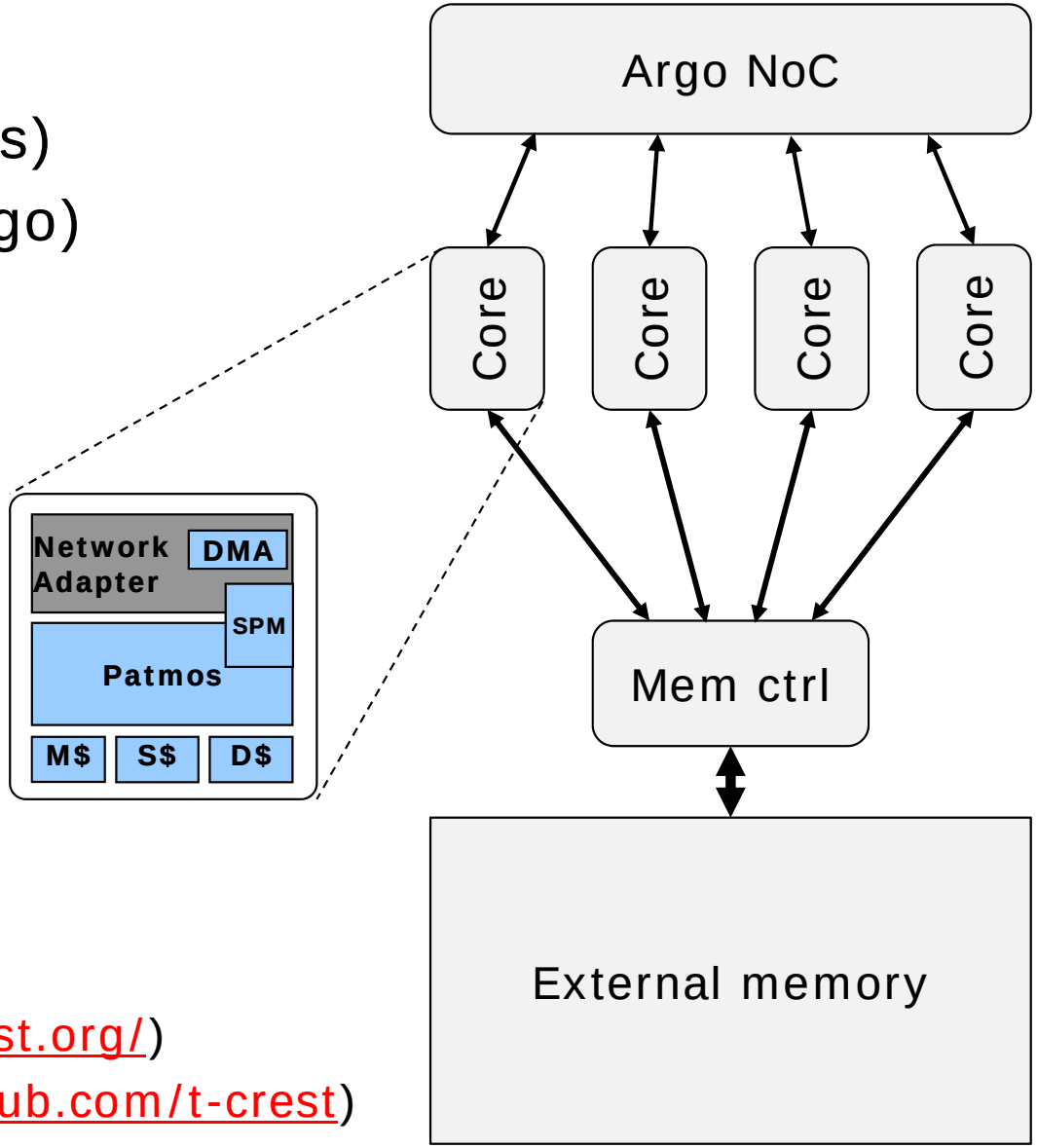
Queuing ports semantics

```
procedure SEND_QUEUING_MESSAGE (
    QUEUING_PORT_ID      : in QUEUING_PORT_ID_TYPE;
    MESSAGE_ADDR         : in MESSAGE_ADDR_TYPE;
    LENGTH               : in MESSAGE_SIZE_TYPE;
    TIME_OUT             : in SYSTEM_TIME_TYPE;
    RETURN_CODE          : out RETURN_CODE_TYPE)
```

```
procedure RECEIVE_QUEUING_MESSAGE (
    QUEUING_PORT_ID      : in QUEUING_PORT_ID_TYPE;
    TIME_OUT             : in SYSTEM_TIME_TYPE;
    MESSAGE_ADDR         : in MESSAGE_ADDR_TYPE;
    LENGTH               : out MESSAGE_SIZE_TYPE;
    RETURN_CODE          : out RETURN_CODE_TYPE)
```

Multicore Platform

- Simple RISC cores (Patmos)
- TDM Network-on-Chip (Argo)
- Open Source platform

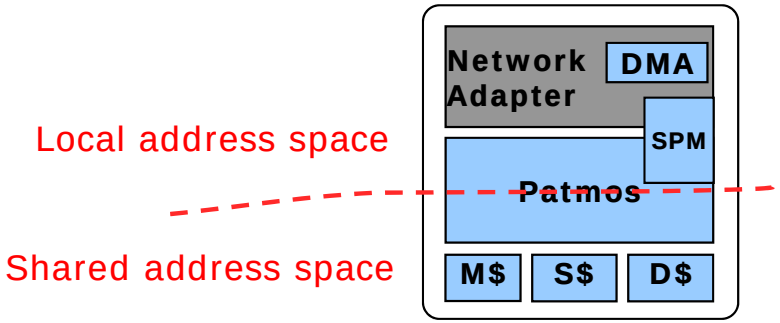


T-CREST project site (http://t-crest.org/)

T-CREST source code (https://github.com/t-crest)

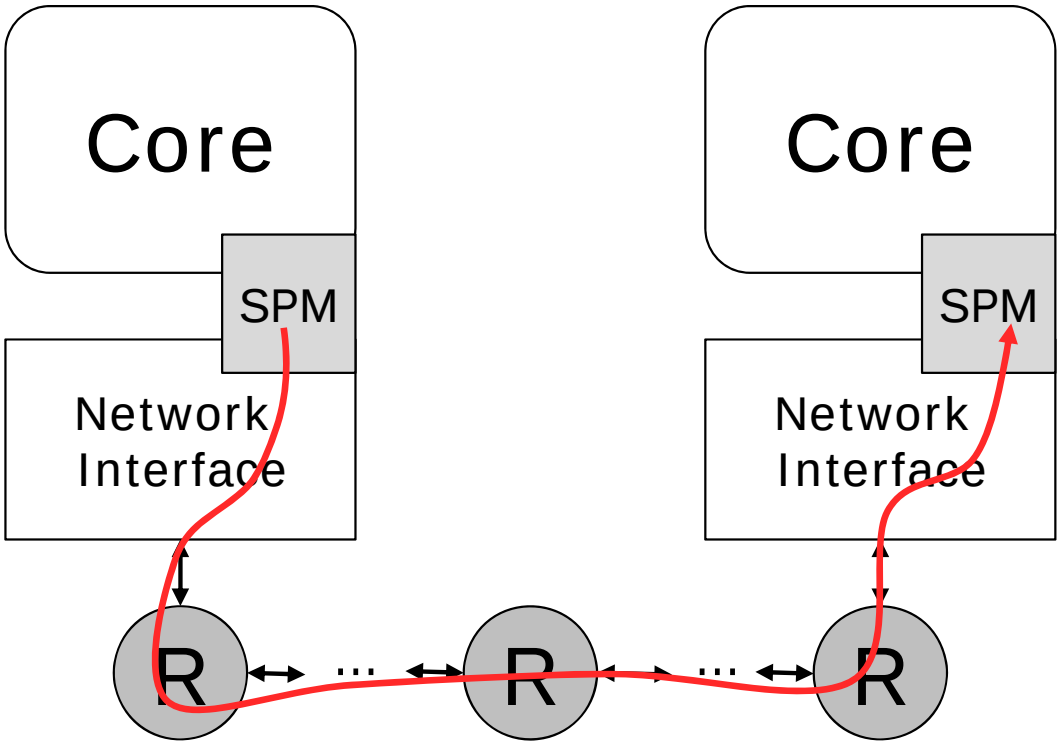
Multicore Platform

- Shared memory and message passing
- Shared address space
  - No hardware cache coherence
  - Uniform access time
- Local address space
  - Private to each core
  - IO devices (UART, timer, Network adapter)
  - Local Scratch Pad Memory (SPM)



Multicore Platform

- Argo architecture
  - Time Division Multiplexing (TDM)
  - Packet switched
  - Statically scheduled
  - Push communication
- No buffering or flow control

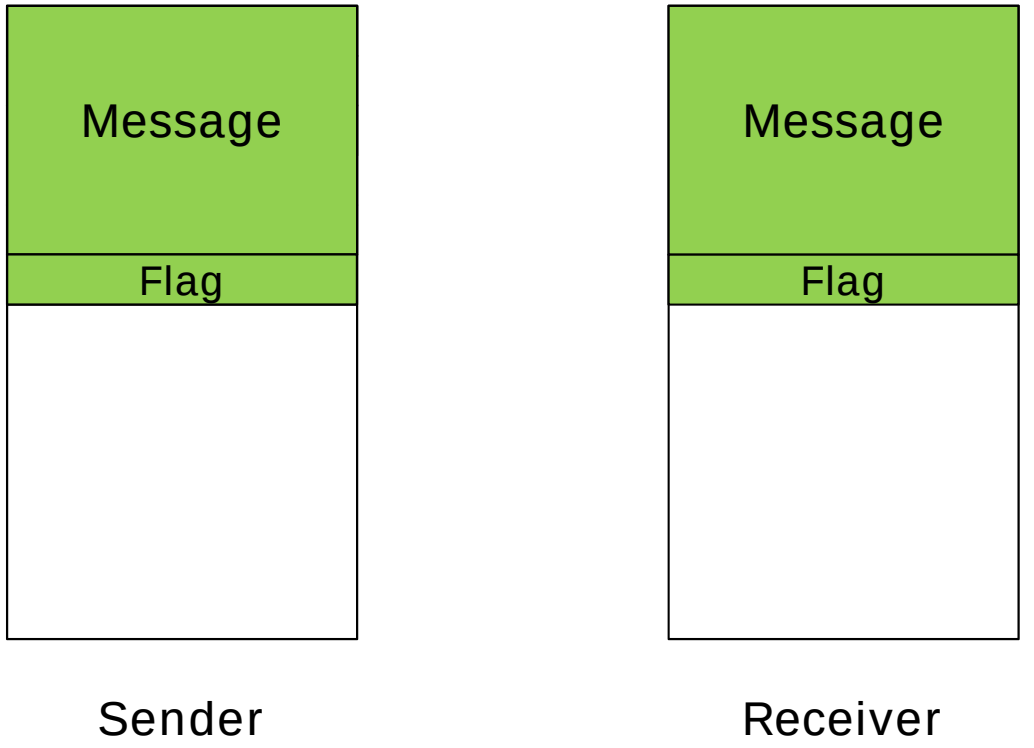


Multicore Platform

- Mapping partitions to cores
- Intra partition communication: core-local memory
- Inter partition communication: core-to-core interconnect

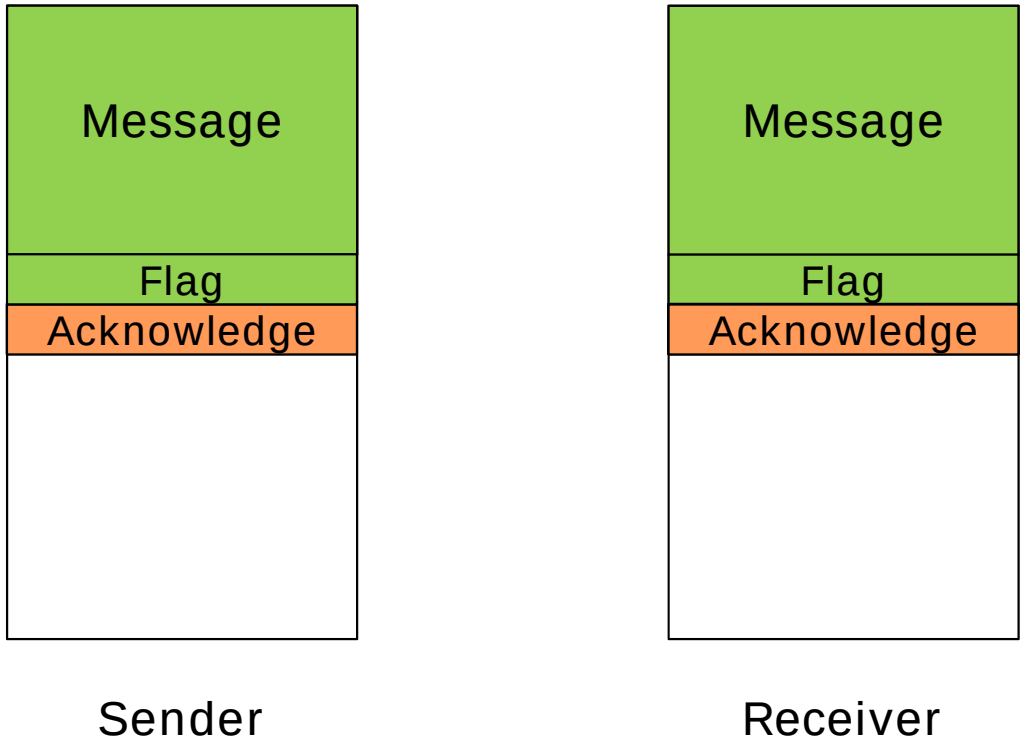
Queuing Message Passing

- Push communication
  - Receive notification at end of packet



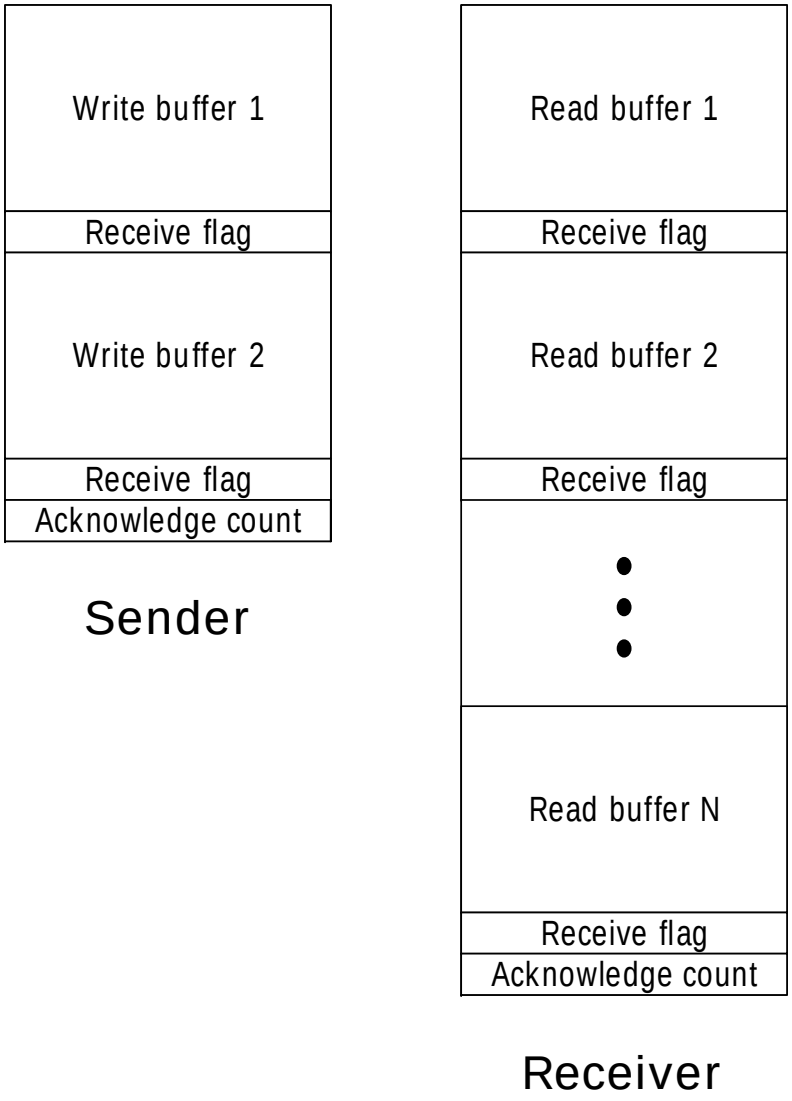
Queuing Message Passing

- Flow control
  - Acknowledge message data



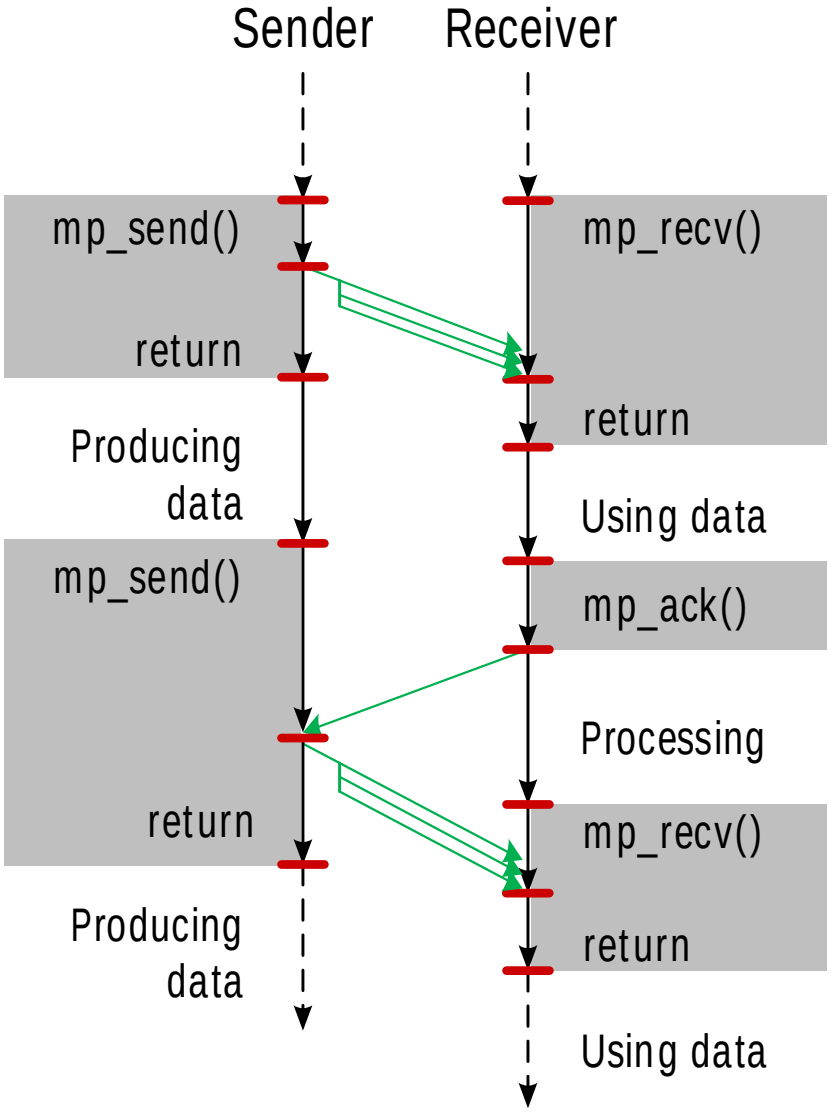
Queuing Message Passing

- Push communication
- Flow control
- Hide latency
  - Double buffering
  - Buffer queue



Queuing Message Passing

- Concurrent threads interact through message passing
- Communication primitives
  - mp_send()
  - mp_recv()
  - mp_ack()



WCET Analysis

Experimental platform (Cyclone IV):

- 9 cores@ 80 MHz
- 2 MB main memory
- 8 KB method cache
- 4 KB communication SPM
- 16 byte cache line size
- Cache line access time 189 cycles

WCET analysis tool

- aiT from AbsInt

WCET Analysis

Communication primitive	ACET (cycles)	WCET (cycles)
mp_send()	74	99
mp_recv()	28	43
mp_ack()	49	77

- Assumption: Code in cache
- Code execution independent on message size

WCET Analysis – End-to-end latency

Message size (bytes)	8	32	128	512	1024	2048
Blocking	211	301	661	2101	4021	7861

- The software contribution to the WCET becomes small as the message size increases
- For 2048 bytes (256 network packets) the software WCET contribution is less than 2 %

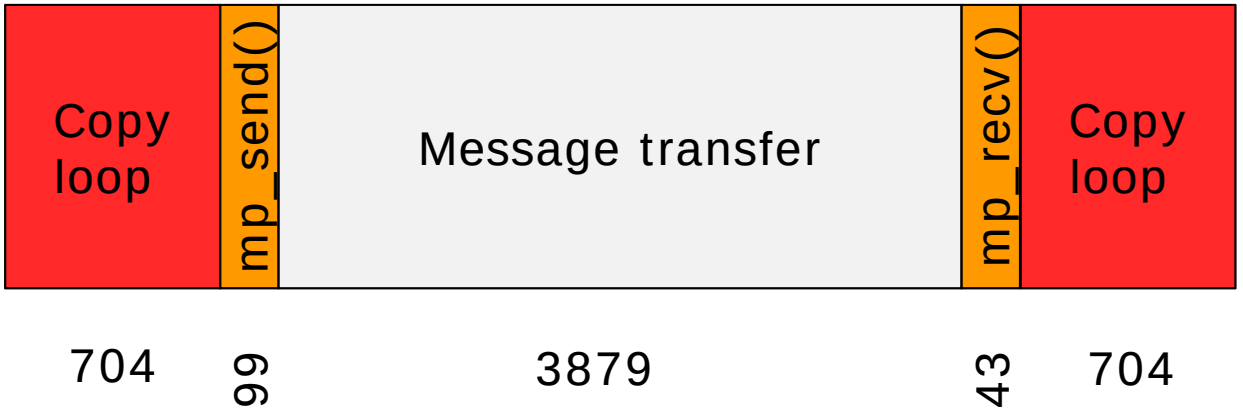
ARINC 653 Queuing ports

- Implementing ARINC 653 on top of our message passing
- We have to copy messages
  - Two additional copy loops
  - Can be unrolled

ARINC 653 Queuing ports

- For 1024 Byte message

4 word copies unrolled



Conclusion

- We can support ARINC 653 on simple hardware
- Our simple interface is faster than ARINC implementation
- Extend the ARINC 653 with non copying primitive for on-chip communication
- Copying messages can be very expensive in terms of latency

Future work

- Sampling ports
- Intra partition
- Applications

ARINC 653 Queuing ports