Efficient Service Allocation in Hardware
Using Credit-Controlled Static-Priority Arbitration

Benny Åkesson
Technische Universiteit Eindhoven
The Netherlands

Liesbeth Steffens
NXP Semiconductors Research
The Netherlands

Kees Goossens
NXP Semiconductors Research &
Delft University of Technology
The Netherlands

Technische Universiteit **Eindhoven** University of Technology

Where innovation starts

Trends in MPSoC Design

- Embedded system design gets increasingly complex
  - Moore's law allows increased component integration
  - Digital convergence creates a market for highly integrated devices

- Systems are implemented as MPSoC platforms with
  - a large number of heterogeneous intellectual property (IP) components
  - many concurrently executing applications with real-time requirements

- Pressure to quickly design systems in a cost-effective manner

Application Requirements

- Applications are mapped on the MPSoC platform
  - Results in communication requirements between IP components
  - IPs wanting access to a resource are referred to as requestors
- In this presentation, we consider hard real-time requestors
  - Example: Audio post processing IP that is a part of an MP3 player
  - Require guaranteed minimum service rate and bounded maximum latency
  - The service requirements may be diverse

MPSoC Constraints

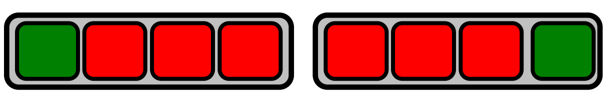
- Resource sharing
  - is required to reduce cost,
  - but introduces interference between requestors,
  - making it difficult to satisfy real-time requirements.

- Access to shared resources provided by a resource arbiter

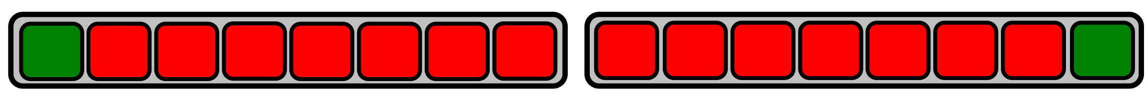
- Resource arbiter requires an implementation that
  - is small.
    - Allows multiple instances to be used in the system with limited impact on area
  - runs at high clock frequency.
    - Enables scheduling on fine granularity, reducing latency and buffers
  - reserves service without over allocating.
    - Prevents wasting scarce resources, such as external memory bandwidth

Related Work

- Existing arbiters fail to satisfy these requirements for three reasons:
  - Allocation granularity coupled to latency
    - Trade-off between over-allocation and low latency
    - Example: frame-based arbiters, such as TDM and Weighted Round-Robin
  - Latency coupled to rate
    - Cannot provide low latency without over allocating
    - Example: Fair queuing family, frame-based arbiters without priorities
  - Cannot run at high clock speed with small implementation
    - Example: Sporadic server (complex accounting)



Frame size = 4
Granularity 1 / 4 = 25%
WC latency = 6



Frame size = 8
Granularity 1 / 8 = 12.5%
WC latency = 14

Credit-Controlled Static-Priority Arbitration

- A Credit-Controlled Static-Priority (CCSP) arbiter has been proposed
  - Comprises a rate regulator and a static-priority scheduler

- Benefits of CCSP:
  - Regulator decouples allocation granularity from latency
  - Static-priority scheduler decouples latency from rate
  - Small hardware implementation that runs at high speed

- Previous work only describes the model behind the rate regulator
  - Assumes infinite precision and is not trivial to implement in hardware

Main Contributions

- In this presentation, we explore
  - how to efficiently represent service allocations in hardware.
  - how over allocation affects provided service.

- Paper also derives implementation of rate regulator and proves correctness
  - Based on proposed service representation
  - Only uses integer arithmetic

Presentation Outline

CCSP Overview

Service Allocation

Experimental Results

Conclusions

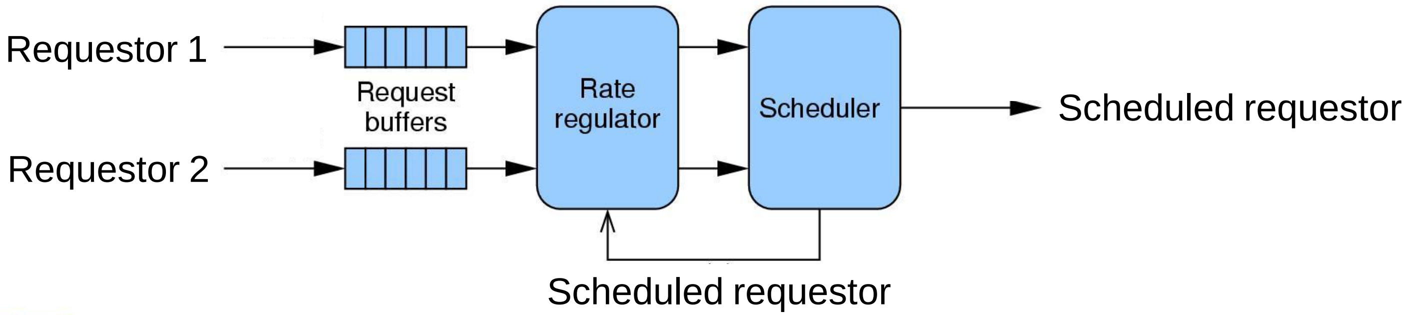
Credit-Controlled Static-Priority Arbitration

- Arbiter consists of a rate regulator and a static-priority scheduler

- Regulator enforces an upper bound on provided service
  - Determines which requestors are eligible for scheduling

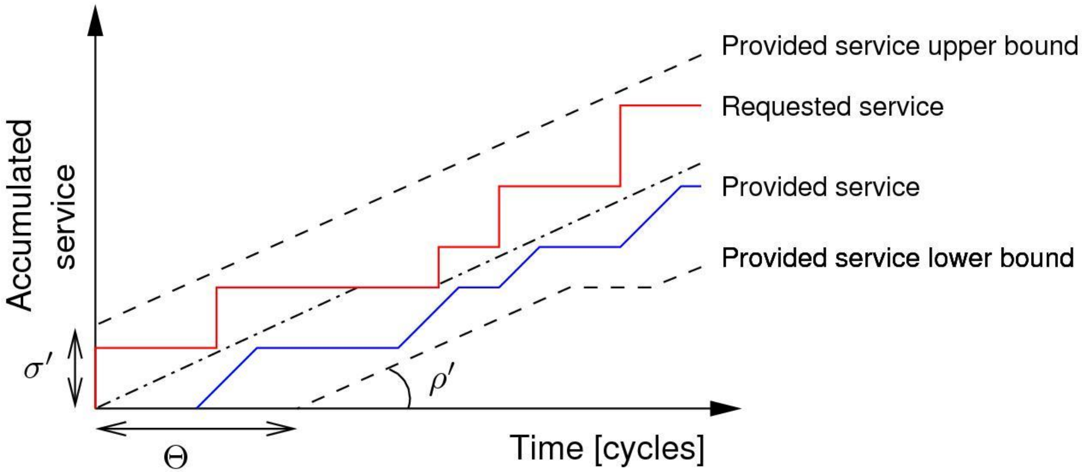
- Static-priority scheduler schedules highest priority eligible requestor

- We consider a preemptive and non-work-conserving instance.



Latency-Rate Server

- In CCSP, service is allocated to a requestor according to an allocated burstiness, σ’, and an allocated service rate, ρ’
- We have shown that CCSP belongs to the class of latency-rate servers
- Allocated service rate, ρ’, guaranteed to a requestor after service latency Θ
  - Lower bound on provided service, bounding the finishing time of a request



$$\Theta = \frac{\sum_{i=0}^{p-1} \sigma_i'}{1 - \sum_{i=0}^{p-1} \rho_i'}$$

Presentation Outline

CCSP Overview

Service Allocation

Experimental Results

Conclusions

Service Representation

- Service allocation in hardware uses finite precision
  - Discretization of intended real-valued allocation
  - Discrete allocation must conservatively approximate the intended allocation

- Discrete allocated rate represented as fraction of integers, $\rho'' = \frac{n}{d}$
  - We refer to the parameters as numerator (n) and denominator (d)
  - Maximum value of n, d are $2^{\beta} - 1$, where β is the accuracy in bits

- As a consequence, discrete allocated burstiness, $\sigma'' = \frac{\lceil \sigma' \cdot d \rceil}{d}$

- Conservative approximations result in over allocation
  - Over allocated rate = ρ'' – ρ'
  - Over allocated burstiness = σ'' – σ'

Allocation Strategies

- There are multiple strategies when selecting the numerator and denominator

- We define a closest burstiness approximation (CBA) strategy
  - Rationale: Minimizing over-allocated burstiness reduces latency
  - Selects largest denominator
  - Selects best numerator to reduce over-allocated rate as second objective

- We also define a closest rate approximation (CRA) strategy
  - Rationale: Minimizing over-allocated rate reduces both waste and latency
  - First selects numerator and denominator that provides closest approximation of ρ’
  - If multiple pairs provide the same approximation, the one with largest denominator is preferred to reduce over allocated burstiness as second objective

$$\Theta = \frac{\sum_{i=0}^{p-1} \sigma_i''}{1 - \sum_{i=0}^{p-1} \rho_i''} \qquad \sigma'' = \frac{\lceil \sigma' \cdot d \rceil}{d} \qquad \rho'' = \frac{n}{d}$$

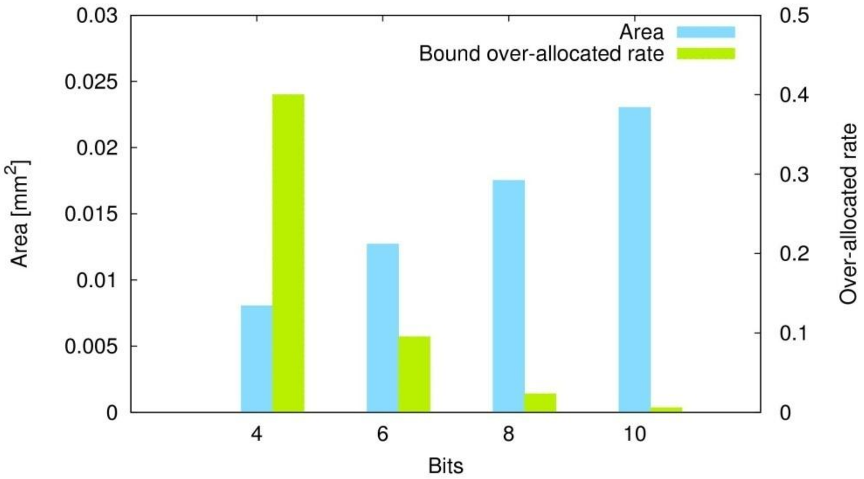
Allocation Properties

- Both strategies provide same bound on over-allocated rate
  - Less than $1 / (2^{\beta} - 1)$
  - However, CRA typically performs much better, since worst-case only happens if allocated service rate is close to zero.
- Bound on over-allocated burstiness of CBA is half of CRA
- Over allocation for both strategies monotonically reduces with increased precision
  - Increasing precision never wastes more capacity nor increases latency
  - Important property for design-space exploration algorithms
  - Property does not hold for frame-based regulators, since latency and rate are coupled

Synthesis Results

- Arbiter implemented in VHDL and synthesized in 90 nm CMOS process
  - Speed target of 200 MHz to fit with a DDR2-400 memory device
  - Instance with 6 ports and 8-bit accuracy requires 0.0223 mm^2

- We experiment by varying the precision of the service allocation
  - Area of the implementation increases linearly with increased precision
  - The bound on over-allocated rate reduces exponentially



Technische Universiteit Eindhoven University of Technology

Presentation Outline

CCSP Overview

Service Allocation

Experimental Results

Conclusions

Experimental Setup

- The context is a predictable MPSoC interconnected with Æthereal NoC
- Arbiter integrated into Predator SDRAM controller
  - Memory device is a 16-bit DDR2-400 @ 200 MHz
  - Guaranteed memory bandwidth is 660 MB/s
  - A request of 64B is served in about 80 ns
- We use synthetic work loads for all experiments
- All service allocations are computed at design time
  - Just exercising tooling
  - No simulation required

Experiment 1 – Allocation Properties

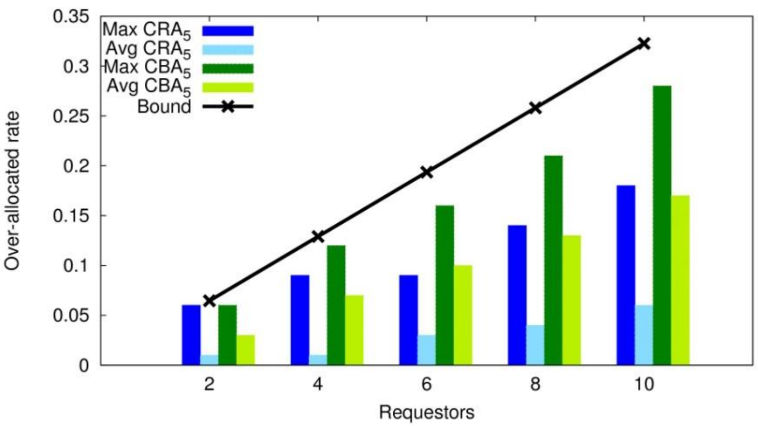
- We start by comparing the allocation properties of CRA and CBA
  - Average and maximum measured over-allocation compared to the bounds

- Description of use cases
  - We use bins with 2, 4, 6, 8 and 10 requestors
  - For each bin, we generate 1000 use cases
  - The total loads are uniformly distributed in the range [0, 100%]
  - Allocated burstinesses are real numbers in the range [1, 5]
  - Five bits of precision, results in access granularity of 1 / 31 = 3.3%

Over-Allocated Rate

- Maximum measured over-allocated rate:
  - Close to bound for both strategies with two requestors
  - Difference increases with the number of requestors
    - Worst-case allocation becomes increasingly unlikely, especially for CRA
  - We measure higher maximum over-allocated rate with CBA

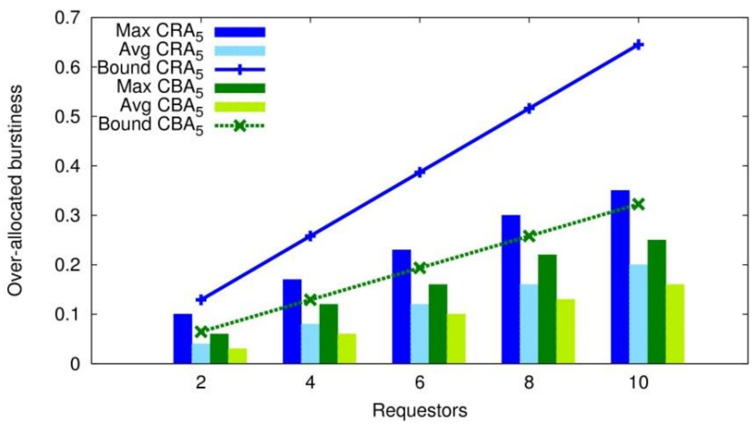
- Average over-allocated rate:
  - CRA performs better on average for all bins, as expected
  - CRA reduces average over-allocated rate with a factor three over CBA



Over-Allocated Burstiness

- Maximum measured over-allocated burstiness:
  - Similar as before, both strategies close to bound with few requestors
  - We observe higher maximum over-allocated burstiness with CRA

- Average over-allocated burstiness:
  - CBA outperforms CRA for all bins
  - Reducing average over-allocated rate by a factor three with CRA comes at the cost of 25% increase in over-allocated burstiness



Technische Universiteit Eindhoven University of Technology

Experiment 2 – Use Case Requirements

- Comparison of how CRA and CBA satisfies use case requirements
  - Use cases have high loads and hard service latency requirements

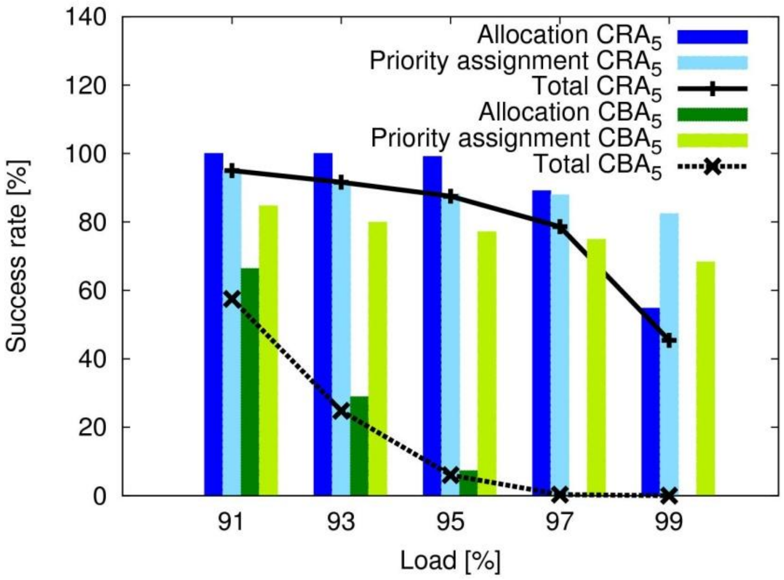
- Description of use cases
  - We use bins with 91%, 93%, 95%, 97%, and 99% total loads
  - For each bin, we generate 1000 use cases
  - Service latency requirements vary uniformly in range [0, 10000 ns]
  - Five bits of precision in rate regulator
  - Priorities are assigned using an optimal algorithm

- Interesting results are percentage of use cases where
  - all bandwidth requirements are satisfied
  - all latency requirements are satisfied
  - both bandwidth and latency requirements are satisfied

Successful Use Cases

- Bandwidth allocation:
  - CRA outperforms CBA significantly for use cases with high load, as expected
- Priority assignment:
  - CRA outperforms CBA for all bins
  - Over-allocated rate is worse than over-allocated burstiness in latency expression
- Total success rate:
  - CRA outperforms CBA for all tested loads
  - CRA satisfies more than 4x as many use cases with high load on average
  - CBA held back by allocation granularity

$$\Theta = \frac{\sum_{i=0}^{p-1} \sigma_i''}{1 - \sum_{i=0}^{p-1} \rho_i''}$$

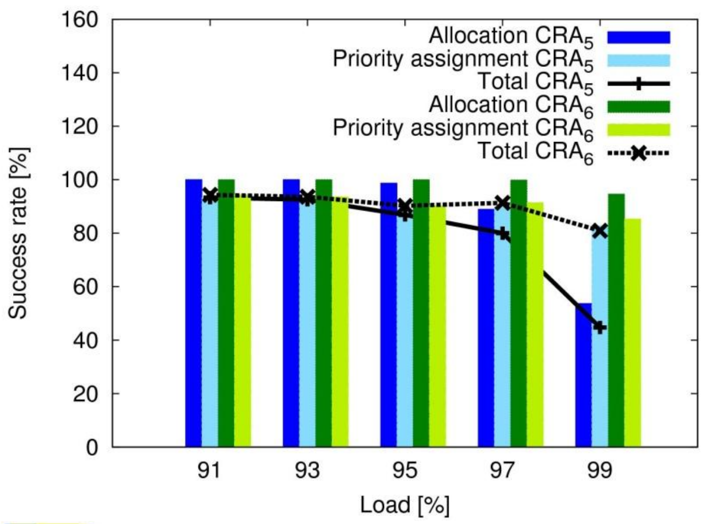


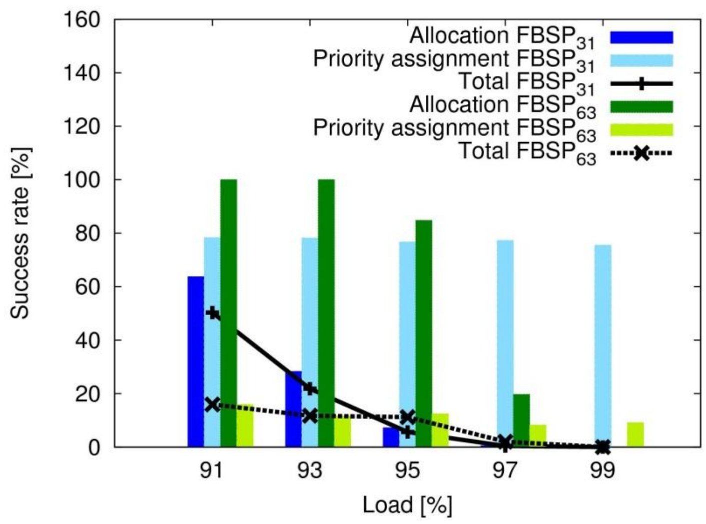
Experiment 3 – Increasing Precision

- We study the effects of increasing precision
  - We repeat previous experiment for CRA with 5 and 6 bits, respectively
- We compare with Frame-Based Static-Priority scheduler (FBSP)
  - Frame size set to 31 and 63 to provide same accuracy
  - Slots in frame allocated proportionally to allocated service rate
  - Details about latency for this combination in paper

CCSP Increasing Precision

- Success rate of CCSP increases with precision
  - Over-allocation and service latency both reduce monotonically
- Success rate of FBSP fluctuates with precision
  - Latency and rate are coupled!
  - Increasing precision is good for bandwidth allocation, but bad for latency





Technische Universiteit Eindhoven University of Technology

Presentation Outline

CCSP Overview

Service Allocation

Experimental Results

Conclusions

Summary and Conclusions

- We presented the hardware implementation of the CCSP rate regulator
  - Simple implementation of active period regulation
  - Accounting based on integer arithmetic with finite precision

- We introduced and compared two allocation strategies
  - Closest Rate Approximation (CRA)
  - Closest Burstiness Approximation (CBA)

- Conclusions:
  - We showed that increasing precision results in exponential reduction of over-allocation at cost of linear increase in area of the implementation
  - Over-allocation and latency reduces monotonically for CCSP with increased precision, unlike frame-based arbiters
  - The CRA strategy is preferred as it satisfies more use case requirements than CBA
  - Having a fine allocation granularity that is decoupled from latency is essential for resources with high loads in real-time systems

Questions?

k.b.akesson@tue.nl