



JUNE 23-27, 2024

MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA





JUNE 23-27, 2024

MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA

Window feedback based multi-master arbiter IP for efficient hardware resource sharing

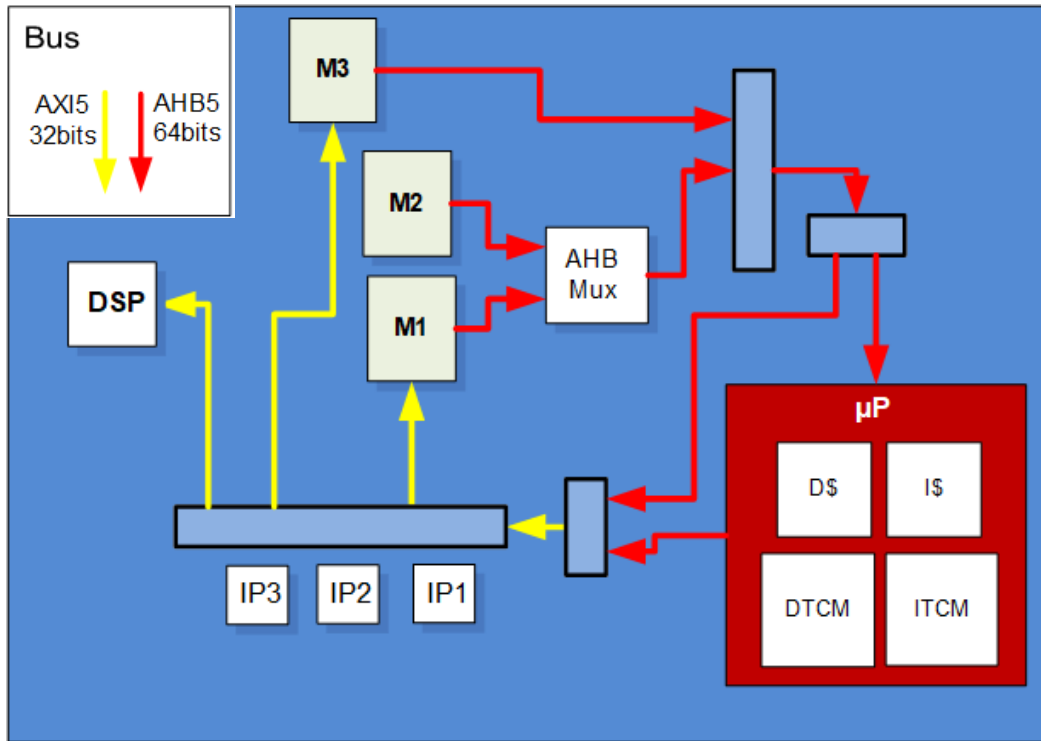
Gianluca Rigano, STMicroelectronics



Outline

- Bus communication and arbitration in SoC
- Standard arbitration algorithms
- Proposed algorithm
- Results
- Conclusions

Bus communication in SoC



Buses are the most widely used within SoC interconnection networks

A bus is a collection of signals (wires) to which one or more IP devices are connected (CPUs, memories, I/O devices)

On-chip interface standards define a specific data transfer protocol (e.g. AMBA – AXI, ACE)

- Broadly speaking, devices may be classified as:
 - **Master:** Those that initiate data transfers (CPU, GPU, memory controller..)
 - **Target:** Those that wait for requests (RAM, DMA, USB interface)

Generally, some devices can act as a master and a target, but not at the same time

Bus arbitration

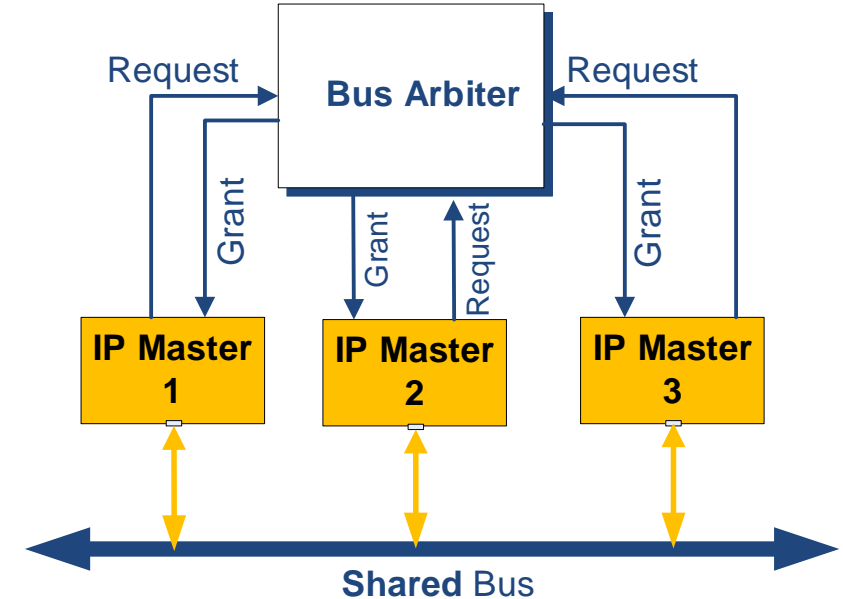


When multiple devices **attempt** to become the master, we need a bus arbitration mechanism to **prevent chaos** and **optimize** the flow of data.



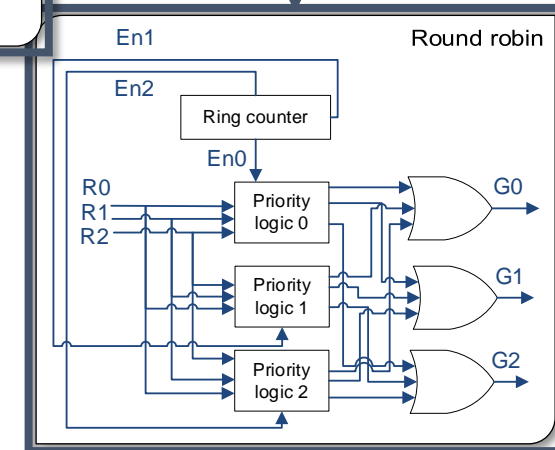
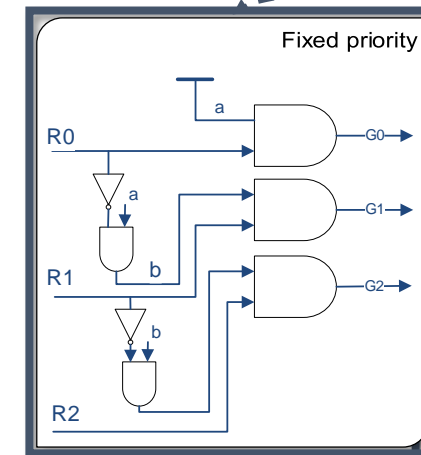
A centralized arbitration:

A **dedicated** bus arbiter, which determines which device is the next bus primary; thus, every device connects to the bus **arbiter** with one (or more) bus request and one (or more) bus grant lines.



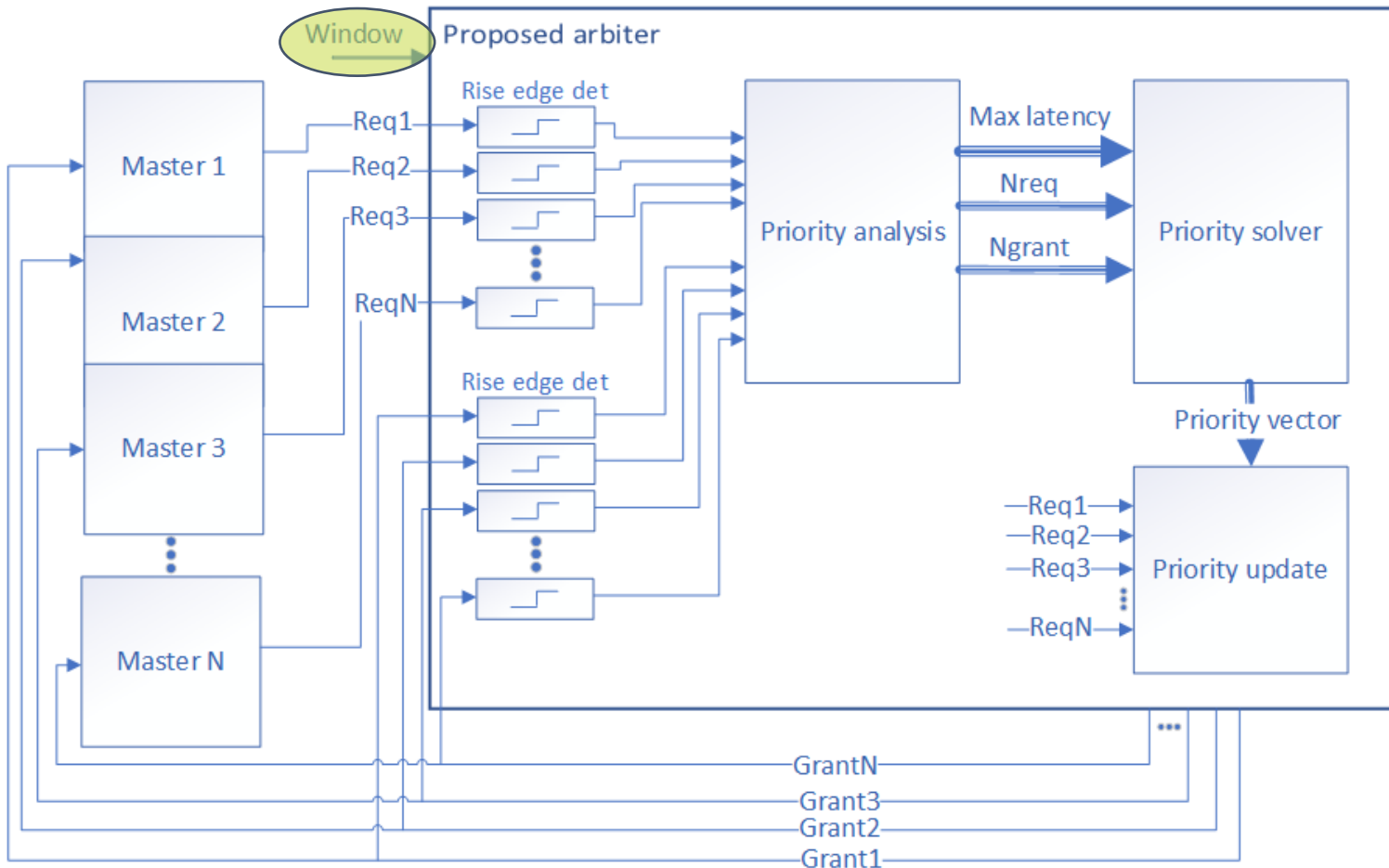
Typical arbitration algorithms

- Standard mechanisms (**open-chain** approaches):
 - Fixed priority arbiter
 - Simplest circuit
 - It could cause **starvation**
 - Just as always giving way to vehicles on the **same** road 😞
 - Round-Robin arbiter
 - Allow the highest-priority status to **rotate** among the master IPs
 - **Fair** arbitration is limited to the uniformly distributed active requests pattern
 - Just as allowing vehicles to pass at **fixed time** intervals 😊
- How to overcome standard algorithm limitations?



Proposed solution

Architecture of the arbiter:

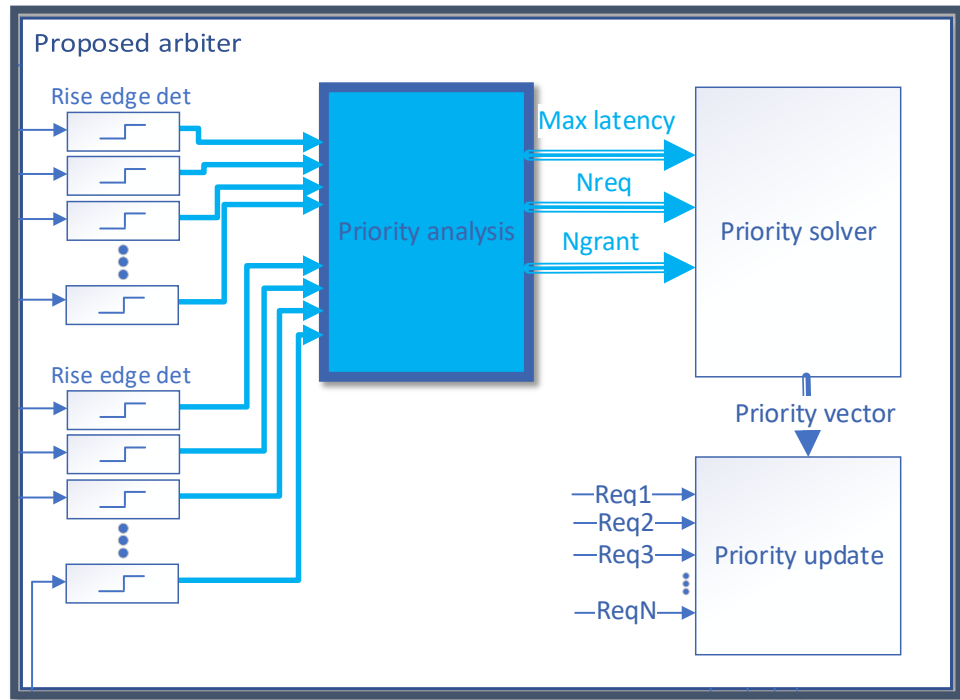
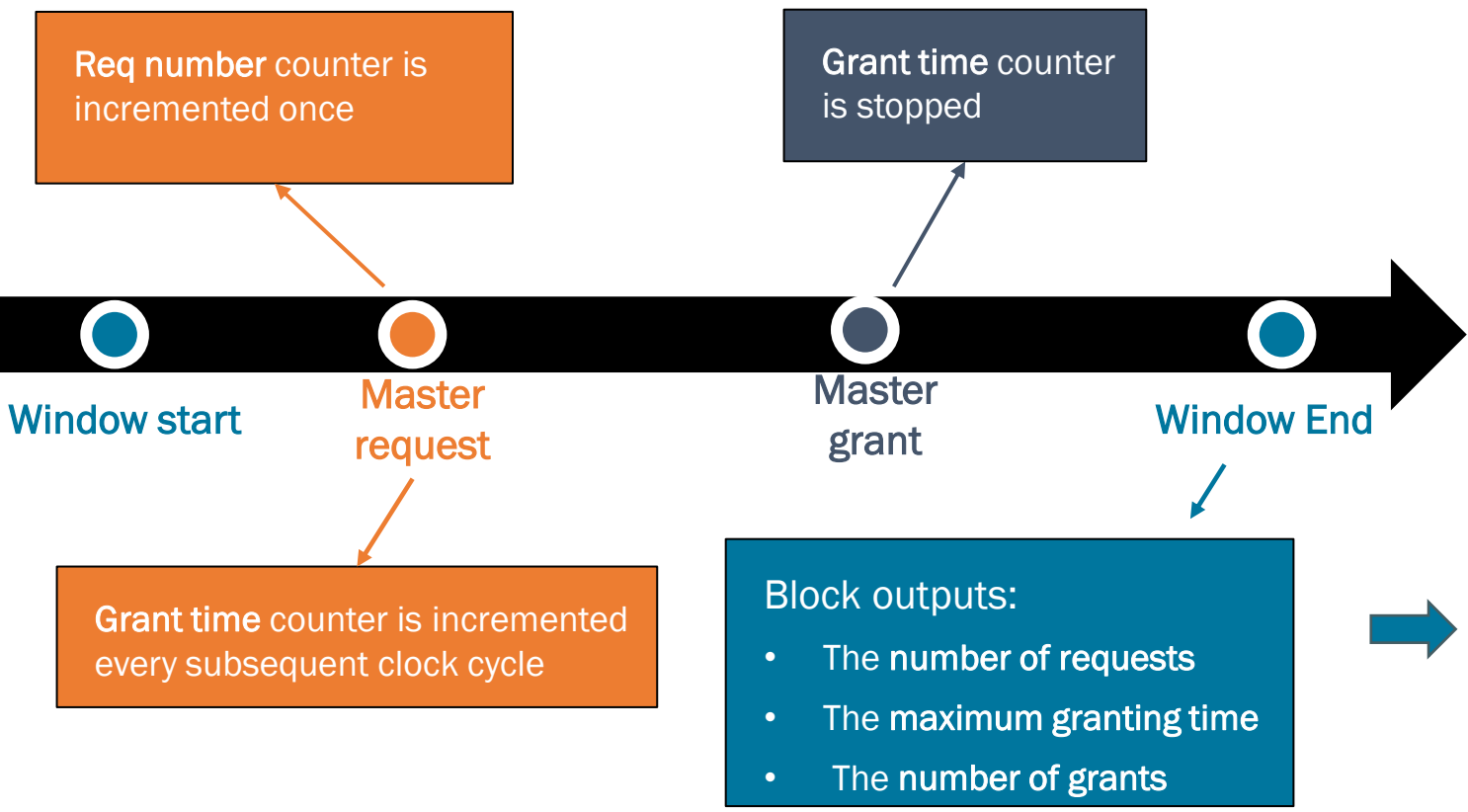


Closed-loop approach

A window is a time-frame, configurable according to the application

This approach can be applied to any req/ack standard bus interface (e.g. AMBA)

Priority analysis block

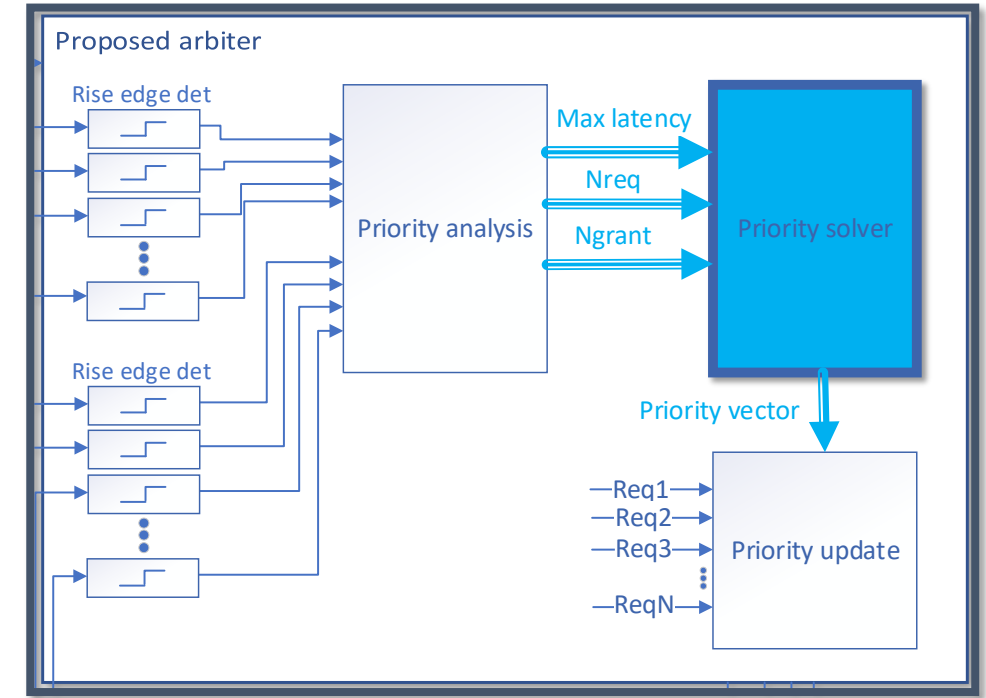
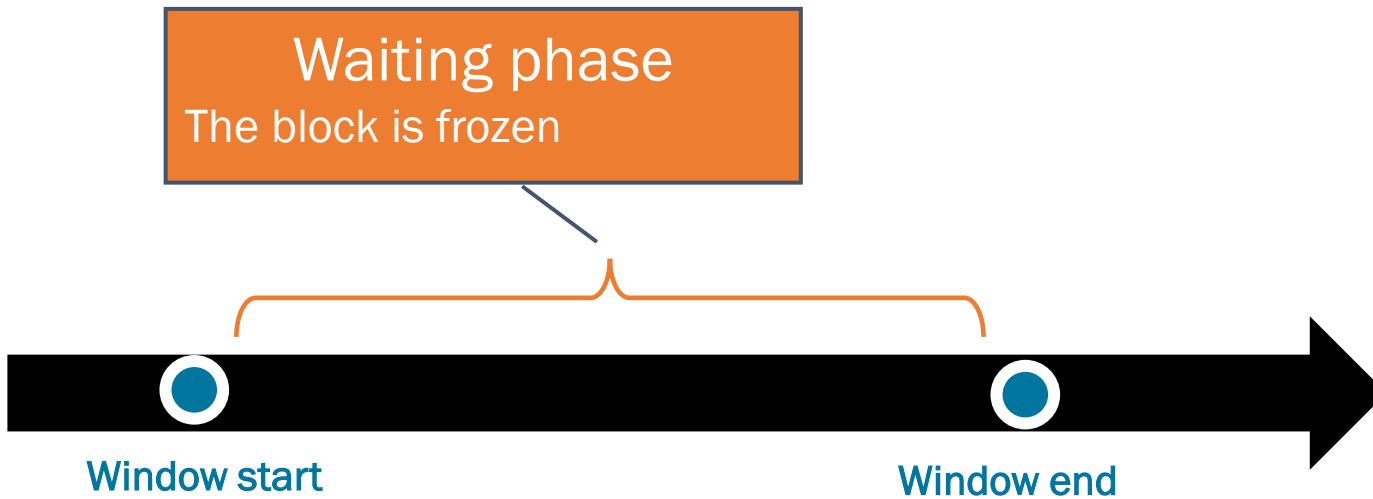


Example

Max grant time1 = 2 cycles
Req number1 = 2 requests

Max grant time2 = 8 cycles
Req number2 = 1 request

Priority solver block



Priority update

- A **weight** is calculated for each master device
- The **priority vector** for the next window is assigned

Example

Weight calculation

Weight1:

$\text{Max grant time1} \ll 1 + \text{Req_num1} = 6$

Weight2:

$\text{Max grant time2} \ll 1 + \text{Req_num2} = 17$

Priority vector calculation

$\text{Weight2} > \text{Weight1}$:
Priority vector: $0 \rightarrow 1$

Priority update block

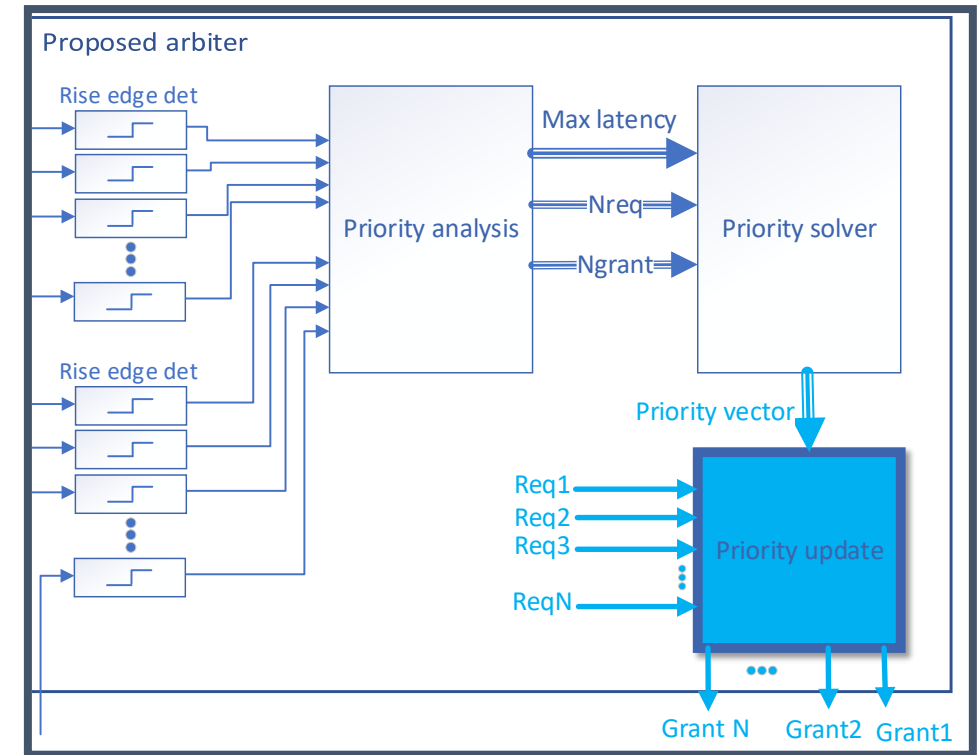
Fixed priority phase
The block applies a fixed priority policy

Window start

Window end

Change priority

- The block stores the new arbitration priority (defined by the **priority vector**)
- Until the next window, the bus will be arbitrated according to this priority



Example

Current window

Priority vector = 0
Highest priority: **Master1**
Lowest priority: **Master2**

Next window

Priority vector = 1
Highest priority: **Master2**
Lowest priority: **Master1**

Key features



Starvation - free

Formal analysis, by means of dedicated SVA assertions, proved that **no master devices can be blocked to access** the bus for an indefinite time.



Real-time operation

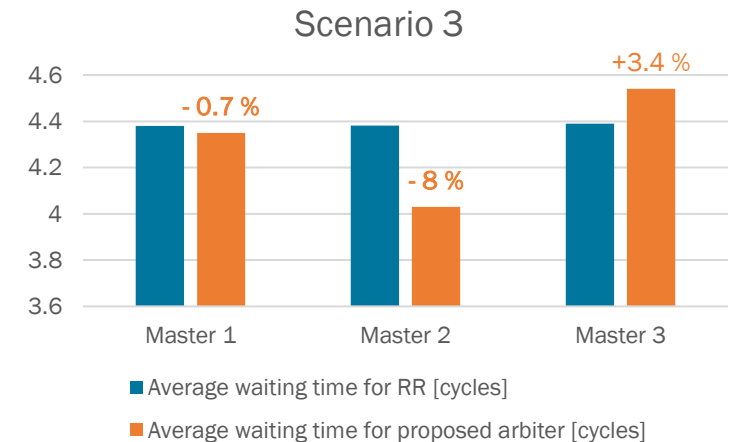
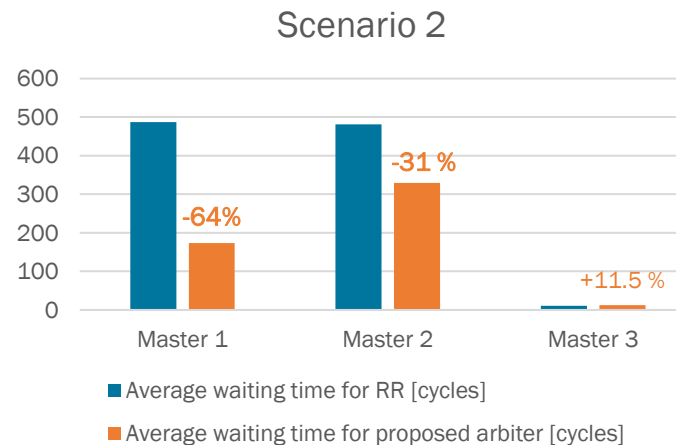
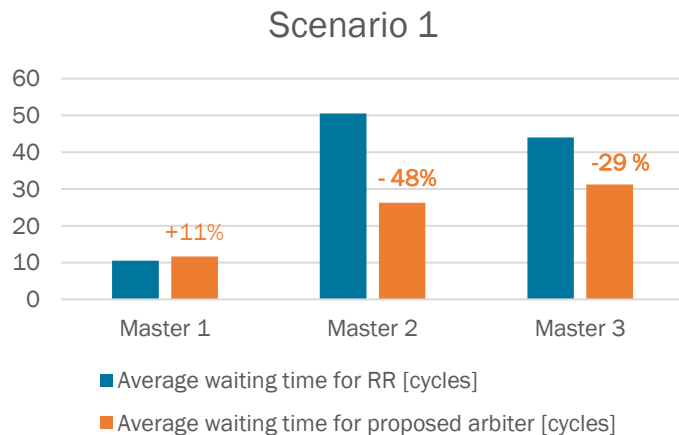
In case of a critical hardware system in which any peripheral connected to the shared must be served within a given time, **it is fundamental** to know exactly the maximum granting time for each device connected to **the shared bus**.

$$\text{Maximum granting time} \leq N_{\text{Master}} \cdot \text{Window}$$

Performance results

The **performance** of the proposed arbiter was compared with a standard solution (Round-Robin algorithm):

It was evaluated based on its **average waiting time**, which is the average time for a particular master in between request and grant of the shared bus.



The proposed algorithm demonstrates a significant improvement in performance (up to 64%) compared to Round-Robin (RR) when there is a **high disparity** in the **bus utilization time** among the master devices.

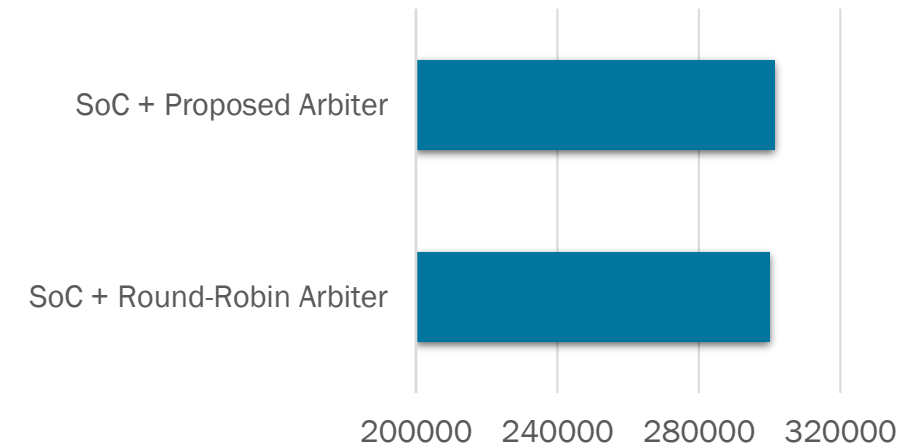
Synthesis results

The proposed arbiter has been **synthetized** through a commercial synthesis tool, by means of an ST technology (CMOS040).

SoC synthesis results	Total area [NAND2 gate Eq]	Target clock frequency [MHz]	Total area overhead [%]
SoC + Round-Robin arbiter	300k	200	0
SoC + proposed arbiter	301.44k	200	0.48

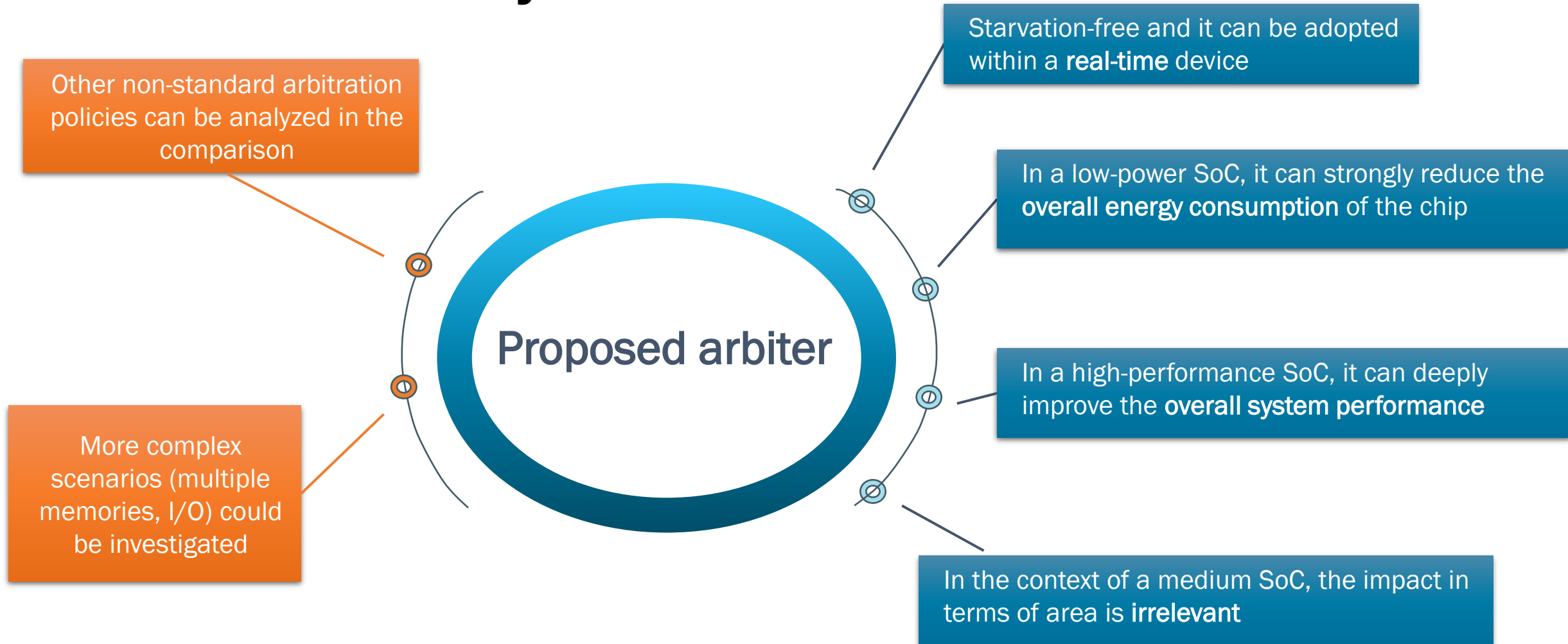


SoC Area [NAND2 Gate Eq]



In a highly complex architecture such as a microprocessor, such an increase in area is likely to have a **negligible impact** in percentage terms on the overall chip area.

Overall summary





JUNE 23-27, 2024

MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA

Thank you!

