MCSoC 15th IEEE International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC-2022)
Universiti Sains Malaysia, Penang, Malaysia, December 19-22,2022

A Reconfigurable Design of Flexible-arbitrated Crossbar Interconnects in Multi-core SoC system

Xuewen He, Yajie Wu, Yichuan Bai, Jie Liu, Li Du, and Yuan Du
Nanjing University, Nanjing, China
Email: {ldu, yuandu}@nju.edu.cn

ISCL (Intelligent Sensing and Communication Lab), Nanjing University, China

Contents

- Introduction
- Related Work & Motivation
- Static Reconfiguration Design
- Dynamic Reconfiguration Design
- Conclusion & Future Work

Introduction

Main bus architecture in multi-core SoC system

- Common shared bus
- Crossbar
- Network-on-Chip (NoC)

Bus Type	Design Complexity	Connection Mode	Efficiency in common multi-core SoC system	Efficiency in ultra-complex SoC system
Shared Bus	Low	Sync	Low	Low
Crossbar	Medium	Sync	High	Medium
NoC	High	Async	Medium	High

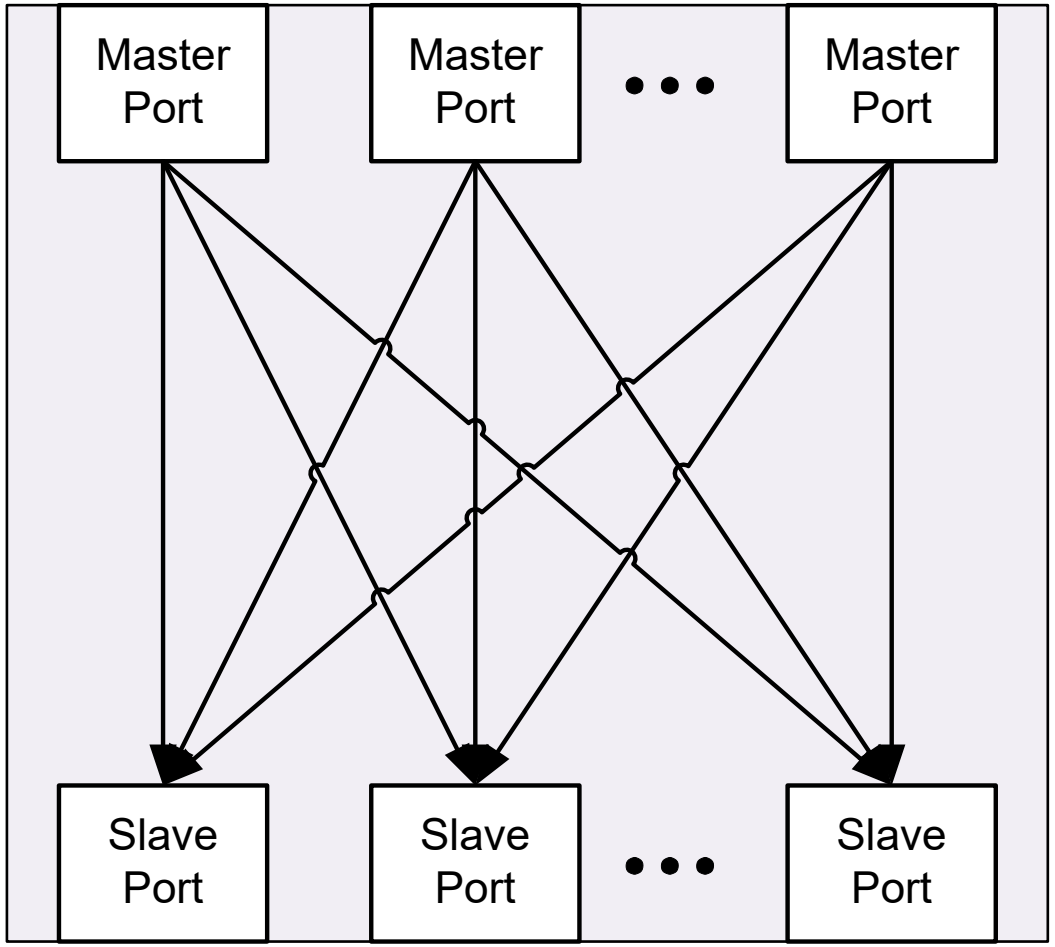
Introduction (Cont.)

Crossbar usage and influence

- Common to the mainstream multi-core SoC
- Great impact on the latency, area, power and transmission efficiency of the SoC system

Two principles to design crossbar

- The inherent communication requirement
- The application requirement on broadband, delay, efficiency and adaptability



Related Works

The related designs of crossbar

- A low-power interconnect for implant SoC with a significant power reduction by Lyrakis, Alexios, et al

- A static receiver-based allocation had been used to optimize the port utilization of the crossbar by Khaled, E, Ahmed, et al

- S. Bansal and S. K. Virdi et al focused on building the control logic system with the crossbar

Related Works (Cont.)

The related designs of arbitration algorithms

- The probabilistic round-robin algorithm improved the SoC performance by Ganiee, Sajad
- Y. Guo et al. proposed two arbitration optimization methods to reduce logic overhead
- A.M.Legkikh claimed the wrapped wavefront algorithm was better than round-robin on reducing the decision time of bus arbitration

Motivation

The static reconfiguration

- The number of master ports
- The number of slave ports
- The definition for priority matching algorithms

Two proposed arbitration algorithms

- The sequential priority matching (SPM) algorithm
- The starvation degree matching (SDM) algorithm

The dynamic reconfiguration

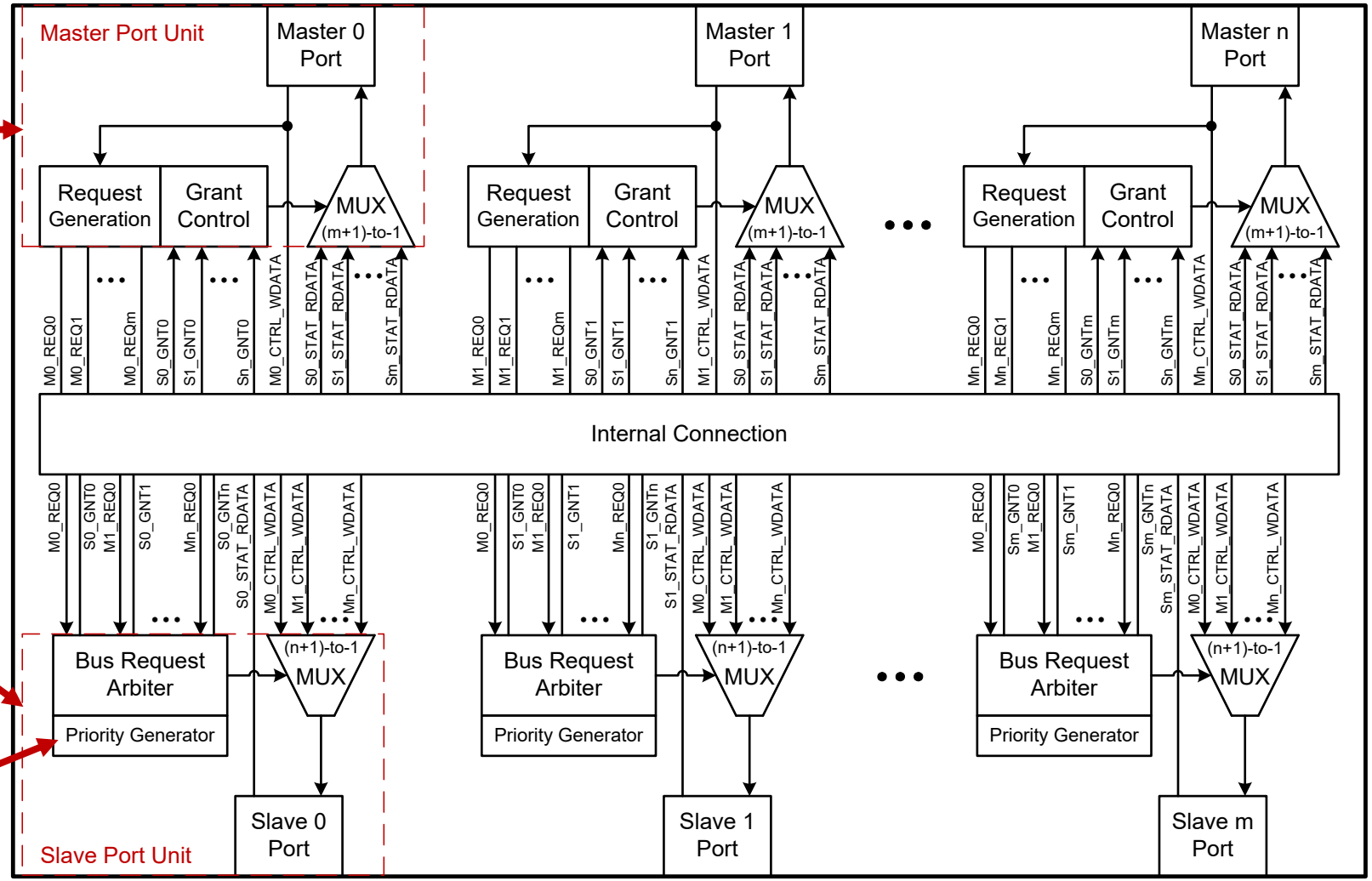
- The enabling control of master and slave ports
- The control of bus combining and dividing
- The selection of 4 arbitration algorithms

Static Reconfiguration Design

1. The number of master ports

2. The number of slave ports

3. The priority algorithms



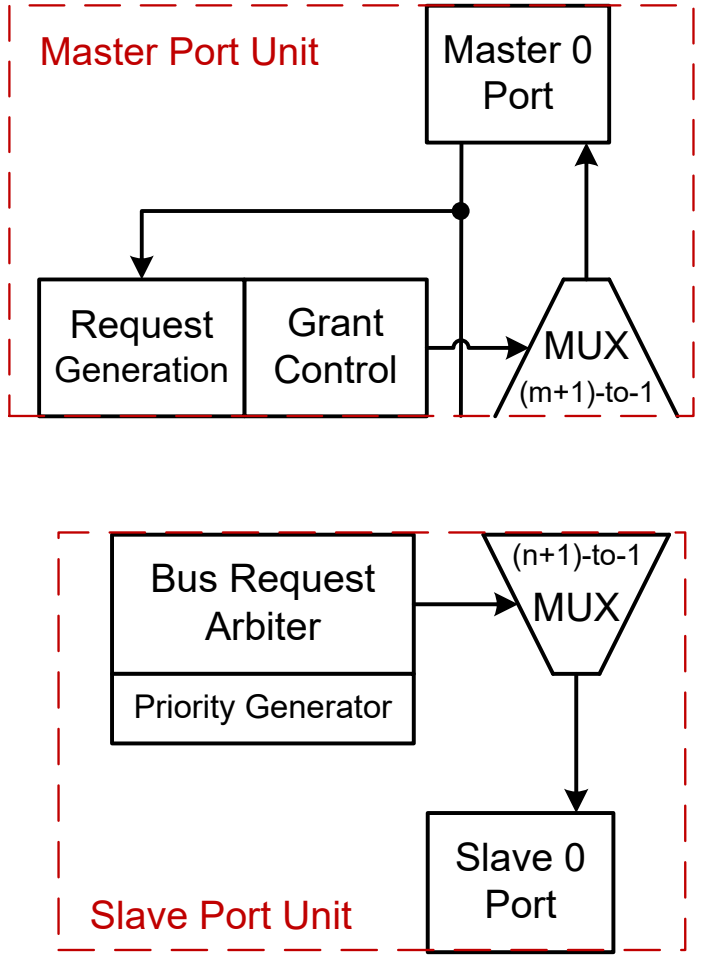
Static Reconfiguration Design (Cont.)

Master Port Unit

- The Request Generator
  - Bus Requesting
  - Bus Waiting
- The Grant Control
- The slave-to-master data multiplexers ((m+1)-to-1 MUX)

Slave Port Unit

- The Bus Request Arbiter
- The Priority Generator
- The master-to-slave multiplexers ((n+1)-to-1 MUX)



Static Reconfiguration Design (Cont.)

Fixed priority matching (FPM) algorithm

- Initialized by design parameter
- Reconfigurable by user software in real-time

Basic round-robin matching (RMM) algorithm

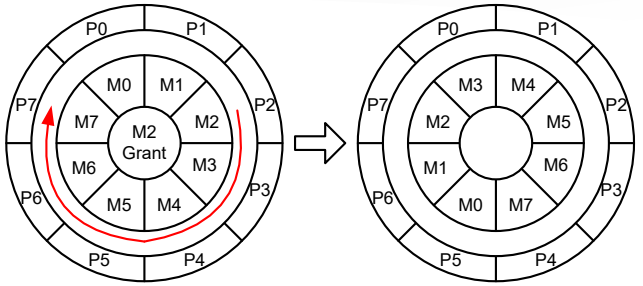
- The granted master rotates to the lowest priority
- All masters keep the relative position unchanged on the disc

Proposed sequential priority matching (SPM) algorithm

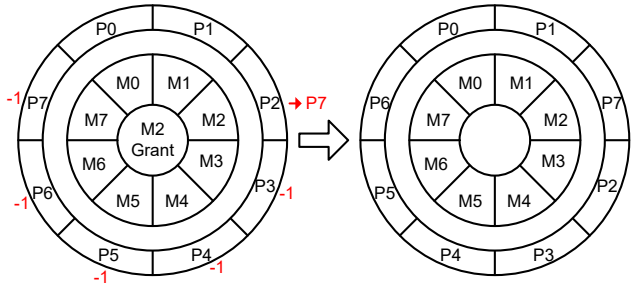
- The latest granted master becomes the lowest priority
- The masters with lower priorities are upgraded by one level
- The other masters remain their priorities unchanged

Proposed starvation degree matching (SDM) algorithm

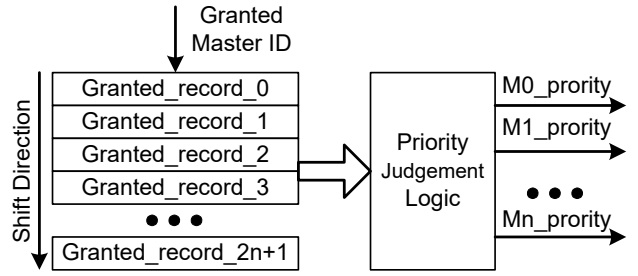
- The priorities are only determined on the starvation degrees



(a) Basic RRM algorithm



(b) Proposed SPM algorithm



(c) Proposed SDM algorithm

Static Reconfiguration Design (Cont.)

The comparison of 4 priority algorithms

- The maximum latency

$$t_{\max} = MAX\left(t_{M0}^{1}, t_{M0}^{2}, \ldots, t_{M0}^{10000}, t_{M1}^{1}, \ldots, t_{M7}^{10000}\right)$$

- The average latency

$$t_{avg} = \frac{\sum\left(t_{M0}^{1}, t_{M0}^{2}, \ldots, t_{M0}^{10000}, t_{M1}^{1}, \ldots, t_{M7}^{10000}\right)}{10000 \times 8}$$

TABLE I. THE LATENCY COMPARISON OF 4 PRIORITY ALGORITHMS

Master Load	Latency (cells*)	FPM	Basic RMM	Proposed SPM	Proposed SDM
1/8	t_{avg}	1.148	1.255	1.242	1.245
	$t_{\max}$	58	7	7	10
1/4	t_{avg}	1.141	1.232	1.223	1.213
	$t_{\max}$	71	7	7	11
1/2	t_{avg}	2.494	3.969	3.941	3.632
	$t_{\max}$	1834	7	7	13
1	t_{avg}	3.5	6.996	6.996	6.530
	$t_{\max}$	70000	7	7	8

*The unit means the time of a basic bus transmission without waiting.

Static Reconfiguration Design (Cont.)

The comparison of 4 priority algorithms

- FPM: best average latency but worst max latency
- RMM: good performance and low hardware overhead
- **SPM**: better transmission balance and average latency than RMM
- **SDM**: a trade-off between average latency and maximum latency

TABLE I. THE LATENCY COMPARISON OF 4 PRIORITY ALGORITHMS

Master Load	Latency (cells*)	FPM	Basic RMM	Proposed SPM	Proposed SDM
1/8	t_{avg}	1.148	1.255	1.242	1.245
	t_{max}	58	7	7	10
1/4	t_{avg}	1.141	1.232	1.223	1.213
	t_{max}	71	7	7	11
1/2	t_{avg}	2.494	3.969	3.941	3.632
	t_{max}	1834	7	7	13
1	t_{avg}	3.5	6.996	6.996	6.530
	t_{max}	70000	7	7	8

*The unit means the time of a basic bus transmission without waiting.

Static Reconfiguration Design (Cont.)

Evaluation environment for the crossbar

- The HVT Library of TSMC 40nm ULP process
- 1.1V voltage, TT process corner and 25℃

Performance conclusion

- Area: the slave port > the master port
- Area with more ports: super-linearly increasing
- The number of ports increases the hardware overhead of each port

TABLE II. THE PERFORMANCE COMPARISON OF STATIC RECONFIGURABLE IMPLEMENTATIONS

Port Number		AREA (um^2)	Latency (ns)	Power (uW)
Master	Slave			
8	8	11205.63	1.019	96.3
6	8	9272.46	0.893	75.1
8	6	9167.15	0.936	73.6
4	8	7391.68	0.689	61.6
8	4	7154.07	0.852	58.2
4	6	5939.91	0.611	48.7
4	4	4476.50	0.585	37.8

Static Reconfiguration Design (Cont.)

Timing Evaluation

- The latency of writing path

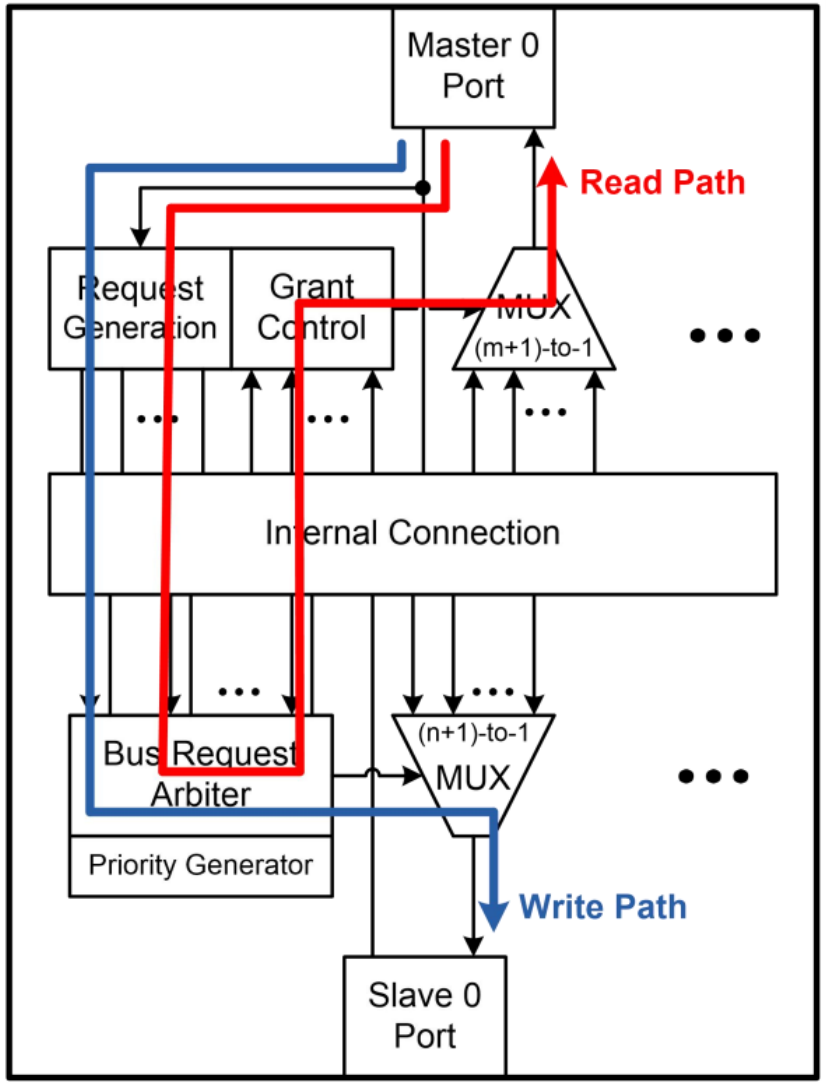
$$t_{WR} = t_{M\_RG} + t_{S\_ARB} + t_{S\_MUX}$$

- The latency of read path

$$t_{RD} = t_{M\_RG} + t_{S\_ARB} + t_{M\_GC} + t_{M\_MUX}$$

Timing conclusion

- The number of master ports has a more substantial impact on the delay of crossbar



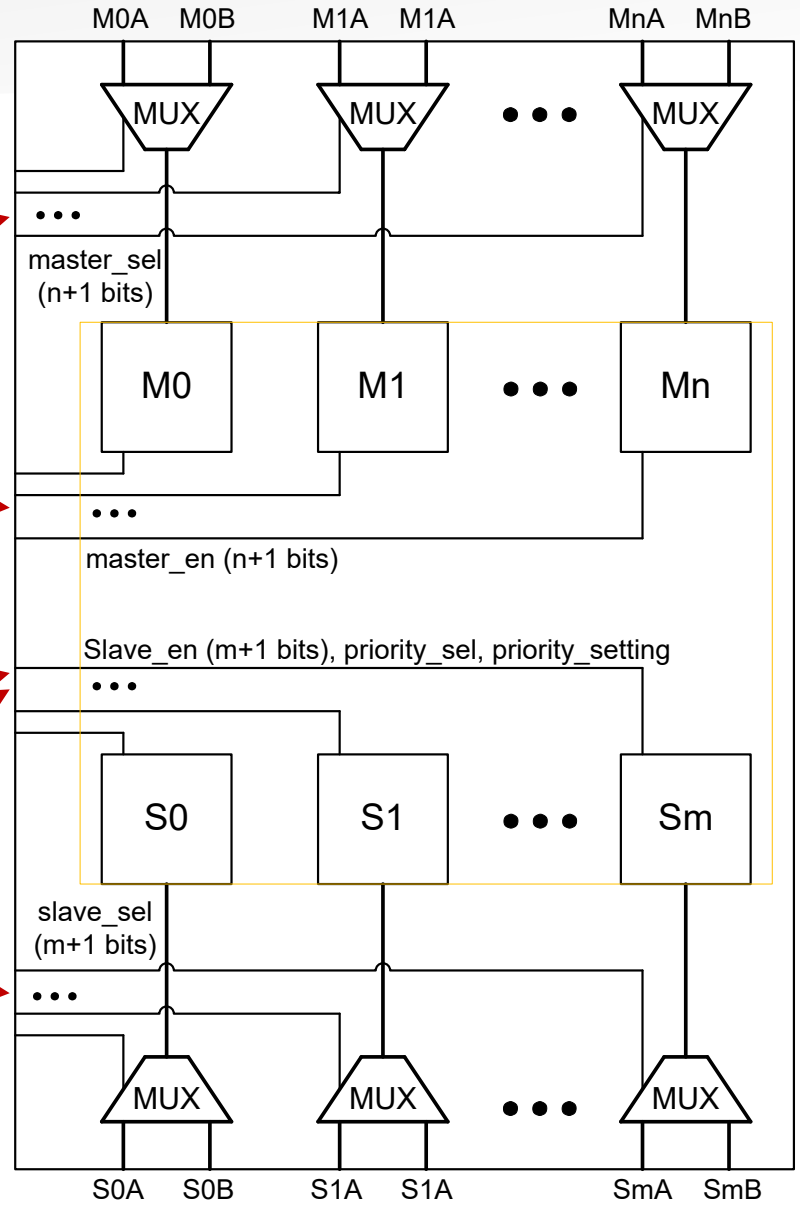
Dynamic Reconfiguration Design

Dynamic reconfiguration for master ports

- Bus combining
- The enabling control of master ports

Dynamic reconfiguration for slave ports

- The enabling control of slave ports
- The selection of arbitration algorithms
- Bus dividing



Dynamic Reconfiguration Design (Cont.)

Evaluation environment for dynamic ports

- The HVT Library of TSMC 40nm ULP process
- 1.1V voltage, TT process corner and 25℃

Performance conclusion

- The area of the dynamic reconfigurable implementation increases by 13.8%
- The latency of critical path increases by 2.0%
- When part of ports are switched off, the power is significantly reduced

TABLE III. THE PERFORMANCE RESULT OF DYNAMIC RECONFIGURATION

Active Port Number		AREA (um^2)	Latency (ns)	Power (uW)
Master	Slave			
8	8	12756.54	1.039	102.3
6	8	12756.54	1.039	92.5
8	6	12756.54	1.039	91.7
4	8	12756.54	1.039	83.7
8	4	12756.54	1.039	82.6
4	6	12756.54	1.039	72.8
4	4	12756.54	1.039	63.6

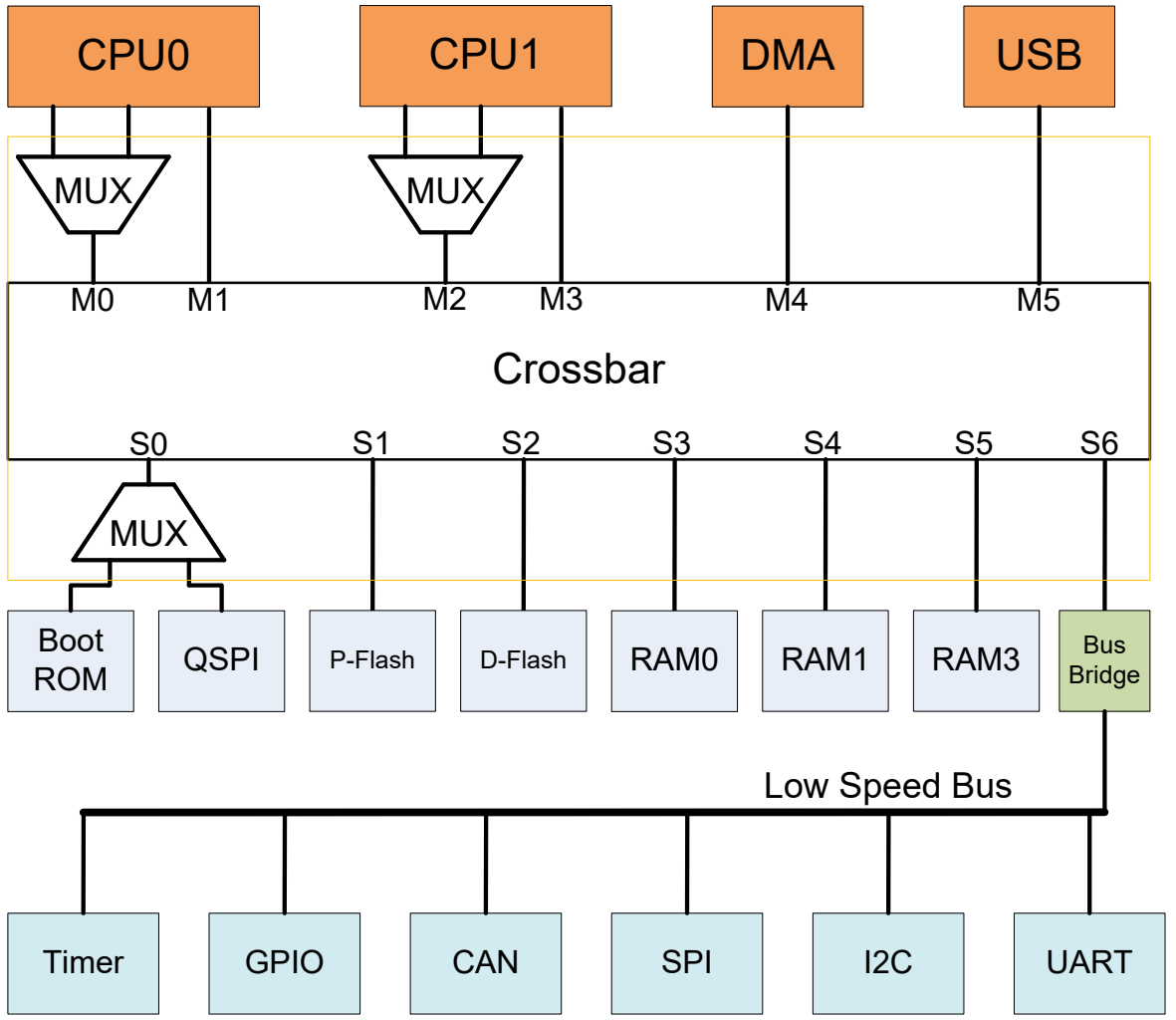
Dynamic Reconfiguration Design (Cont.)

Case analysis for reconfiguration

- Bus combining for CPU0 and CPU1
- Bus dividing for S0 of the crossbar
- The enabling control for the ports

Performance conclusion

- The area is reduced by 23.3%
- The latency is reduced by 15.7%
- The power is reduced by 23%
- The power can be further reduced with switching off some ports



Conclusion & Future Work

Conclusion

- A flexible-arbitrated crossbar is introduced with two proposed arbitration algorithms, the static reconfiguration, and the dynamic reconfiguration
- An application scenario in multi-core SoC system is exemplified to illustrate significant improvement on the performance of the crossbar
- A valuable reference is provided for bus design in SoC system

Future work

- The design of bus cache
- The functional safety of end-to-end (E2E) bus

Thank you!

If any question, please contact:

yuandu@nju.edu.cn

ldu@nju.edu.cn