Designing multi-resource arbiter

Stanislavs Golubcovs Andrey Mokhov Alex Yakovlev

Newcastle University

20^{th} UK Async Forum, 1–2 September 2008

Summary

1 Introduction

2 Arbiter design

3 Verification and Analysis

4 Conclusions

Newcastle University

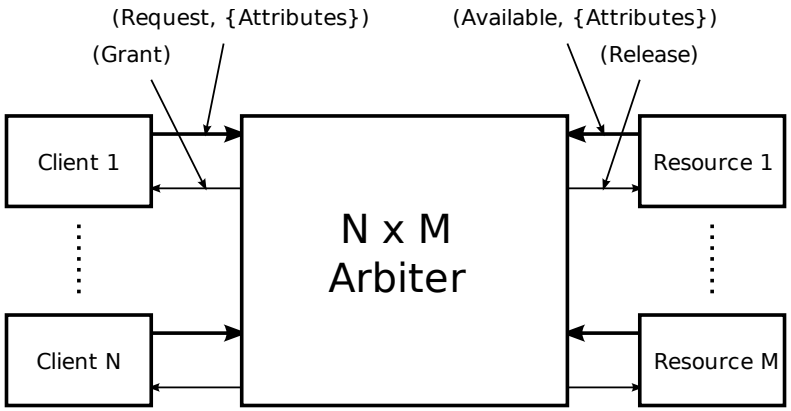
Idea of arbitration

Some resources is used by many agents (clients). However, not more than one such agent can actually use the resource at the same time. Arbiter is the entity that manages client access.

- Cars + Cross roads $\longrightarrow$ Traffic lights
- Processes + Critical sections in the program code $\longrightarrow$ Semaphores
- Data + Memory, Processor, Data channels $\longrightarrow$ Arbiters

Newcastle University

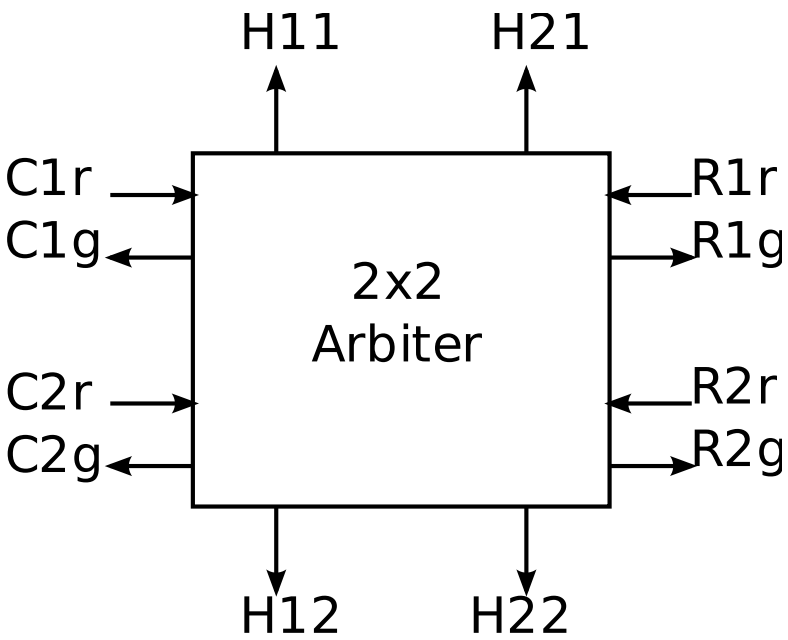
General arbiter structure



- Resources inform about their availability
- Arbiter responds by allowing certain pairs of handshakes

Newcastle University

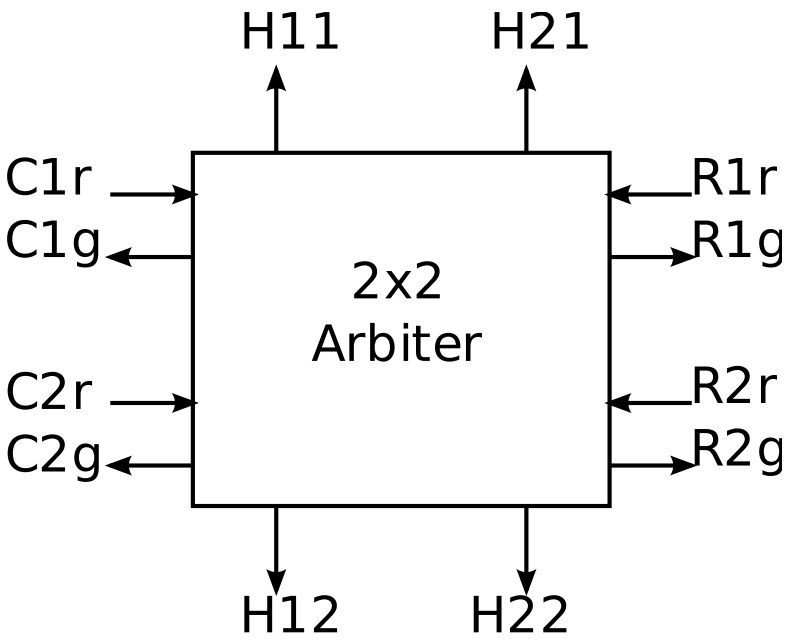
Task specification

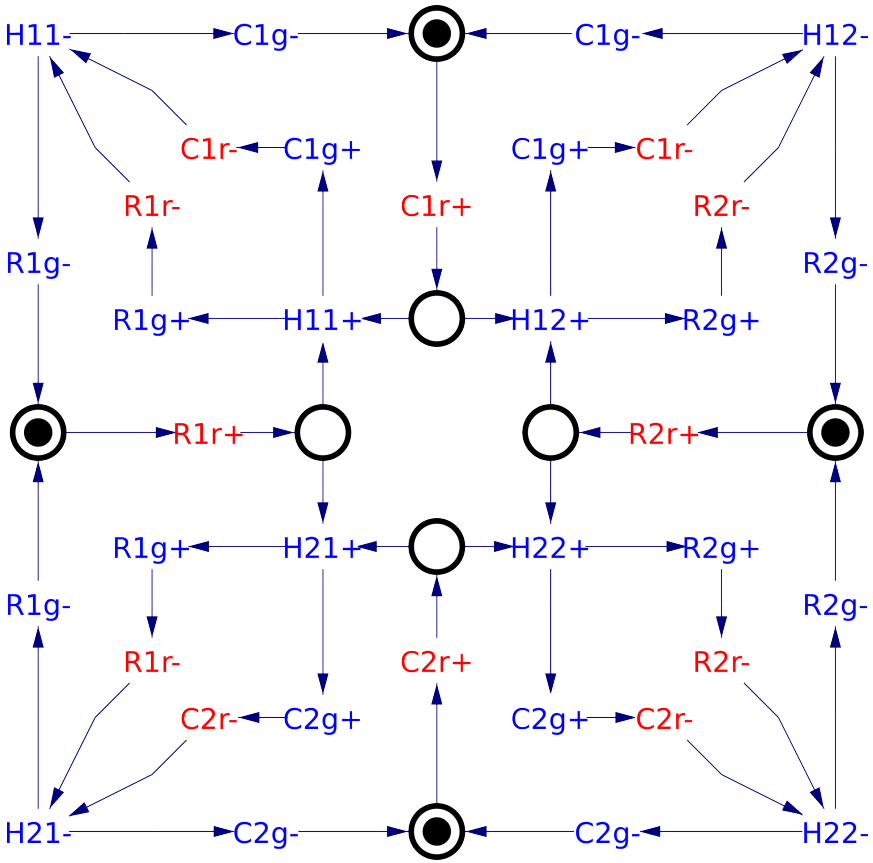


- Design the 2×2 multi-resource arbiter
- Find SI, asynchronous implementation
- Allow simultaneous handshakes
- Do not allow conflicting handshakes
- Minimise response latency

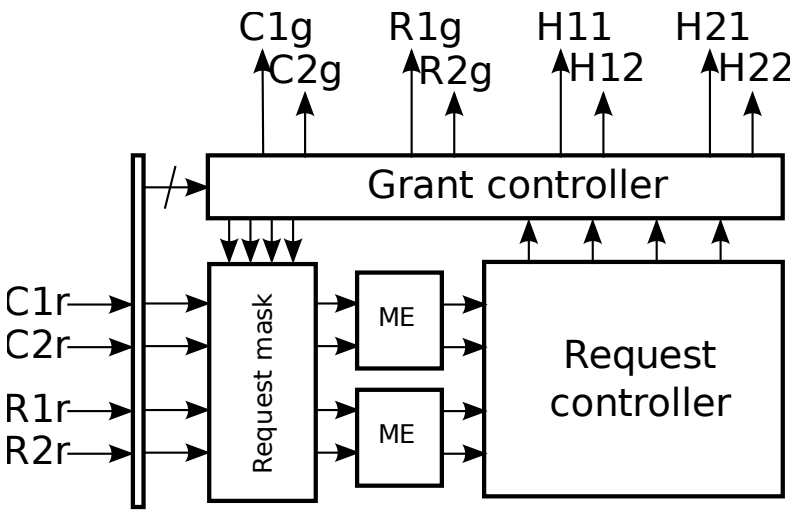
Newcastle University

Task specification





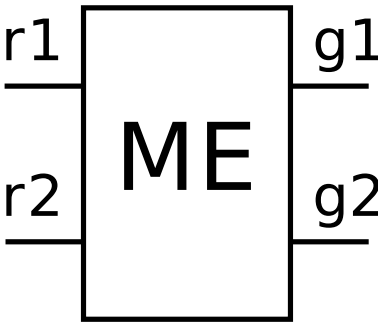
Overall structure



- MUTEX elements ensure not more than one channel can be selected by the request controller
- Request controller selects the channel to be activated
- Grant controller returns grant signals and activates the request mask
- Request mask hides initial requests to free MUTEX-es and open the Request controller for further requests

Newcastle University

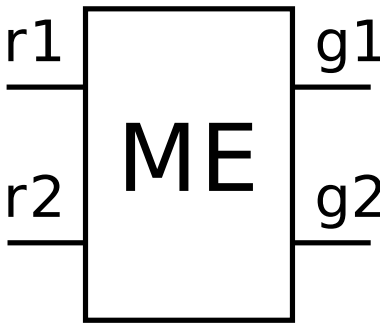
Key components

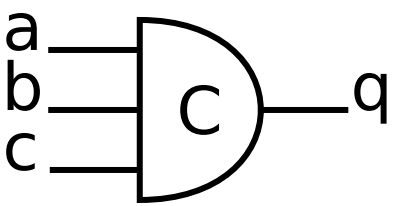


- MUTEX element
- Three-input C-element
  $q = a \cdot b \cdot c + q \cdot (a + b + c)$
- Asymmetric C-element
  $q = a \cdot b \cdot c \cdot d + q \cdot (c + d)$

Newcastle University

Key components

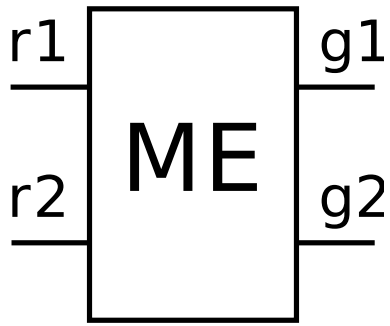




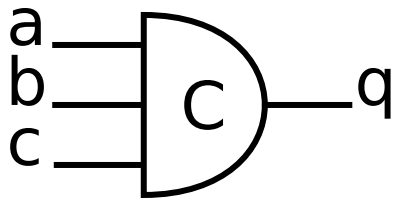
- MUTEX element
- Three-input C-element
  $q = a \cdot b \cdot c + q \cdot (a + b + c)$
- Asymmetric C-element
  $q = a \cdot b \cdot c \cdot d + q \cdot (c + d)$

Newcastle University

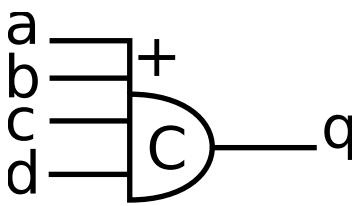
Key components



- MUTEX element



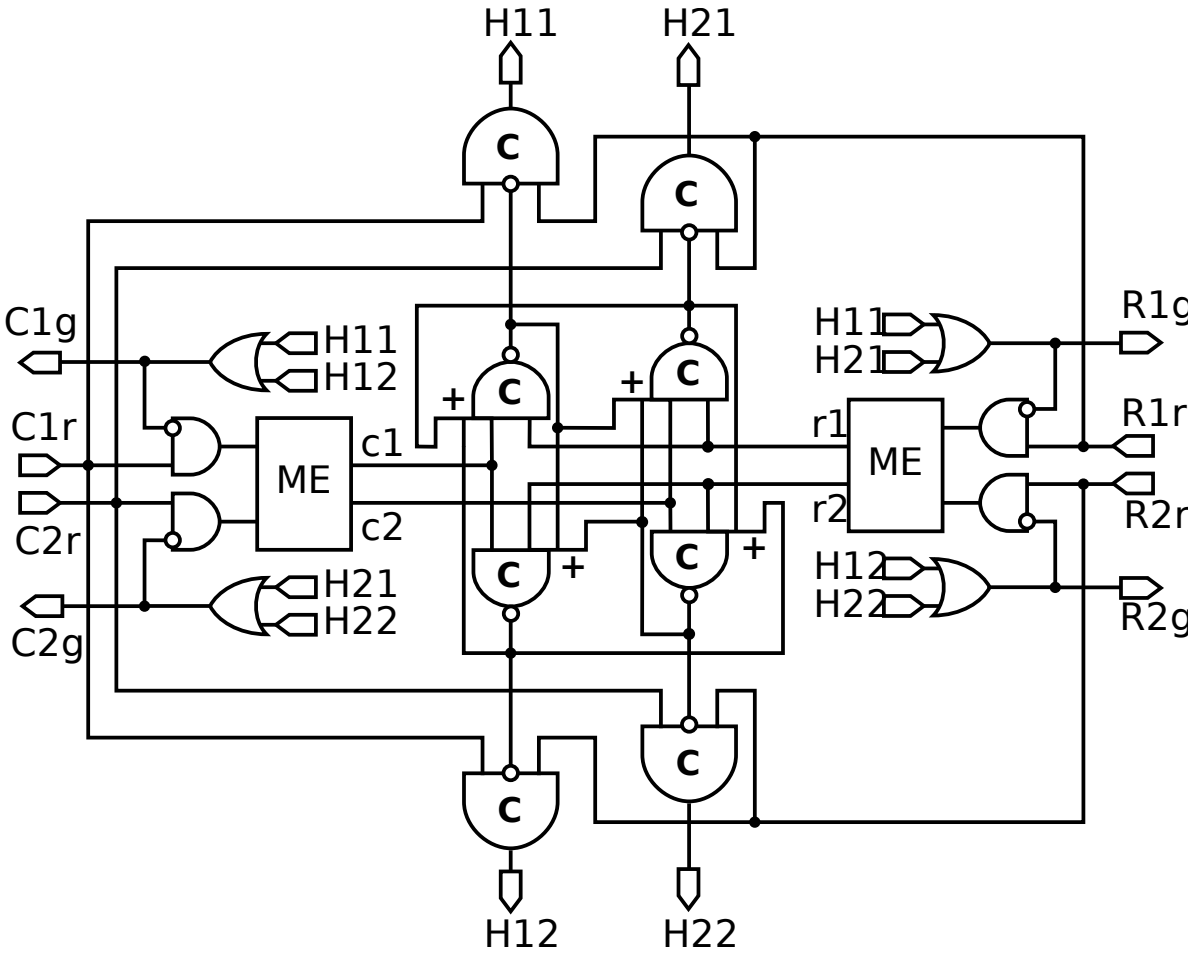
- Three-input C-element
  $q = a \cdot b \cdot c + q \cdot (a + b + c)$



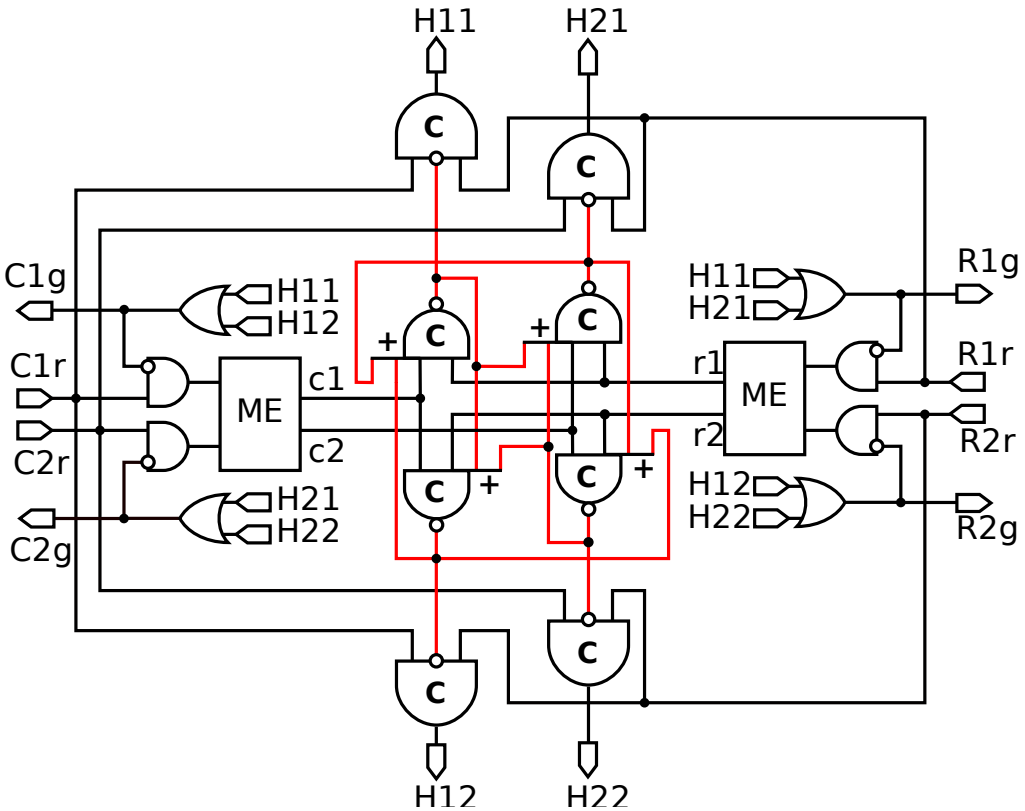
- Asymmetric C-element
  $q = a \cdot b \cdot c \cdot d + q \cdot (c + d)$

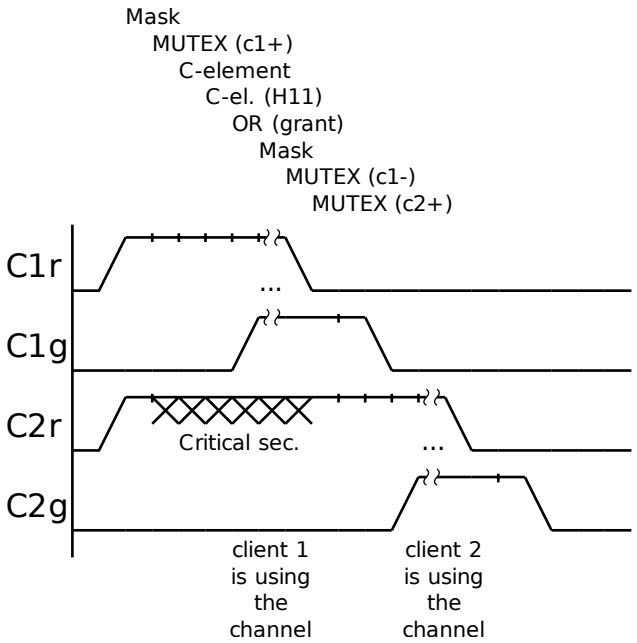
Newcastle University

Implementation

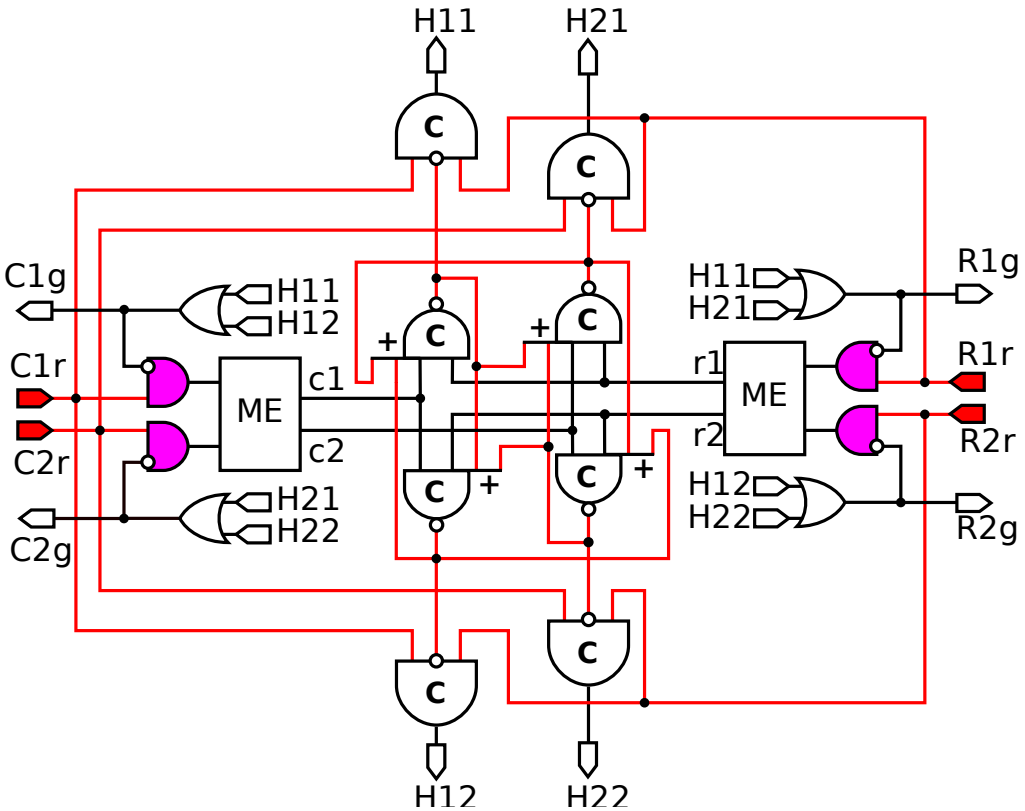


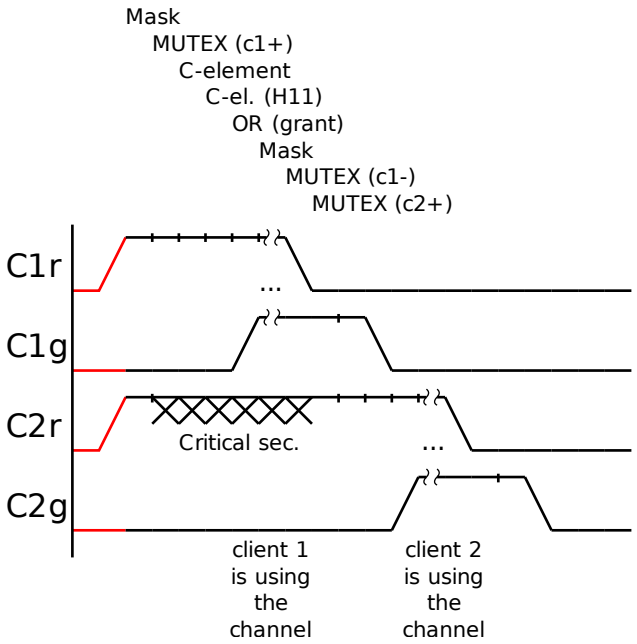
Timing estimation



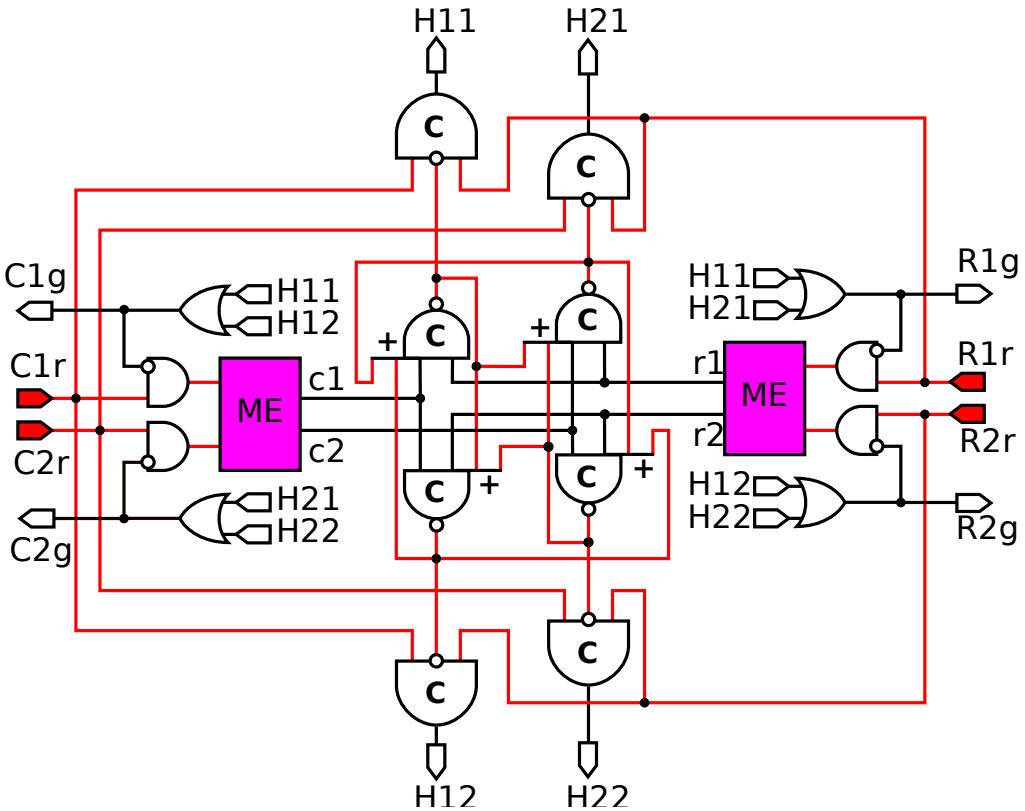


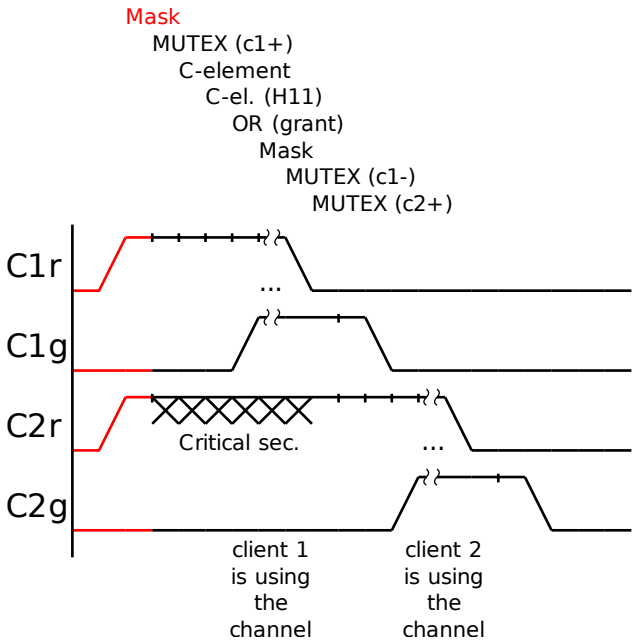
Timing estimation



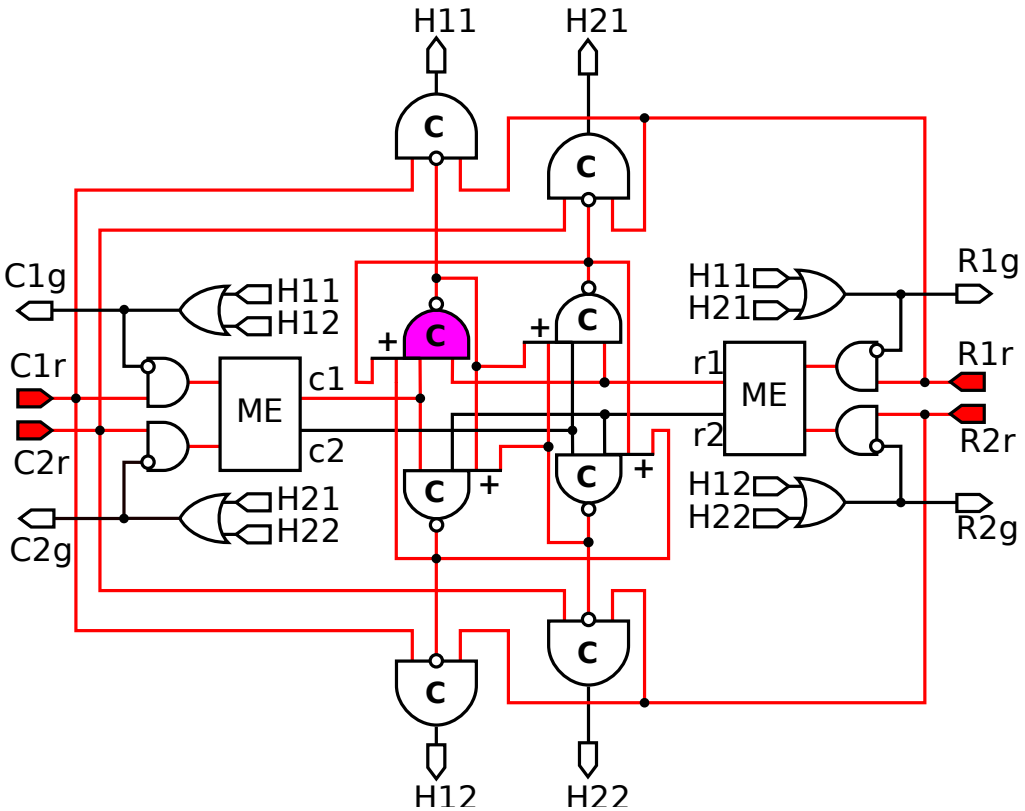


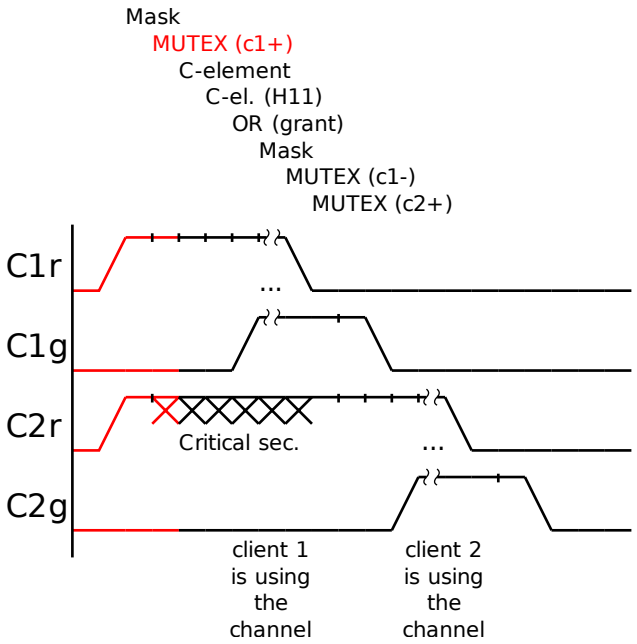
Timing estimation



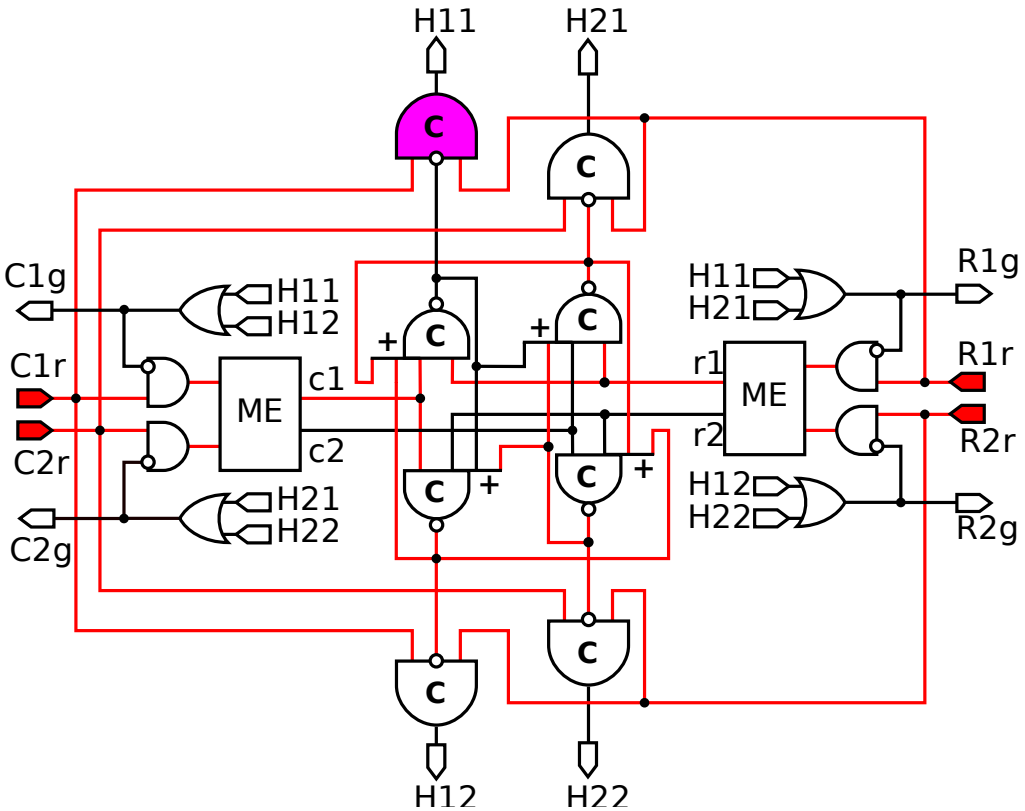


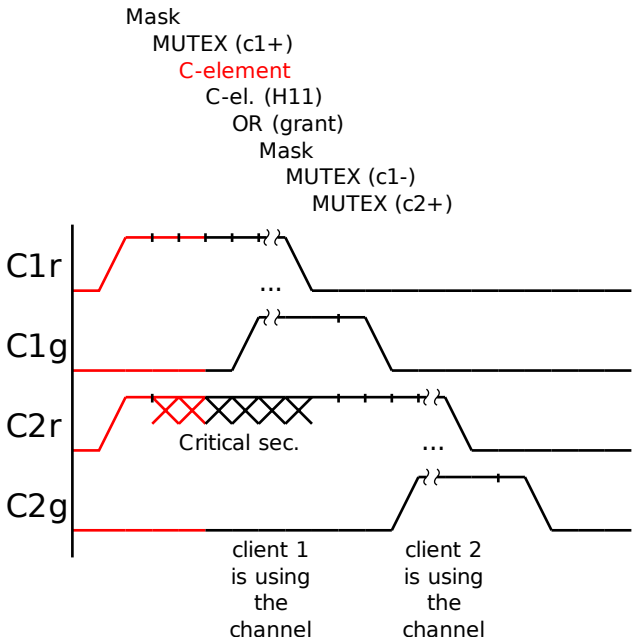
Timing estimation



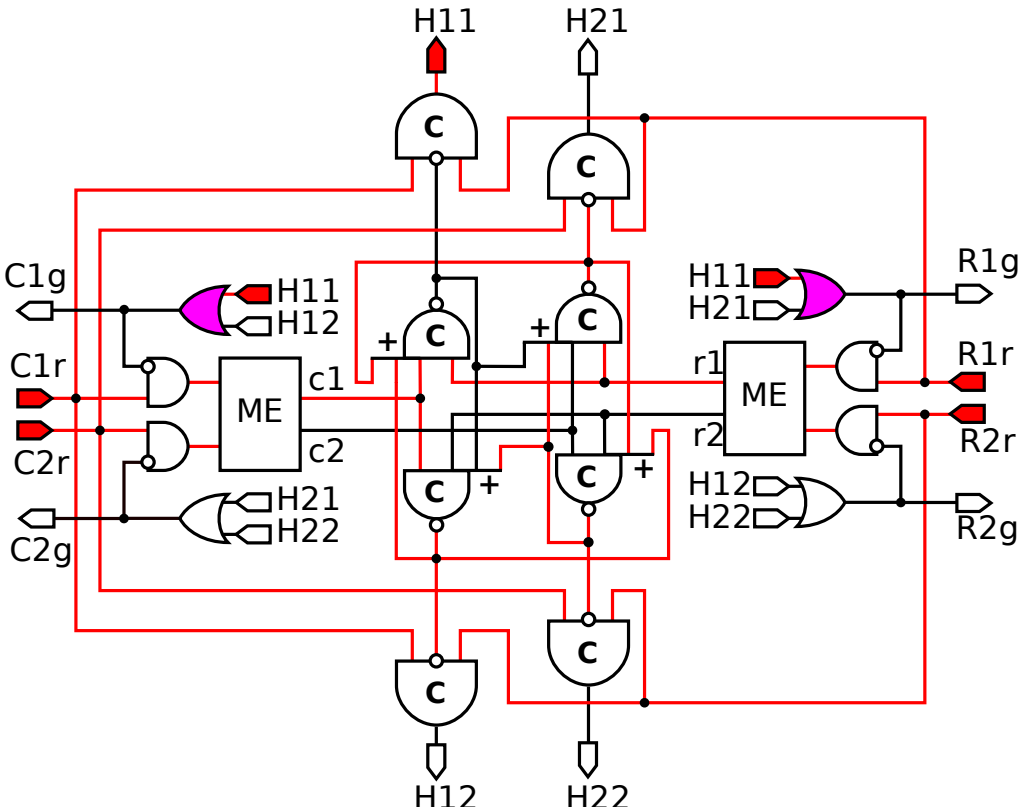


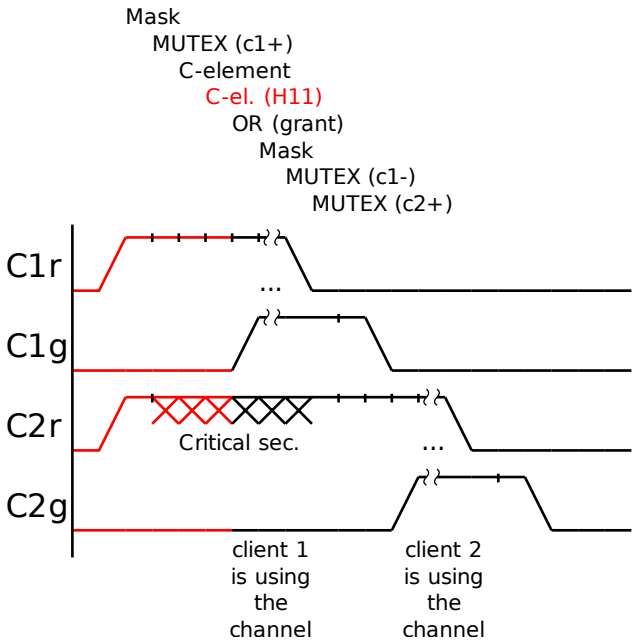
Timing estimation



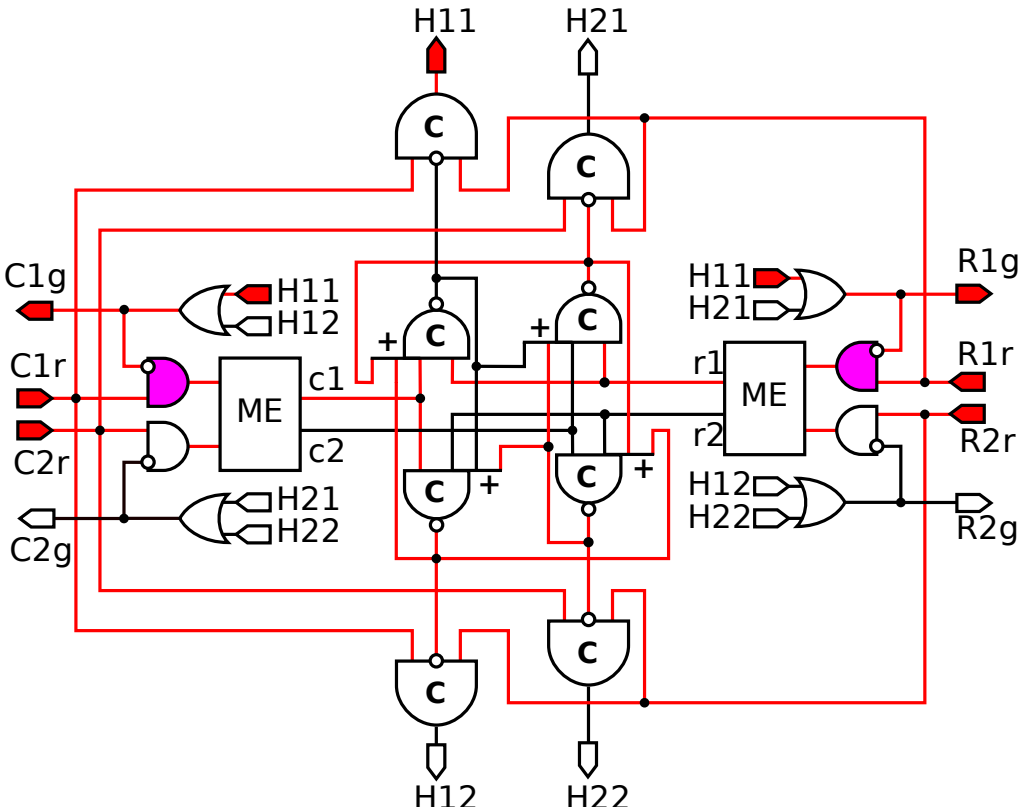


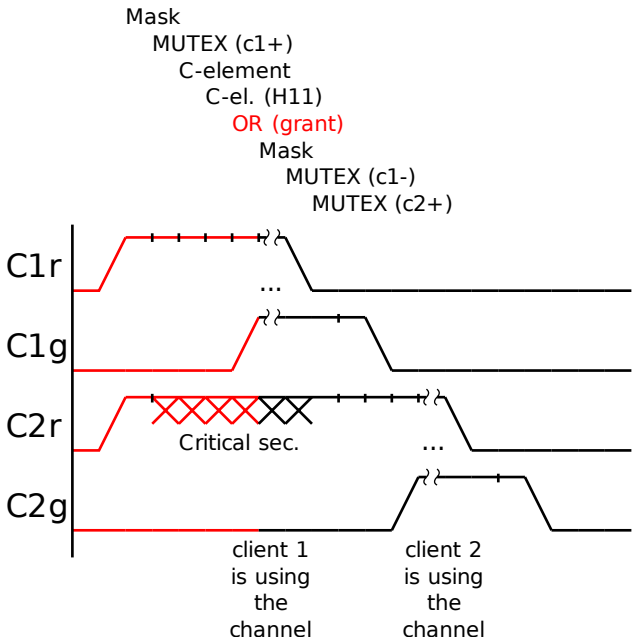
Timing estimation



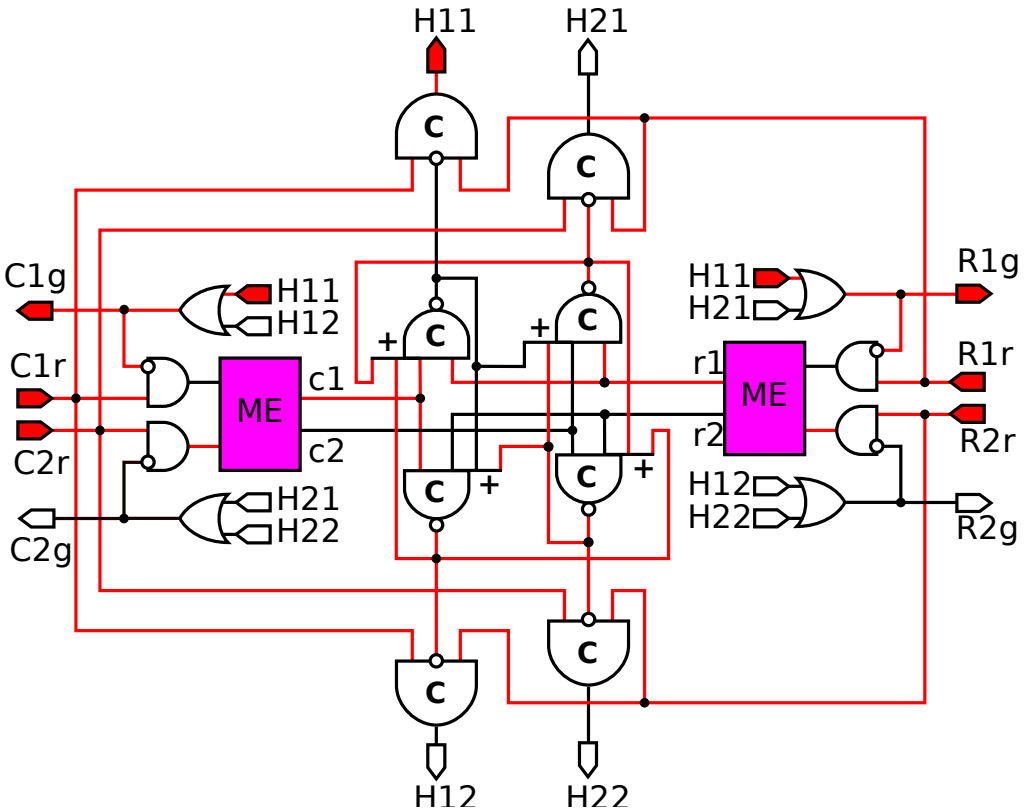


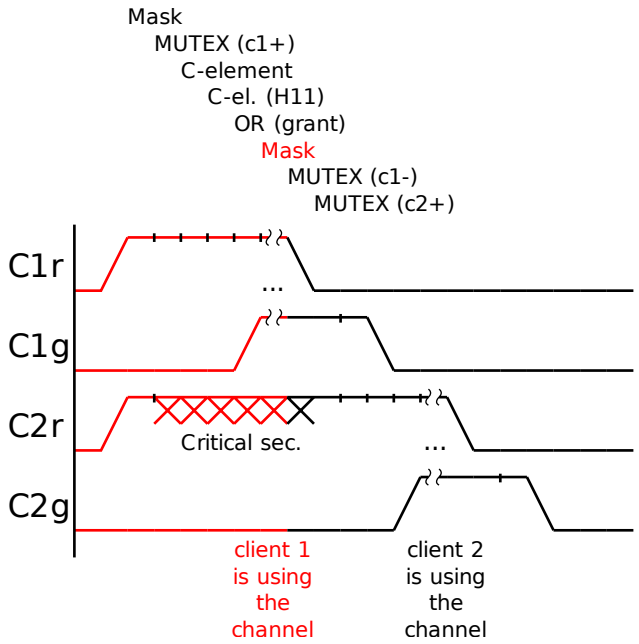
Timing estimation



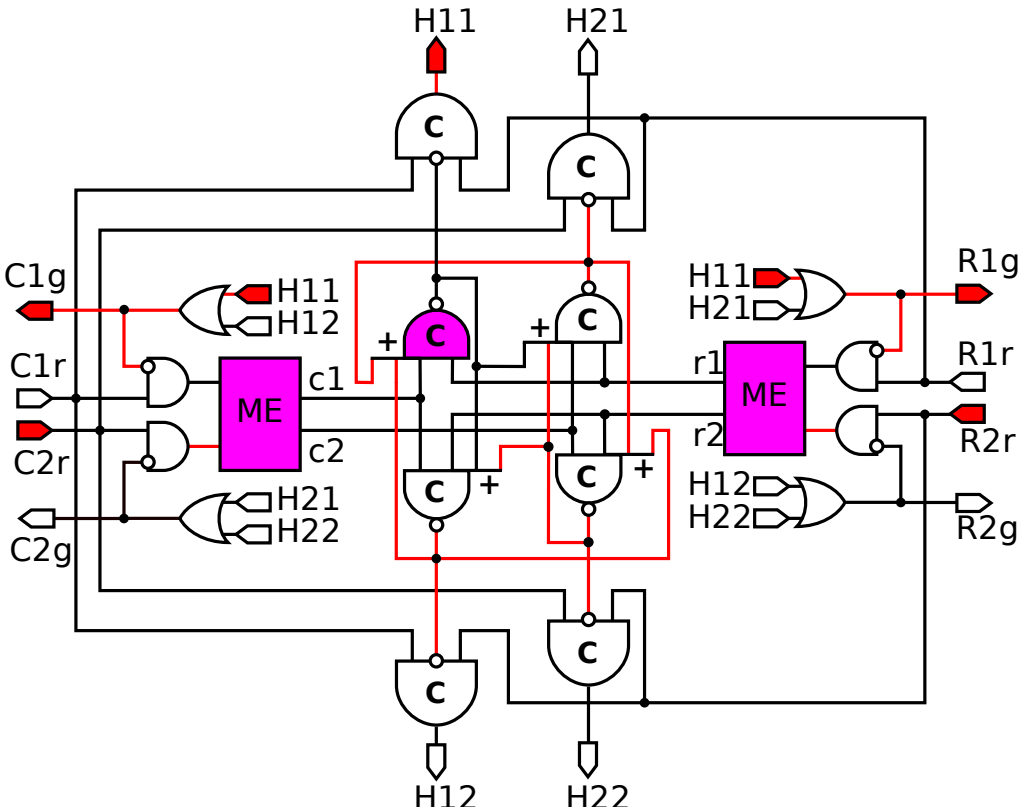


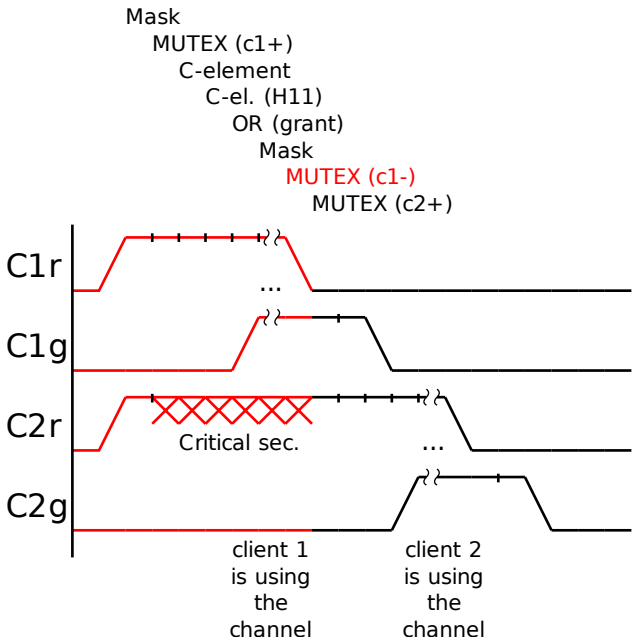
Timing estimation



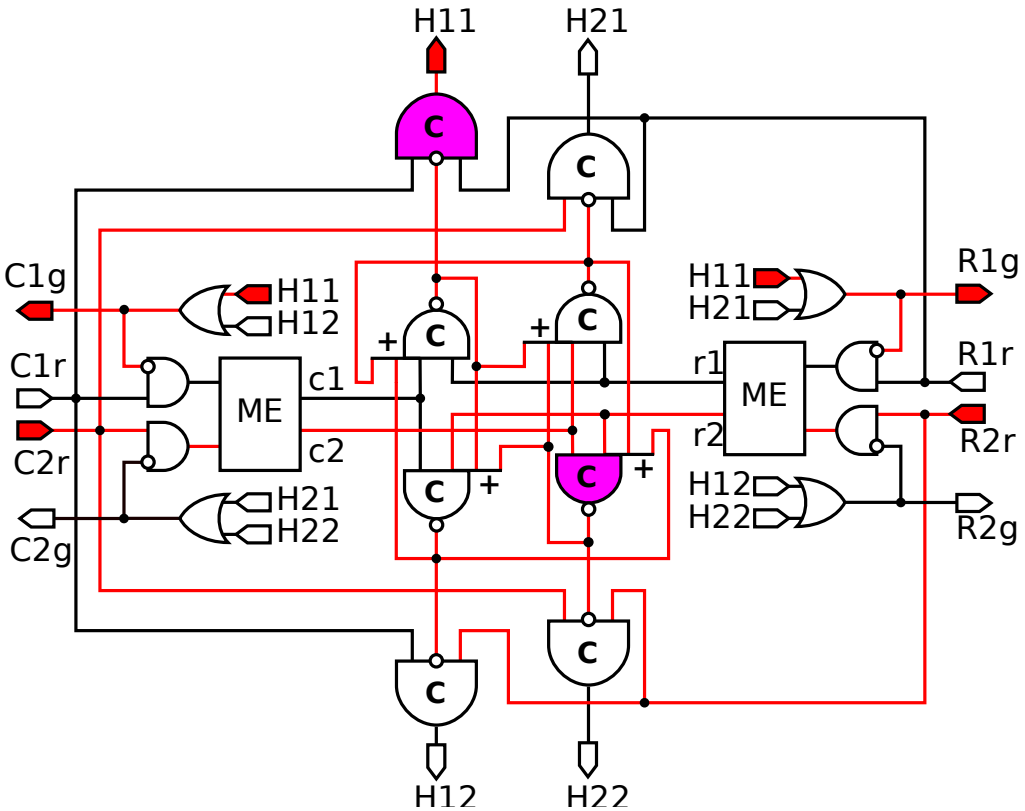


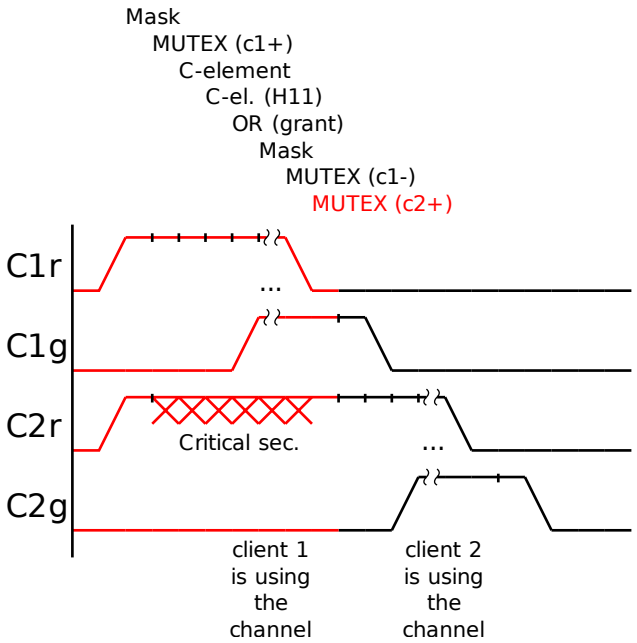
Timing estimation



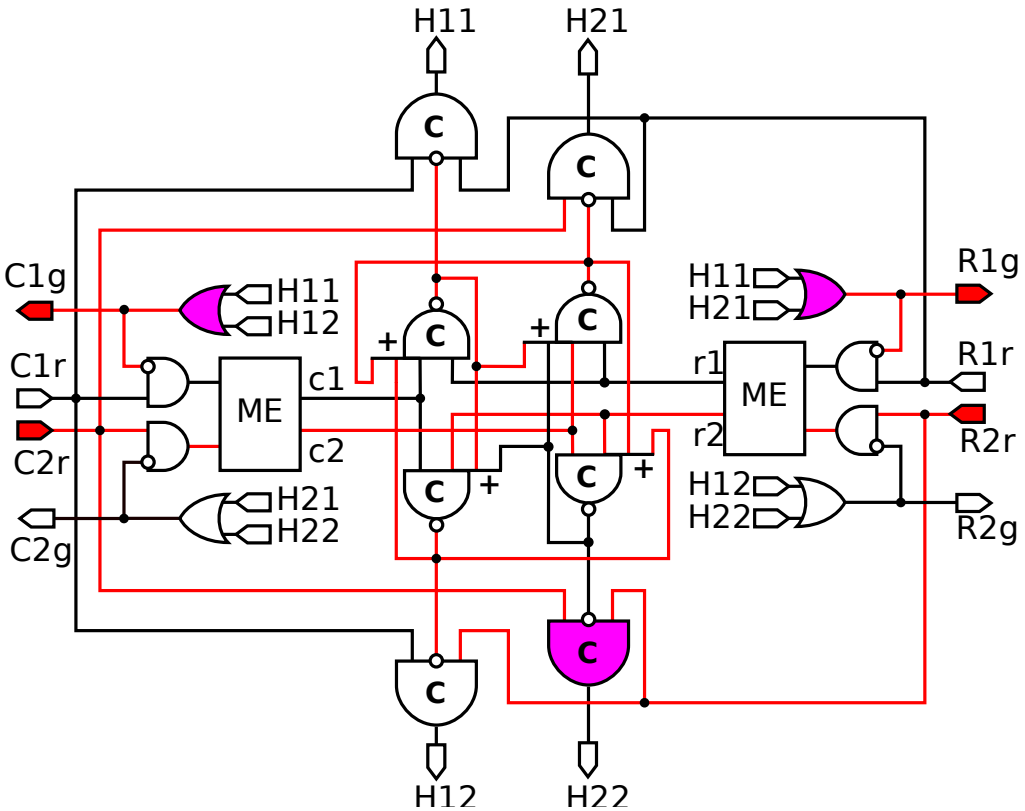


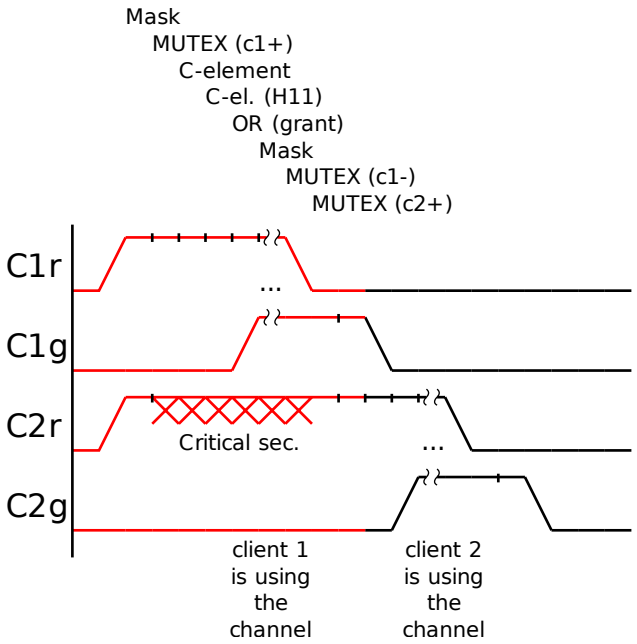
Timing estimation



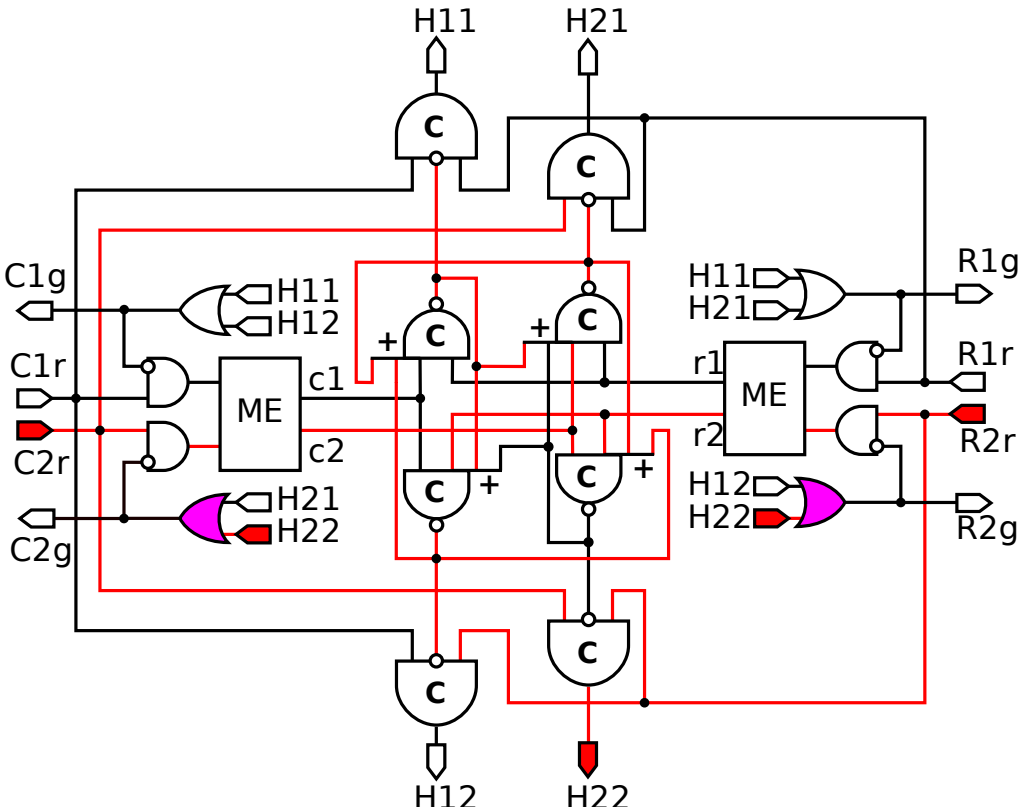


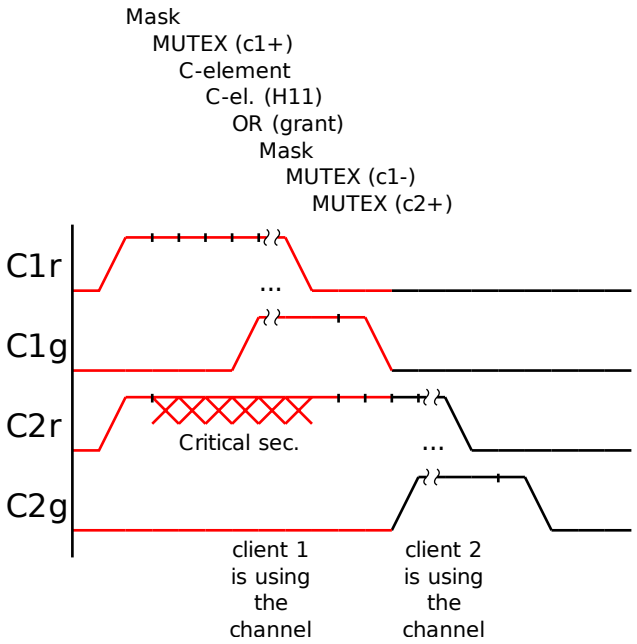
Timing estimation



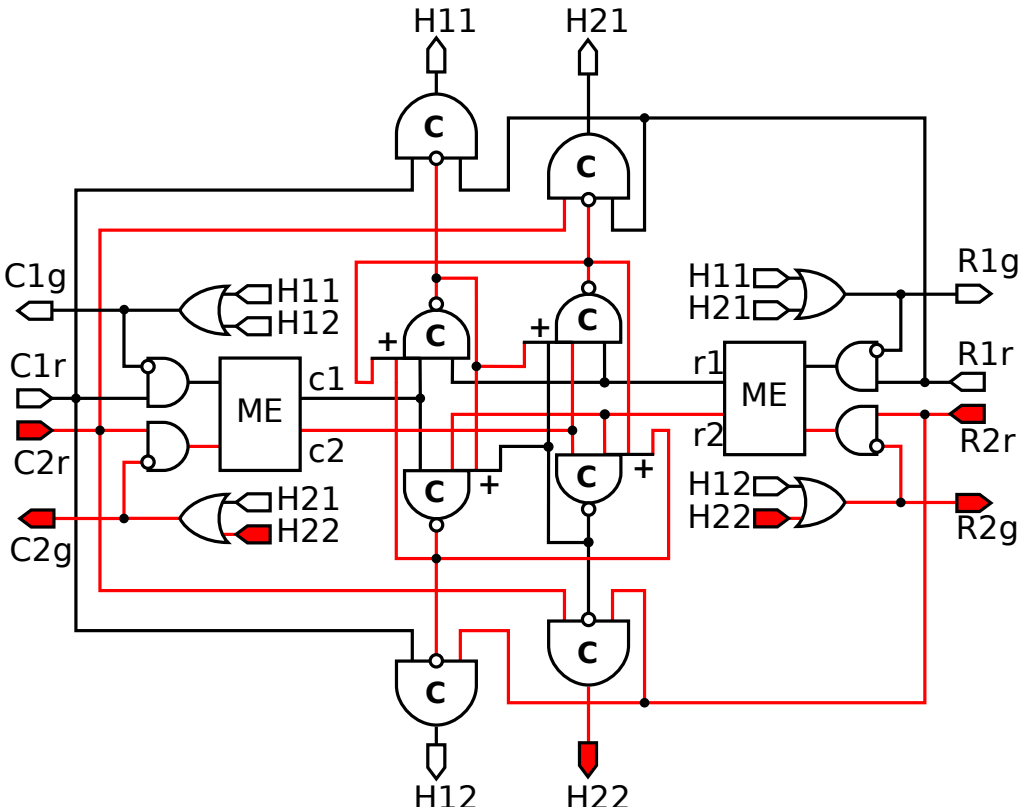


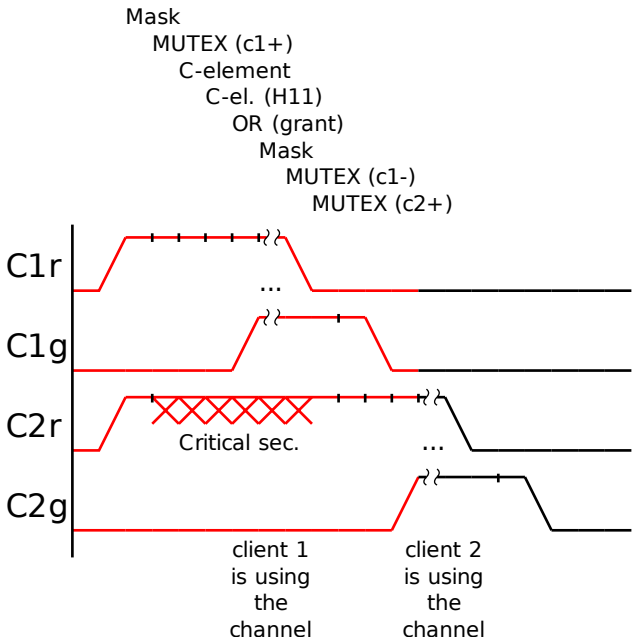
Timing estimation





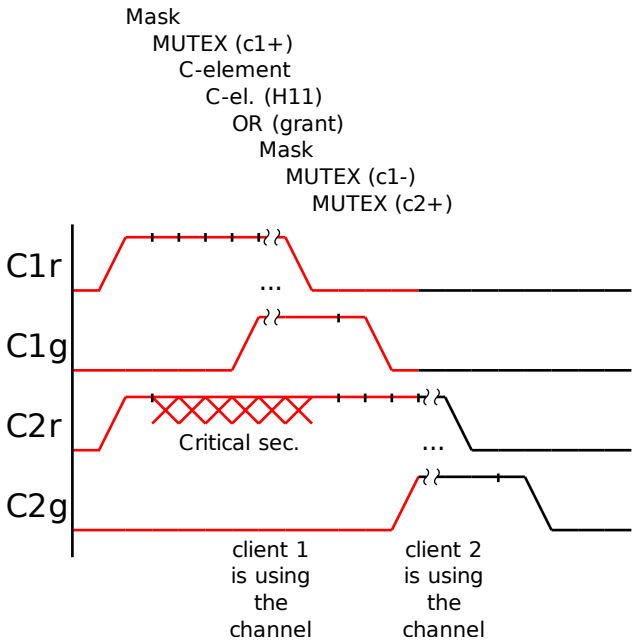
Timing estimation





Timing estimation

- 5 to 11 gates delays, with the critical section of 6 gate delays



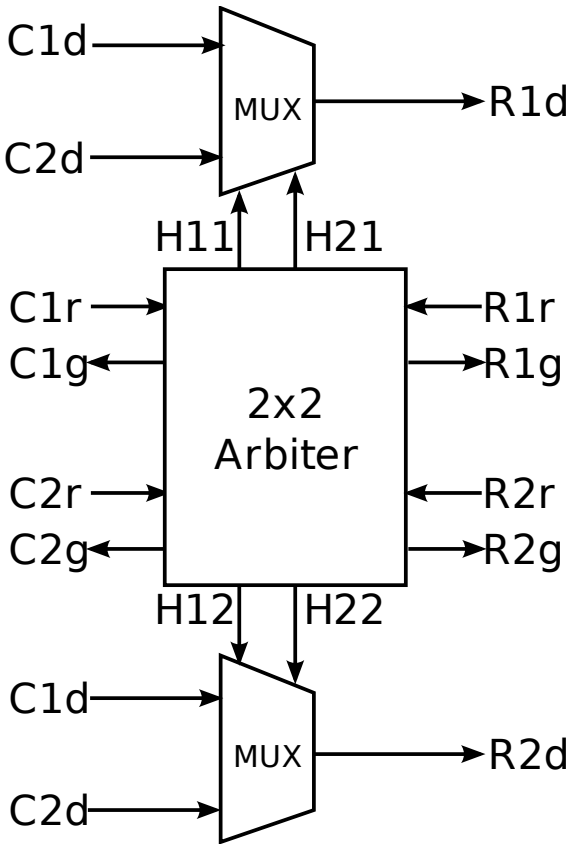
Proving to be Speed-Independent

- The arbiter was verified using the approach presented in [ASYNC'08]:
  - Gate Level Model
  - Generate circuit Petri Net
  - Composition with the environment STG
  - Unfolding of the obtained Petri net
  - Reachibility analisys on the obtained unfolding

Newcastle University

Example of usage 1

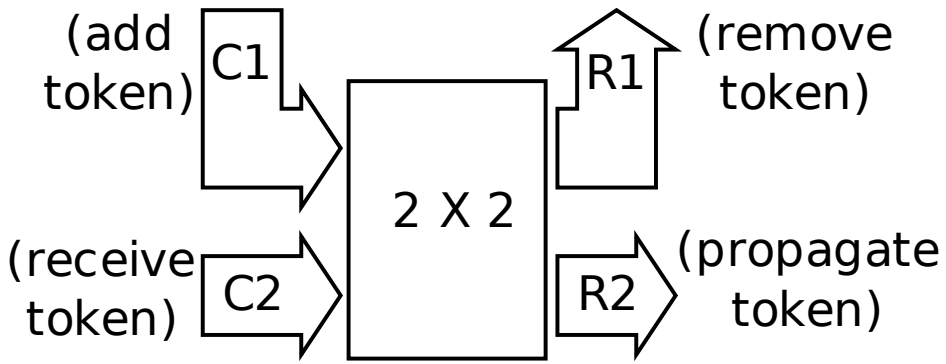
Data transportation



- Handshakes form dual rail signal
- Multiplexers select corresponding channel for data transport from clients to resources
- Useful for data producer/consumer interactions, for instance, processors executing instructions

Example of usage 2

Multi-token arbiter



- Forms busy token ring
- Clients add and remove tokens
- Needs design modification to disable $C1 \rightarrow R1$

Extending the design

- To extend the design to $N \times M$ inputs it is necessary to use N and M input arbiters
- More complicated structure of channel controller and grant controller
- The arbiter will have $N \times M$ different channel outputs
- Problem: Each additional client or resource will linear increase the response time, hence, is only applicable for small increase of inputs
- Solve the problem using approach similar to Priority Arbiters?

Newcastle University

Conclusions

- The SI implementation for 2×2 arbiter case is found
- Arbiter can be useful to balance load between data producers and consumers. Alternatively, can form multi-token arbiter ring structure
- The effective $N \times M$ arbiter implementation still needs to be found

Newcastle University

Bibliography


David J. Kinniment.
Synchronization and Arbitration in Digital Systems.
John Wiley & Sons, Ltd, 2007.


Ivan Poliakov, Andrey Mokhov, Ashur Rafiev, Danil Sokolov, and Alex Yakovlev.
Automated verification of asynchronous circuits using circuit petri nets.
Asynchronous Circuits and Systems, 2008. ASYNC '08. 14th IEEE International Symposium on, pages 161–170, April 2008.

Newcastle University

Thank you!

Questions?