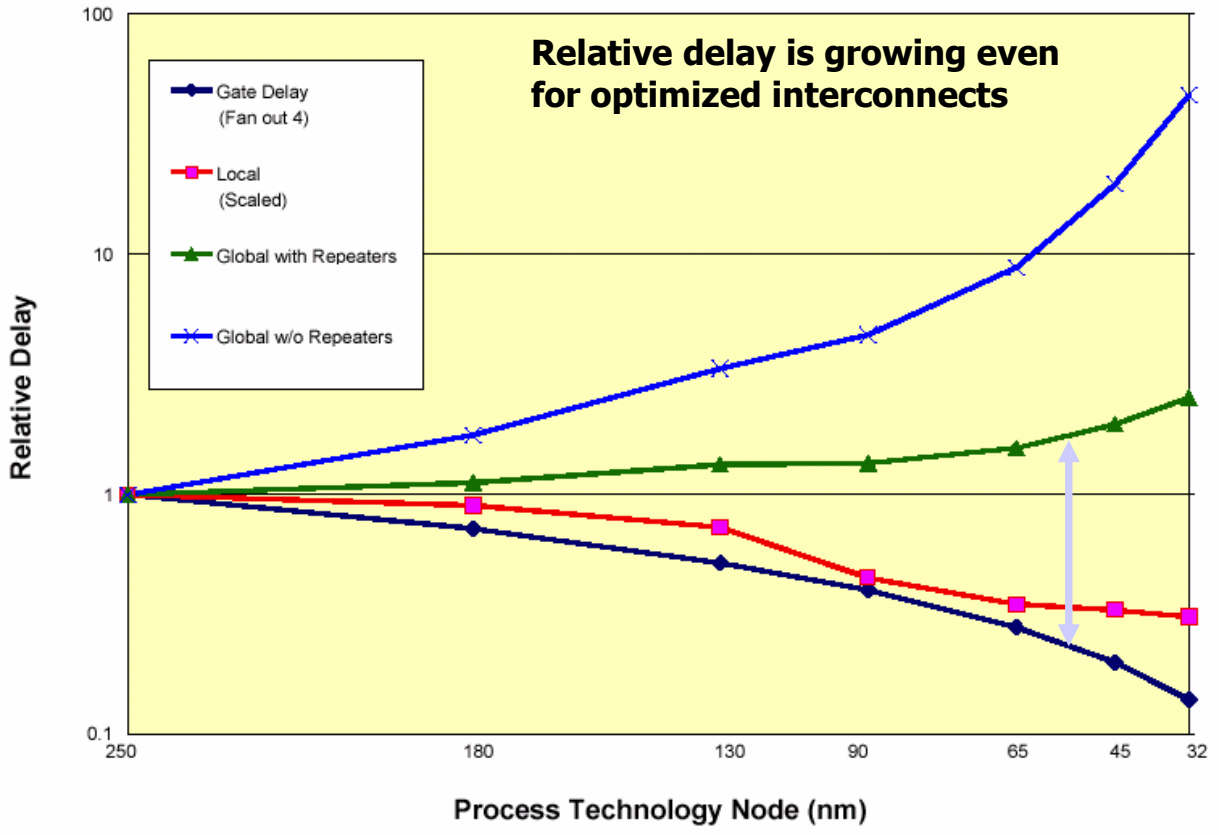
ARTIST Workshop at DATE'06
W4: "Design Issues in Distributed, Communication-Centric Systems"

Information Society Technologies

Communication Issues on MPSoC Platforms: Performance, Power and Predictability

Luca Benini:ARTIST2 Activity Leader
DEIS – Università di Bologna

On-chip interconnect delay trend



Wire delay vs. logic delay

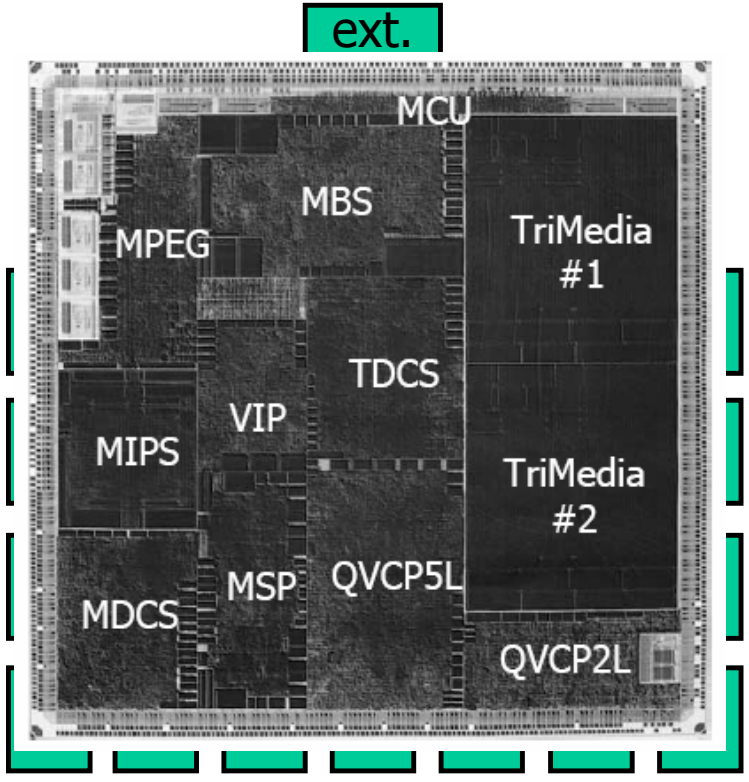
Operation	Delay (0.13um)	(0.05um)
32b ALU Operation	650ps	250ps
32b Register Read	325ps	125ps
Read 32b from 8KB RAM	780ps	300ps
Transfer 32b across chip (10mm)	1400ps	2300ps
Transfer 32b across chip (20mm)	2800ps	4600ps

2:1 global on-chip comm to operation delay
9:1 in 2010

Taken from W.J. Dally presentation: Computer architecture is all about interconnect (it is now and it will be more so in 2010) HPCA Panel February 4, 2002

Unpredicatability & on-chip communication

- Global behavior
  - depends on a set of interacting local behaviors
  - communication is key for interaction
- Unpredictable communication
  - ➔unpredictable global behavior
  - EVEN IF local behavior is predictable



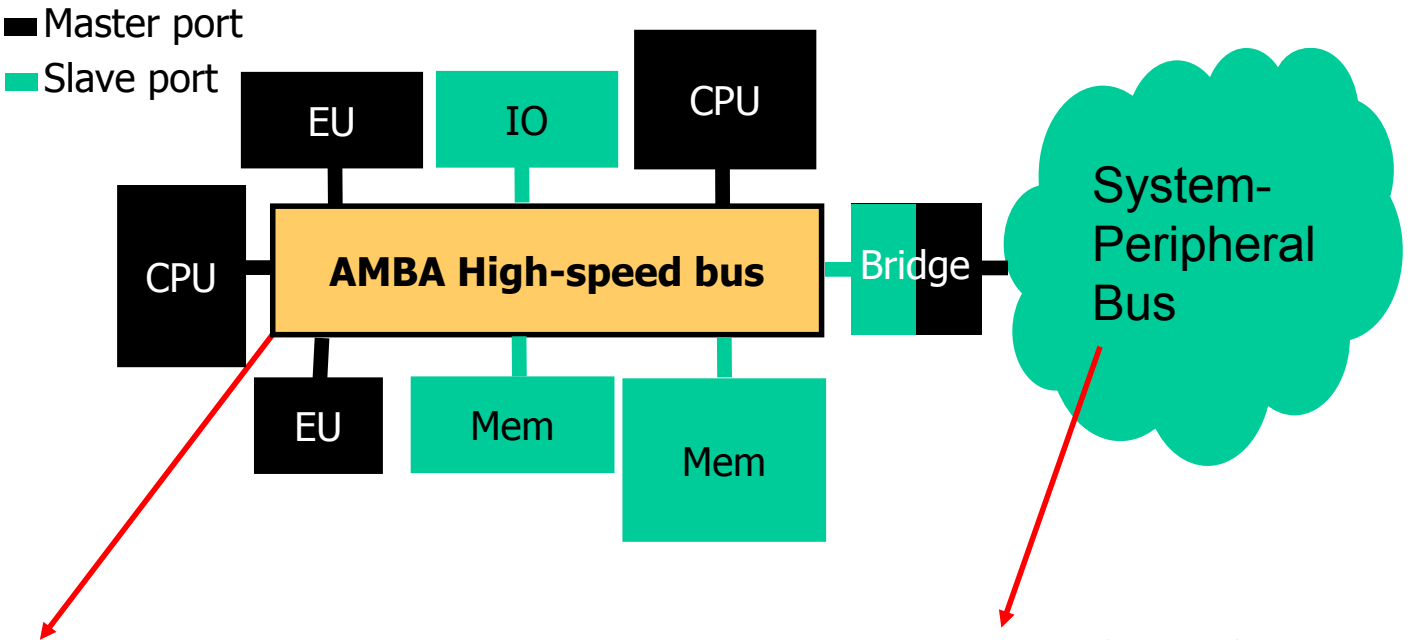
Communication predictability is critical even in a NON-HRT context

On-chip bus Architecture

- Many alternatives
  - Large semiconductor firms (e.g. IBM Coreconnect, STMicro STBus)
  - Core vendors (e.g. ARM AMBA)
  - Interconnect IP vendors (e.g. SiliconBackplane)
- Same topology, different protocols
- Shared medium → contention → predictability losses

How do we support predictability in high-performance on-chip busses?

AMBA bus

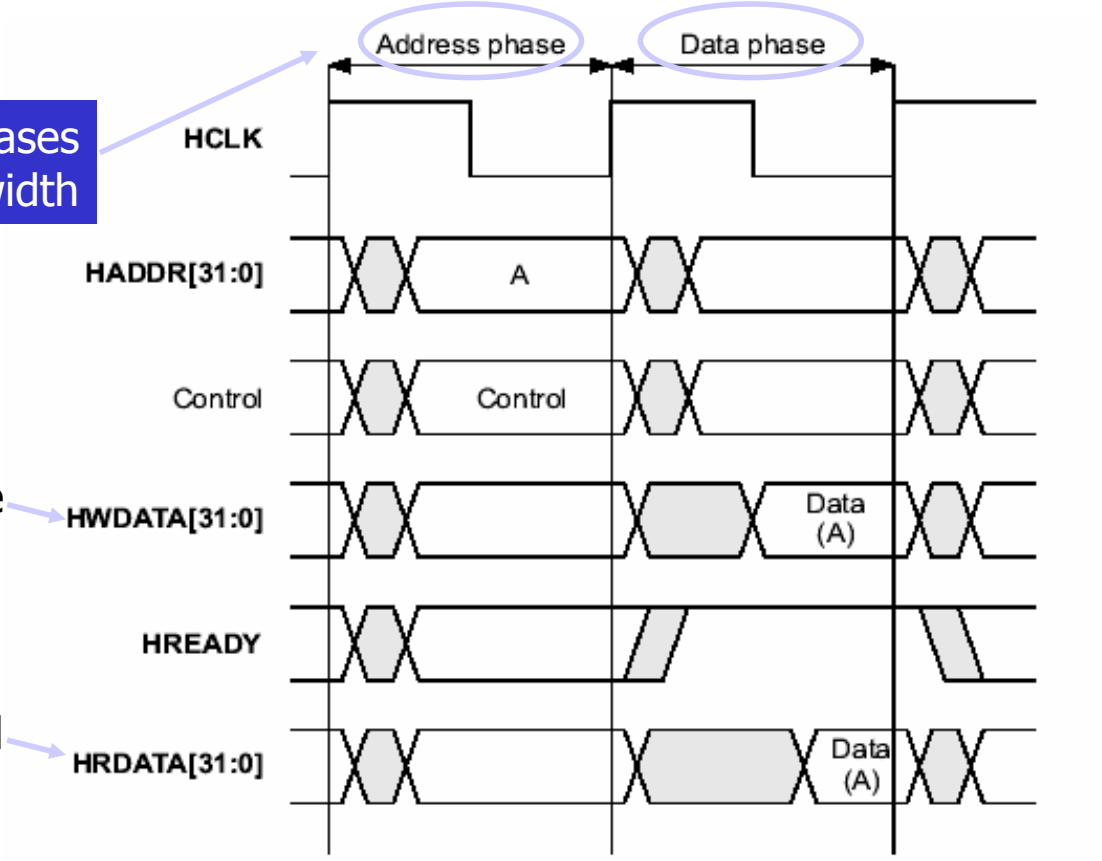


AHB: *high-speed high-bandwidth multi-master bus*

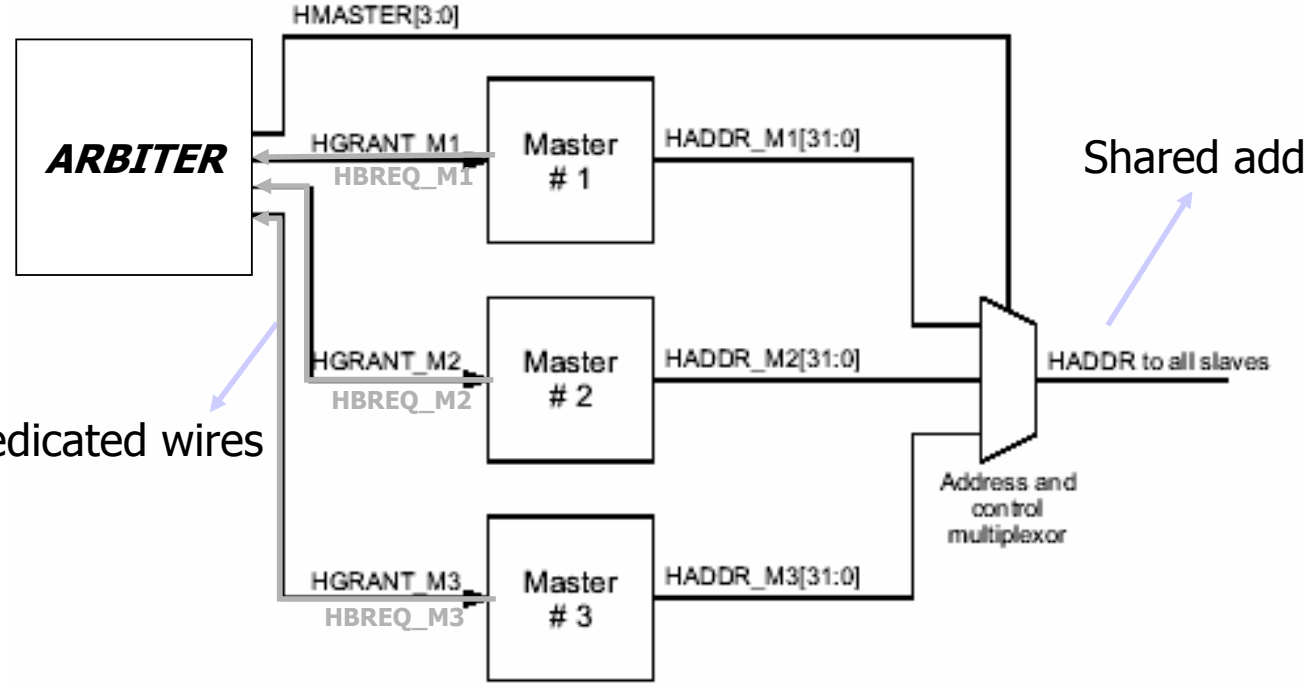
APB: *Simplified processor for general purpose peripherals*

AMBA basic transfer

Pipelining increases Bus bandwidth



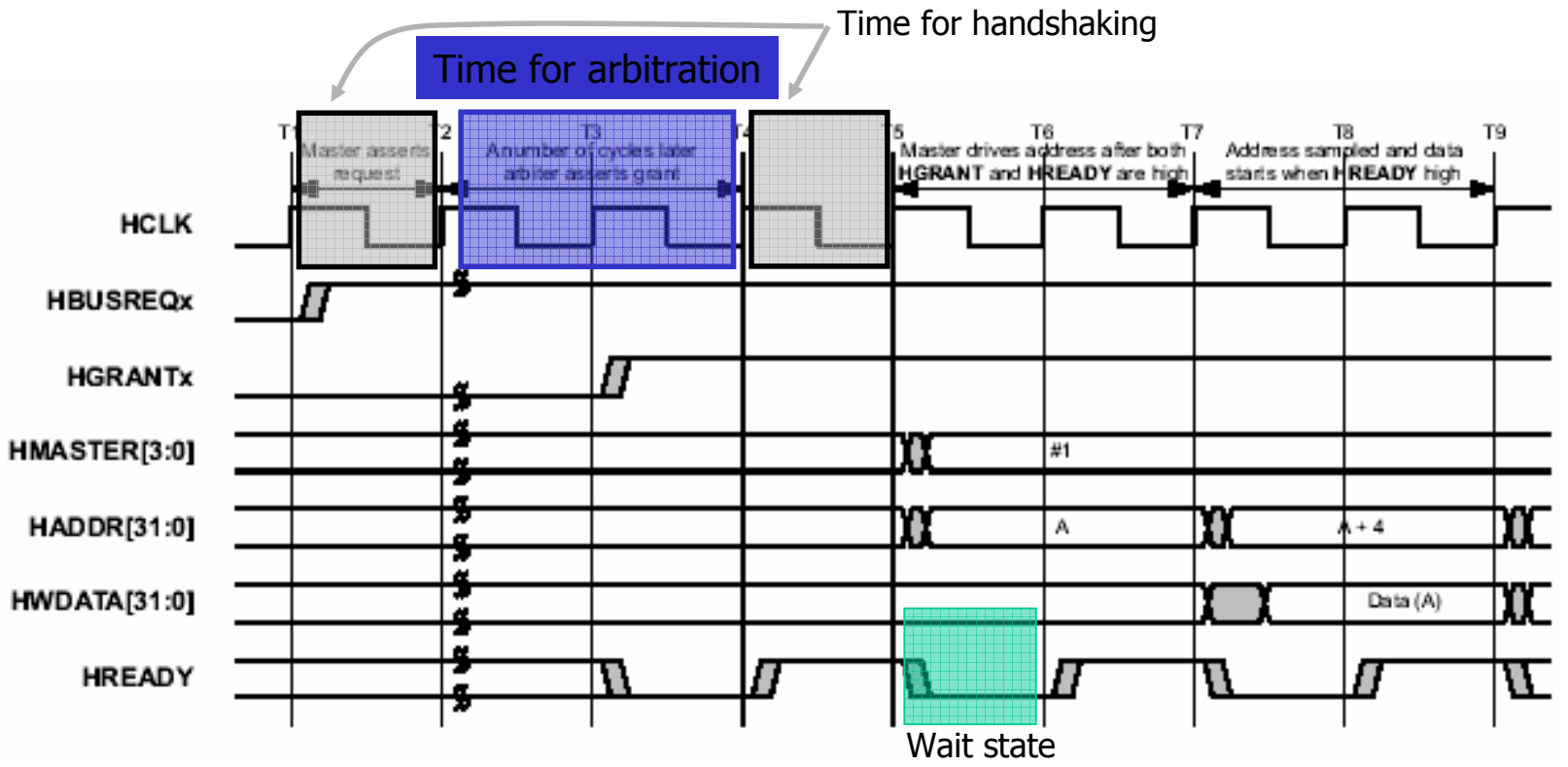
Bus arbitraton



Arbitration Protocol is defined, but Arbitration Policy is not

Priority based protocols are the most common (round robin if priorities are not set)

The price for arbitration



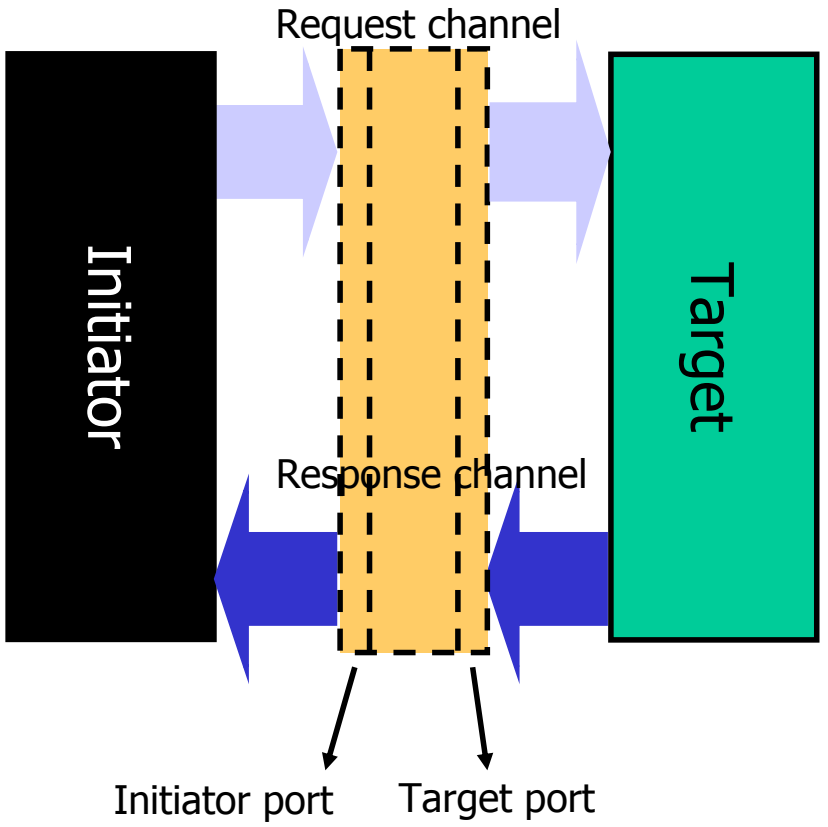
Critical analysis: bottlenecks

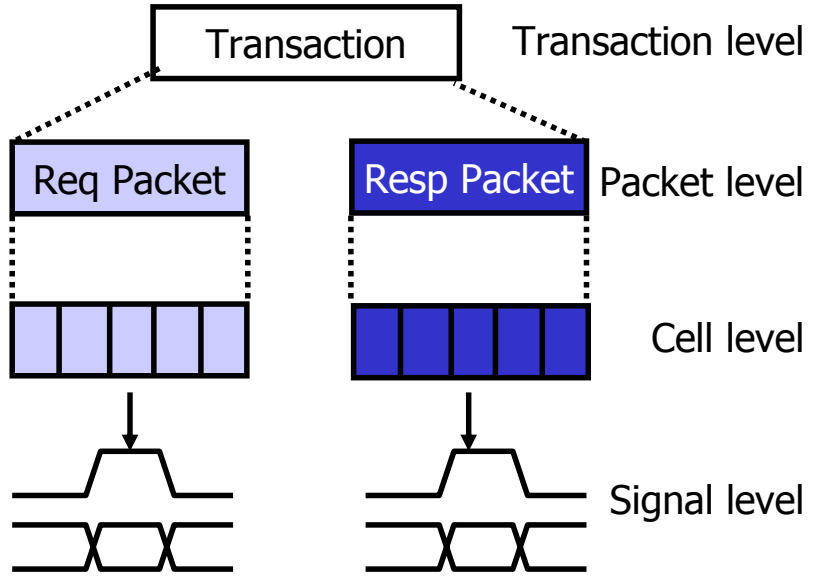
- Protocol
  - Lacks parallelism
    - *In order completion*
    - *No multiple outstanding transactions: cannot hide slave wait states*
  - High arbitration overhead (on single-transfers)
  - *Bus-centric* vs. *transaction-centric*
    - *Initiators and targets are exposed to bus architecture (e.g. arbiter)*
- Predictability losses come from
  - Arbitration
  - Blocking behavior (in-order completion)

STBUS

- On-chip interconnect solution by ST
  - Level 1-3: increasing complexity (and performance)
- Features
  - Higher parallelism: 2 channels (M-S and S-M)
  - Multiple outstanding transactions with out-of order completion
  - Supports deep pipelining
  - Supports Packets (request and response) for multiple data transfers
  - Support for protection, caches, locking
- Deployed in a number of large-scale SoCs in STM

STBUS Protocol (Type 3)





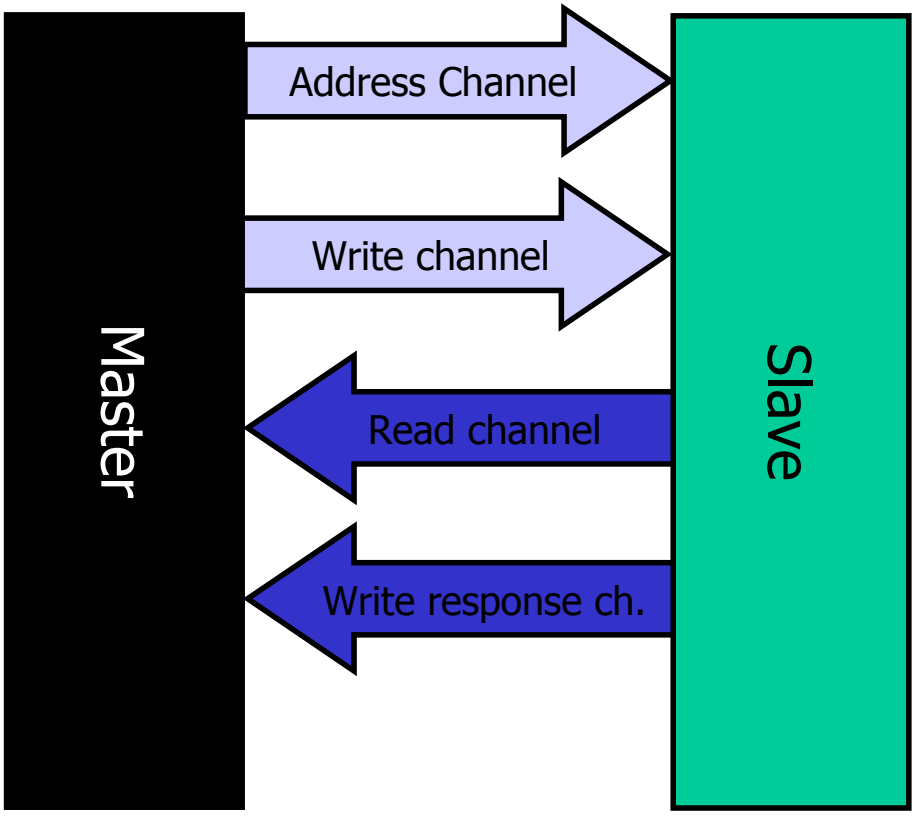
STBUS bottlenecks

- Protocol is not fully *transaction-centric*
  - Cannot connect initiator to target (e.g. initiator does not have control flow on the response channel)
- Packets are atomic on the interconnect
  - Cannot initiate/receive multiple packets at the same time
  - Large data transfers may starve other initiators
- Predictability losses
  - Caused by congestion and by atomic blocking

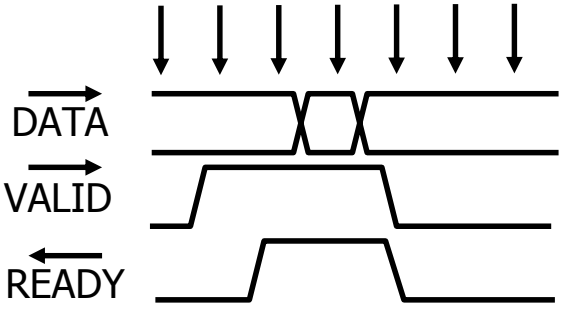
AMBA AXI

- Latest (2003) evolution of AMBA
  - Advanced eXtensible Interface
- Features
  - Fully *transaction centric*: can connect M to S with nothing in between
  - Higher parallelism: multiple channels
  - Supports bus-based power management
  - Support for protection, caches, locking
- Deployment
  - Becoming widespread

Multi-channel M-S interface



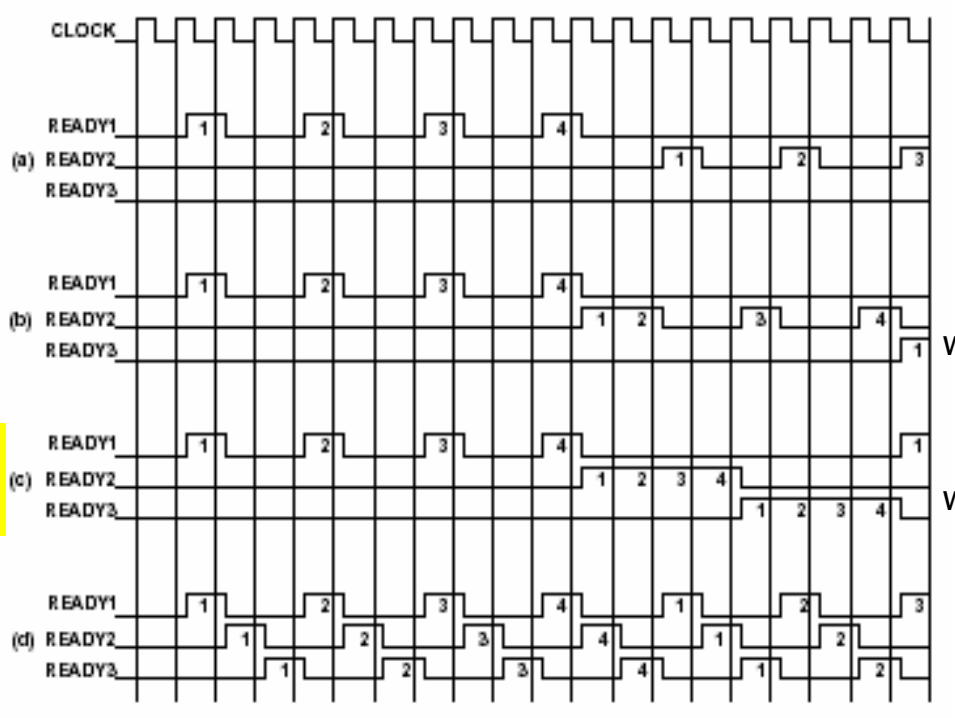
Channel hanshaking



4 parallel channels are available!

Protocol scalability

2 wait states memory



AHB

Cannot hide arbitration and slave response latency

STBUS low buf

One new request processed while a response is in progress

STBUS high buf

More requests processed while a response is in progress

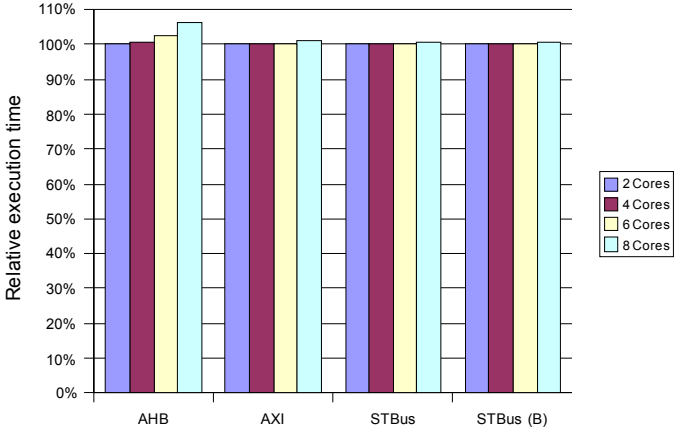
AXI

Interleaving of transfers on the internal data lanes

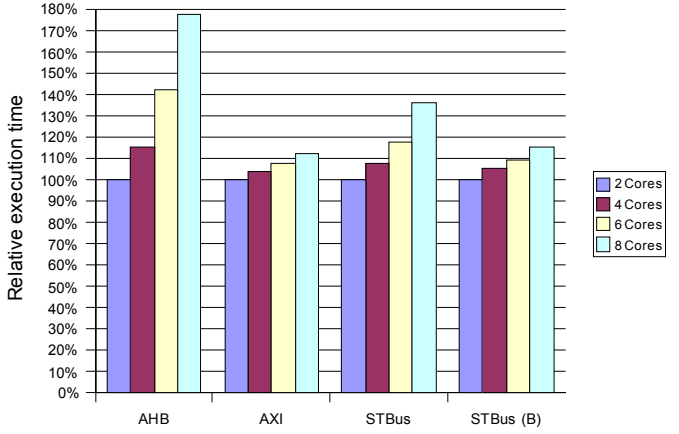
More avanced protocols have less blocking…

Execution time analysis

- Advanced protocols provide better predictability
- But: more area, higher latency, higher power consumption

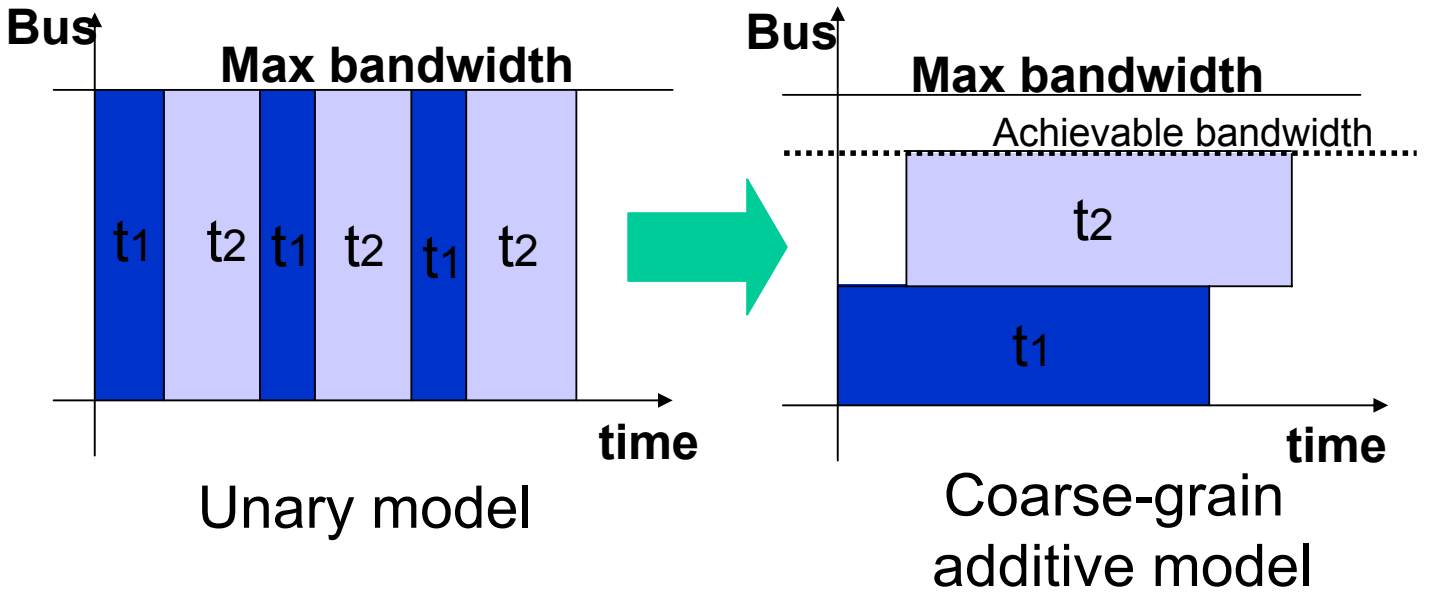


1 kB cache (low bus traffic)



256 B cache (high bus traffic)

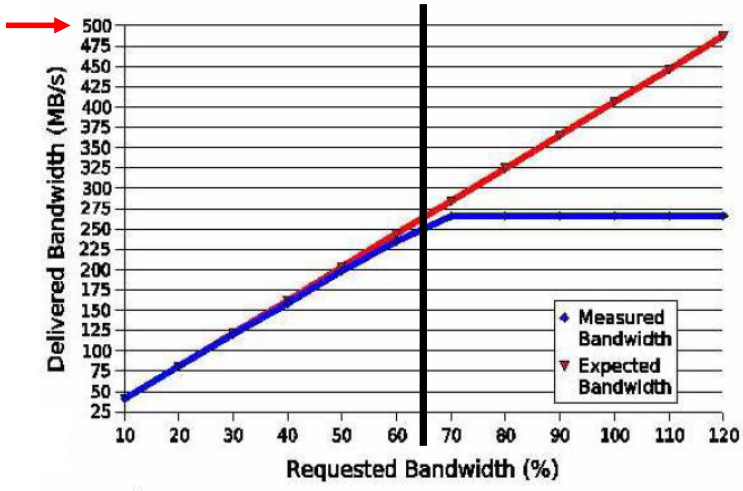
The additive model

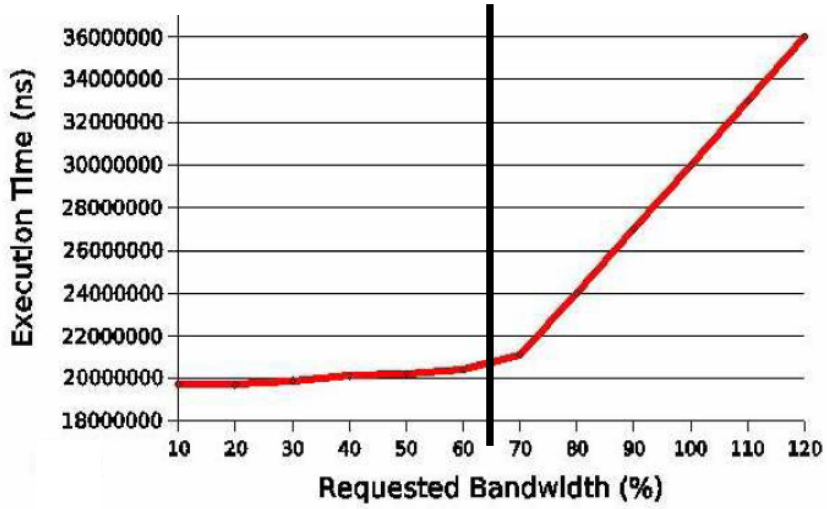


- Characterize the range of applicability of the model
- Force the system to work under the additive regime.

Tuning of the additive model

Theoretical max BW

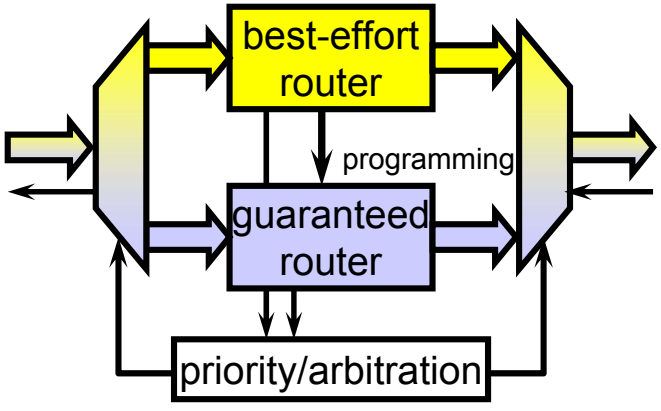




- Requesting more than TH% of the theoretical maximum bandwidth causes the additive model to fail;
  - Generalized to a flow-based model for multi-hop networks
- Benefits of working in additive regime:
  - Task execution time almost independent of bus utilization;
  - Performance predictability greatly enhanced.
- Works well but imposes under-utilization
- Breaks down for long, non-interruptible data bursts
- No deterministic guarantees on single transactions

Alternative approach: static slot allocation

- Increase utilization – guaranteed delivery latency
- **Case study: Æthereal multi-hop interconnect (NoC)**
  - Conceptually, two disjoint networks
    - ✓ *a network with slotted time contention-free arbitration (guaranteed services)*
    - ✓ *a network with contention and buffering (best effort)*
  - ✓ Several types of commitment in the network
  - combine guaranteed worst-case behaviour with good average resource usage

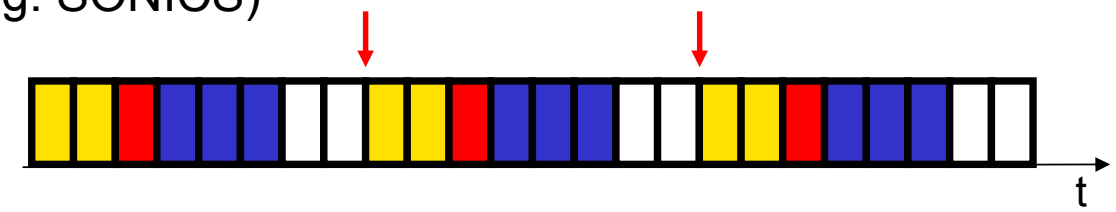


Router architecture

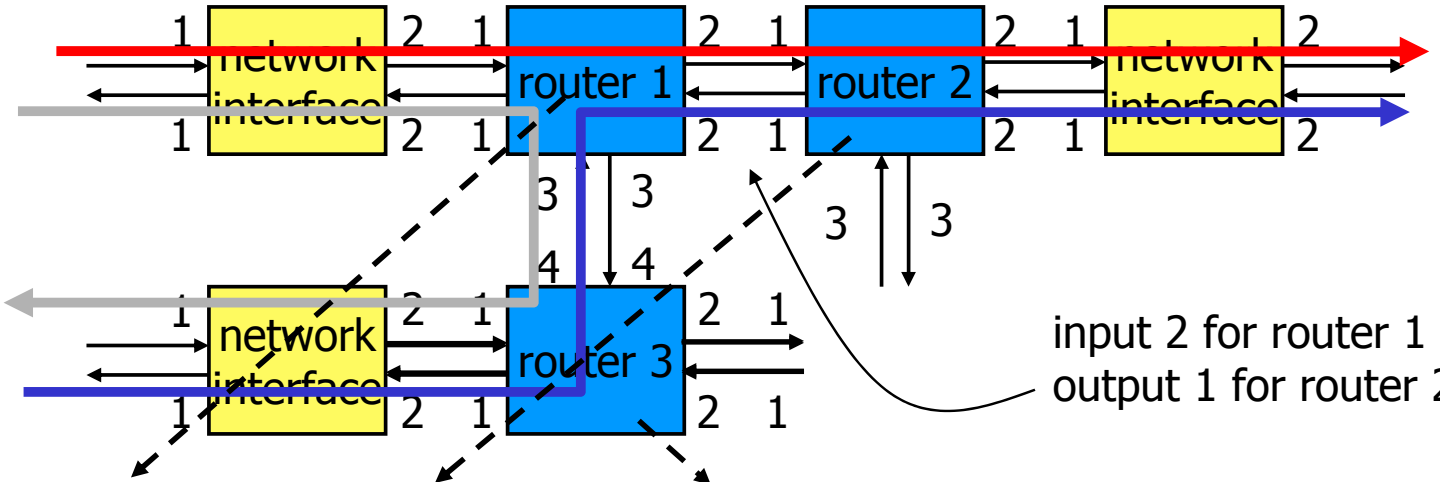
- Best-effort router
  - Worm-hole routing
  - Input queueing
  - Source routing
- Guaranteed throughput router
  - Contention-free routing
    - *synchronous, using slot tables*
    - *time-division multiplexed circuits*
  - Store-and-forward routing
  - Headerless packets
    - *information is present in slot table*

Contention-free routing

- Latency guarantees are easy in circuit switching
- Emulate circuits with packet switching
- Schedule packet injection in network
  such that they never contend for same link at same time
  - in space: disjoint paths
  - in time: time-division multiplexing
  - or a combination
- In a basic, shared bus context: static slot allocation protocol (e.g. SONICS)



CFR by Example



o1	o2	o3	o4
-	i1	-	-
-	i3	-	i1
-	-	-	-
-	-	-	-

o1	o2	o3	o4
-	-	-	-
-	i1	-	-
-	i1	-	-
-	-	-	-

o1	o2	o3	o4
-	-	-	i1
-	-	-	-
i4	-	-	-
-	-	-	-

the input routed to the output at this slot

Use slots to

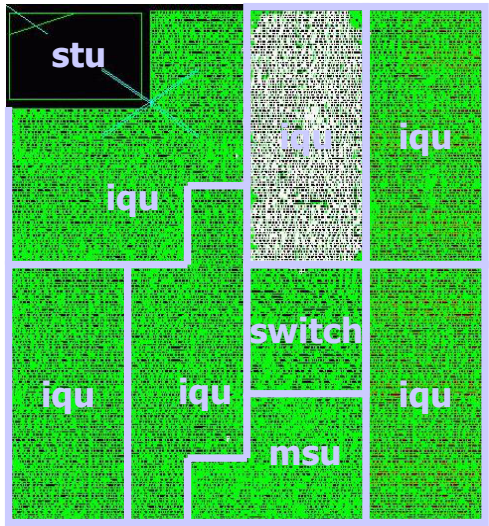
- avoid contention
- divide up bandwidth

CFR setup

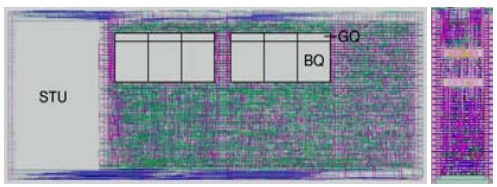
- Use best-effort packets to set up connections
  - set-up & tear-down packets like in ATM
- Distributed, concurrent, pipelined
- Safe: always consistent
- Compute slot assignment compile time, run time, or combination
- Connection opening is guaranteed to complete (but without a latency guarantee) with commitment or rejection

Router implementation

- Memories (for packet storage)
  - Register-based FIFOs are expensive
  - RAM-based FIFOs are as expensive
    80% of router is memory
  - Special hardware FIFOs are very useful
    20% of router is memory
- Speed of memories
  - registers are fast enough
  - RAMs may be too slow
  - Hardware FIFOs are fast enough



routers based on
register-file and hardware fifos
drawn to approximately
same scale (1mm2, 0.26mm2)



Critical Analysis

- Latency of delivery is guaranteed
  - But the upper bound in insertion waiting time is loose
- Can achieve 100% utilization
  - Only if communication requirements do not fluctuate rapidly
- Re-configuration of scheduling tables is long and expensive
  - Can do it only once in a while
- Hardware (area and power) overhead is significant
  - Larger switches and slot tables

Conclusion and some thoughts

- Advanced dynamic arbitration protocols enhance predictabiliy
  - But they add significant hardware complexity
  - Price for under-utilization
- Static slot allocation protocols give deliver latency and bandwidth guarantees
  - But they may increase average latency
  - And they have very significant HW cost
- Simple protocols with many physical channels (low utilization per channel) may represent a good alternative
- All break down with large atomic data transfers
  - These require explicit holistic scheduling approaches
- Can we combine bursty and fragmented traffic?
  - Yes, but it's not easy to analyze these systems