Asynchronous vs. Synchronous Design Techniques for NoCs

Robert Mullins

Aims of Tutorial

- Highlight the wide range of system timing alternatives for NoCs
- Discuss the impact of the choice of timing regime on the architecture of NoC routers
- Contrast different approaches
Synchronous to Delay-Insensitive Approaches to System Timing

Timing Assumptions

Synchronous to Delay-Insensitive

- Local clocks/interaction with data (becoming aperiodic)
- Less detection

Global, Sub-System, Local, Local Relative, Wire Delay, Isochronic Forks, None

Multiple clocks, Pausable clocks and locally triggered clock pulses, Bundled Data, Quasi-Delay Insensitive
System Timing

• Approaches to system timing are distinguished by what delay assumptions they make.

• A number of different approaches to system timing may also be combined:
  – Globally-Asynchronous Locally-Synchronous (GALS)
    • e.g. Synchronous IP interconnected by an asynchronous network.
Synchronous On-Chip Networks
Generic On-Chip Router
Synchronous Router Pipeline

- Router Pipeline may be many stages
  - Increases communication latency
  - Can make packet buffers less effective
  - Incurs pipelining overheads
Speculative Router Architecture

- VC and switch allocation may be performed concurrently:
  - Speculate that waiting packets will be successful in acquiring a VC
  - Prioritize non-speculative requests over speculative ones

Single Cycle Speculative Router

- Single cycle router made possible by use of speculation
- Clock period is almost unchanged (compared to pipelined design)
  - Approx. 30 FO4 (simple standard-cell design)
- Presence of clock simplifies design
  - Arbitration
    - Fast combinational matrix arbiters
    - Can easily be extended to handle priority traffic etc.
  - Speculation
    - Aided by the clear notion of a clock “cycle”
    - Simple abort logic (abort detection and actual abort)
Single Cycle Speculative Router

- 4x4 mesh network, 25mm²
- Single Cycle Routers (router + link = 1 clock)
  - Low common case latency
- 4 virtual-channels/input
- 80-bit links
  - 64-bit data + 16-bit control
- 250MHz (worst-case PVT)
  16Gb/s/channel, 0.18um.

R. D. Mullins, A. West and S. W. Moore, "The design and implementation of a low-latency on-chip network", In Proceedings ASP-DAC’06
Beyond a Single Global Clock
Limitations of Fully-Synchronous Networks

1. Difficult to distribute clock
   - Network spread over die & may have irregular layout
   - Minimising skew costs complexity and power

   • Alternatives/extensions to PLL and H-tree:
     - Clock deskewing techniques
     - Distributed Clock Generator (DCG).
     - Distributed PLLs
     - Standing-wave oscillators and rotary clock schemes
     - Resonant global clocks, optical clock distribution etc.
Limitations of Fully-Synchronous Networks

2. Single Network Clock Frequency

– Communicating synchronous IP blocks may operate at different and potentially adaptive clock frequencies

– What is most appropriate network clock frequency?

  • We don’t want to have to generate and distribute a very high frequency clock in order to emulate an asynchronous network
Frequency Distribution

- Clock skew may force the system to be partitioned into multiple clock domains.
- Can exploit the fact that only the phase of each router’s clock differs, simple error-free clock-domain crossing possible (single clock source).
Router clocks derived from a single source

- Each router’s clock may be generated from the global network clock, either by:
  - Clock division \textit{or}
  - Clock multiplication

- Clock domain crossing techniques can exploit known clock frequency relationships

Chakraborty and M. Greenstreet, “\textit{Efficient Self-Timed Interfaces for Crossing Clock Domains}”, In Proceedings ASYNC’03

L. F. G. Sarmenta, G. A. Pratt and S. A. Ward, “\textit{Rational Clocking}”, ICCD’95
Locally Generated Clocks (periodic & free-running)

- Can exploit knowledge about clocks (when crossing clock domains) even if all we know is that they are periodic, examples:
  - predictive synchronizers [Dally][Frank/Ginosar]
  - asynchronous FIFOs [Chakraborty/Greenstreet]
Synchronous Routers with Asynchronous Links

- Synchronization:
  - Time Safe: e.g. Traditional 2 FF synchronizers
  - Value Safe: Clock Pausing/Data-driven clocks
Locally Clocked Routers/Asynchronous Interconnect (GALS style network)

• Can support asynchronous interconnects
  – No longer exploiting periodic nature of router clocks
  – Correct operation is independent of the delay of the link

• GALS interfaces with pausable clocks
  – If necessary clock is stretched, data is always transferred reliably (value safe)
  – Need to construct local delay line
GALS – Clock Pausing

- Simple GALS interface (receiver)
- Note: Req/Ack uses 2-phase handshaking protocol
GALS – Multiple Inputs

- Clock is free running (although it can be paused)
- It is the clock that really determines if asynchronous data is transferred into the synchronous clock domain on a particular cycle
- Impact on performance in on-chip network requiring multiple input data/control ports?
GALS – Stoppable Clock
Local aperiodic clock generation

- Discard free-running clock but retain a single delay assumption for router
- Options for clock pulse generation:
  1. Use stoppable GALS interface and attempt to stop every cycle – overheads?
  2. Wait for data/null-data from all neighbours before generating pulse (global synchrony!)
  3. Data driven clock
  4. Traditional asynchronous bundled-data approach (with a single delay assumption for whole router)
- Can still exploit synchronous router implementation
Data-Driven Local Clock

Idea:

– If data at any input, sample all inputs
– Determine which inputs are to be admitted on next clock cycle (requires MUTEX)
– Ensure data that is not admitted is ‘locked out’ for next clock cycle
– After all MUTEXes have made a decision (and never faster than the delay line!) generate a clock pulse

• Similarities to stoppable GALS interface and asynchronous priority arbiters
Data-Driven Clock Waveform

Valid data triggers decision to be made about all other inputs.
Are they valid or not? admitted or locked out until next cycle.

PORT A
PORT B
PORT C

CLOCK

Valid Data at Input Port
Imagine data from two packets arriving at a single router node at different rates.

An aperiodic clock may be generated to minimise latency and power.

Minimum clock period set by delay line.

Value safe synchronization (no chance data is ever lost).
May be generalized to n-input ports. Only the control interfaces are shown here (r1,a2 and r2,a2)

\(grant_n\) is simply used to control the latching of data at each input port (register enable)
Data-Driven Local Clock

• Simple implementation shown (work in progress)
  – Some small timing constraints
  – Performance tweaks possible

• Possible Extensions
  – Force synchronization on subset of inputs
    • Some inputs must be present for clock to be generated
  – Generate additional clock pulses to handle pipelining
    • Counter & clock driven lock signal
  – Select a different clock period (delay line) depending on which inputs have been granted
    • Data-dependent clock period

See also: M. Krstic and E. Grass, “New GALS Technique for Datapath Architectures”, PATMOS 2003. (and ASYNC’05 paper)
Clocking alternatives for Synchronous Routers

- Assume constant link delay (simple to clock predictive synchronizers)
  - Derived Clocks
    - Independent Router clocks
  - Single Matched Delay
    - Local aperiodic clock generation

SYNCHRONOUS

- Single Clock, Multiple Clock Domains, Source Synchronous Communication
- Independent Router Clocks (time safe sync.)
- Pausable Clocks (value safe sync.)
- Multiple Matched Delays, Async. H/S (Bundled Data or "desynchronized")

No assumptions about link delay
Synchronous Routers - Summary

- Can design high-performance single cycle routers
- Design is simplified by presence of global synchrony
- Distribution of global clock can be eased by:
  - New clock generation/distribution techniques
  - Source synchronous communication
- Network operating frequency
  - Relax global synchrony further
  - Data-driven clocking determines most appropriate router clock frequency automatically
Asynchronous On-Chip Networks
Why are asynchronous NoCs interesting?

- Simple/elegant solution when networked IP blocks run at different clock frequencies
  - Data driven, no superfluous switching activity
  - No synchronization/clock alignment issues at interfaces
- Ability to exploit data/path-dependent delays
  - Low-latency common or high-priority paths through router
- No clock distribution issues
- Security and EMI advantages
  - Clock focuses EM emissions
  - The presence of a clock can also aid fault-induction and side-channel analysis attacks
Why are asynchronous NoCs interesting?

• Freedom to optimize network links
  – Not constrained by need to distribute/generate multiple clock frequencies. Can exploit high-frequency narrow links.
  – Dynamic latency/throughput trade-offs (adaptive pipeline depth)
  – Exploit dynamic optimizations on links (e.g. DVS)

• Reduced design time
  – Easy to use interfaces, modularity.
  – Robust and simple implementation

• Some arguments for reduced power
Asynchronous Circuit Basics

• Control in asynchronous circuits often relies on simple handshaking protocols (req/ack event cycles)

• Delay-insensitive event-driven system - every signal transition is acknowledged

• The C-element is a fundamental building block of many asynchronous circuits
  – Can be thought of as a AND-gate for events

IF inputs match in state
THEN copy it for output
ELSE hold previous state;

if $a=b$ then $y := a$

$y = ab + y(a+b)$

(State Holding Element)
Simple Pipelines

Event FIFO

Micropipeline

Arbitration

Bistable

Metastability Filter

MUTEX

MUTEX

ARBITER

(Arbitrated Call)

Multiway Arbiters

Tree Arbiter

Cascaded MUTEXes
Static Priority Arbiters

• “Priority Arbiters”  
  Bystrov/Kinniment/Yakovlev  
  (ASYNC’00)
• First stage samples/locks current request vector
• Static or dynamic priority
• Original design updated to tackle performance and QoS issues  
  Felicijan/Bainbridge/Furber  
  (ICM’03)
Delay-Insensitive Communication

<table>
<thead>
<tr>
<th>D1 D0</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>clear</td>
</tr>
<tr>
<td>0 1</td>
<td>valid &quot;0&quot;</td>
</tr>
<tr>
<td>1 0</td>
<td>valid &quot;1&quot;</td>
</tr>
<tr>
<td>1 1</td>
<td>error (alarm)</td>
</tr>
</tbody>
</table>

1. REQ+
2. ACK+
3. REQ-
4. ACK-

4-phase dual-rail protocol
Delay-Insensitive Switched Interconnect

- The basic DI latch can be extended to support steering, multiplexing and arbitration

CHAIN

• Basic link is 6 wires
  – 2-bits of data (1-of-4) + end of packet + ack
    • any N-of-M code could be used
  – around 1Gbps (0.18um, 160Mbps per wire)
  – Links may be ganged together

• Route information tapped off and used to steer remainder of packet

• If arbitration is required, arbiter grant is retained for duration of packet (no fragmentation of packets)
Asynchronous on-chip networks

• How do we build more complex on-chip routers?
  – Support for virtual-channels
  – QoS

• Challenges
  – Multi-way & prioritised arbitration
  – Control overheads
    • Arbitration and DI circuits can be slow!
    • How can control overheads be hidden?
Overview of Some Published Asynchronous On-Chip Networks

• “Quality-of-Service (QoS) for Asynchronous On-Chip Networks”
  T. Felicijan (Ph.D. 2004, Manchester)
  http://www.cs.manchester.ac.uk/apt/publications/

• “An Asynchronous Router for Multiple Service Levels Networks on Chip”, R. Dobkin et al, ASYNC’05. (QNoC Group)

• MANGO Clockless Network-on-Chip
  – “A router Architecture for Connection-Orientated Service Guarantees in the MANGO Clockless Network-on-Chip”, T. Bjerregaard and J. Sparsø, DATE’05
Virtual Channels

- **Best Effort Routers**
  - Virtual-Channel allocation is performed at each router
  - any free VC (at the required output) may be assigned to a new packet
    - Significant performance gains over simpler static schemes
  - Can also prioritize packets

- **QoS Routers based on Static VC allocation**
  - Packets retains the same VC throughout the network.
  - Each VC is assigned a static priority level

- **Connection-Orientated Router**
  - VCs are reserved at each router along a path to create a connection
  - Hard QoS guarantees possible
QoS Support

• All these asynchronous networks provide QoS support

• MANGO
  – Guaranteed Service (GS) connections
  – A connection is a reserved sequence of VCs through the network
  – Hard latency and bandwidth guarantees are provided
Static VC assignments

- [Felicijan][Dobkin] implement QoS through static VC assignments
  - i.e. packet is assigned VC and uses this VC at all routers
  - May need to contend with other packets assigned the same VC
  - Packets with same VC cannot be interleaved
  - VC is reserved for duration of packet (reserved rather than allocated from pool of free VCs)
Felicijan/Manchester

- Implementation style:
  - QDI, 1-of-4 encoded data with RTZ signalling
- Simplest switching network of asynchronous designs (multiplexed crossbar)
- 8-bit data flits
- Performance Results (0.18um)
  - Maximum router frequency ~300MHz
  - Minimum router latency ~5ns?
- Two constraints on provision of QoS
  - First due to multiplexed crossbar
  - Second related to minimum buffer requirements
Each output selects from all input VCs (Fully-connect crossbar)

Flits/requests from all VC0 inputs

\[ \text{ARBITER} \]

\[ \text{Grant held for duration of packet} \]

\[ \text{Static Priority Arbiter} \]

Output Port

Input Port

VC0
VC1
VC2
VC3

VC0
VC1
VC2
VC3
4 service levels (statically assigned VCs)

Implementation style:
- bundled data
- Significant area reduction over QDI approach

8-bit data flits

Synchronous versus Asynchronous router study
- Throughput is reported to be similar
- Minimum Latency (head flit) input to output (0.35um, typ. PVT)
  - Synchronous 3.7ns
  - Asynchronous 13.0ns (x3.5)
MANGO Clockless Network-on-chip

Input 1

Input P

non-blocking switching module $P_x(P^*V)$

ALG Admission Control

Output 1

Output P

SPA

VC Buffers
MANGO Clockless Network-on-chip

• Non-blocking switching network means link access arbitration is all that must be considered for hard QoS guarantees

• VCs are assigned statically (no contention)
  – Simple BE router used to program GS router (not shown)

• Basic Static Priority Arbiter (SPA) is preceded by admission control logic
  – Part of Asynchronous Latency Guarantee (ALG) scheduling algorithm (see ASYNC’05 paper)
  – Prevents lower priority flits being stalled more than once by each higher priority flit
MANGO Clockless Network-on-chip

- 515MHz port speed (WC, 0.13um)
- 32-bit data flits
- Implementation style:
  - Internally uses a bundled-data (RTZ) circuit style
  - Links use a DI two-phase encoding
- Router Latency ~5.2ns
  - Switch ~2.1ns, VC Buffers/Control ~1.2ns
  - VC merge ~1.6ns
- MANGO provides hard latency/throughput guarantees unlike other VC prioritization based schemes
Low-Latency Best-Effort Asynchronous Networks
Improving Network Latency

• Asynchronous router latency can be high
  – Fine-grain pipelining can provide good throughput figures but control overheads can extend latency
    • Completion detection, RTZ phase, H/S
    • Fast combinational matrix arbiters have also been replaced by cascaded MUTEXes or complex priority arbiters
    • Overheads even greater in a BE router that must allocate VCs dynamically

• Approaches to reduce latency?
  – Speculation
  – Decoupled control and data networks
Low-Latency Asynchronous Routers

• Exploit speculation?
  – Use Priority arbiter organisation
  – Assume only a single grant will be present after lock is asserted
  – Use MUTEX grant outputs to steer data immediately

• Issues
  – Complex abort procedure?
  – Invalid data and DI encoding?
  – Careful not to make common-case slower
Decoupled Control and Data Networks

Idea: Operate two independent networks:

1. **Control Network:**
   Simple/fast and lightly loaded

2. **Data Network:**
   Supporting virtual channels, packets, wide datapath
Decoupled Control and Data Networks

- Control network runs ahead of data network, hiding latency of scheduling logic
  - In an asynchronous environment, each network will operate at its natural rate
- Control network latency will be much lower compared to data network
  - Narrower links and simpler datapath
    • No virtual channels - little arbitration, less switching
  - Less traffic, single control flit per packet only
  - Could also exploit ‘fat’ wires and early requests to send packet
- Separate control and data networks can also be exploited in synchronous network [Peh/Dally]

Decoupled Control and Data Networks

- Schedule is queued and steers incoming data flits (data flits contain no routing information)
- Scheduler could perform VC allocation or both VC and switch allocation in advance
- Control network could also control power-gating of data network, waking network/links as needed from sleep mode.
Decoupled Control and Data Networks

• Design Decisions
  – Design can be simplified by keeping input port VC requests in order
  – Has obvious implications for performance
  – Out-of-order VC allocation scheme also possible
  – Performing switch allocation ahead of time could be inefficient
    • Order data actually arrives could be different

• Decoupled control and data networks may help hide scheduling overheads. More appropriate than speculation for asynchronous NoCs?
Synchronous or Asynchronous NoCs?
Comparing Approaches

• Little published work on asynchronous routers and networks
  – Single latency/throughput figures don’t tell whole story
  – Detailed comparative studies with real traffic are required

• Comparing synchronous and asynchronous designs has always been difficult
  – Often difficult to isolate impact of choice of system timing style, many things tend to be different:
    • Technology, circuit style, architecture
  – Difficult to reproduce and simulate asynchronous designs from published work. No notion of cycle-accurate model. Published work often lacks detailed control and datapath delays.
Questions about Asynchronous design?

• Testing asynchronous circuits
  – An asynchronous circuit replaces the clock with a large number of distributed state holding elements
  – Large area overhead associated with test
  – Testing of non-deterministic elements (MUTEX)

• Performance Guarantees
  – “Asynchronous circuits avoid issues of timing closure, they are correct-by-construction” – But performance guarantees are still required. Slow synchronous circuits are easy to build!
  – Value safe versus time safe
  – Less predictable, non-deterministic
  – Predicting performance is more complex

• EDA Tool Requirements

• Perhaps on-chip communication is an application where such characteristics can be tolerated?
Synchronous or Asynchronous?

• A clockless on-chip network appears to be an elegant solution although some questions remain:
  – Test
  – Performance concerns
    • Shouldn’t asynchronous designs offer latency advantages?
      – Fast local control, path/data dependent delays, DI interconnects
    • Perhaps asynchronous routers mimic synchronous architectures too closely?
      – Exploit flexibility, novel architectures, different topologies
    • Overheads for data-driven clocking or GALS currently look small in comparison

• Synchronous design has advantages too
  – Predictability and determinism can be exploited
    • fast single cycle routers possible
  – Global snapshot of state is good for scheduling

• Still lots of interesting research to be done
  – Need more data points
Conclusions

• High cost associated with both global synchrony and delay-insensitive circuits
  – Can relax constraints in both directions

• Which techniques achieve the best cost/benefit mix for on-chip networks?
  – Data-driven clocks look promising
Thank You

Comments/Questions?
Email: Robert.Mullins@cl.cam.ac.uk
Talk abstract, slides, notes and full bibliography at:
http://www.cl.cam.ac.uk/users/rdm34