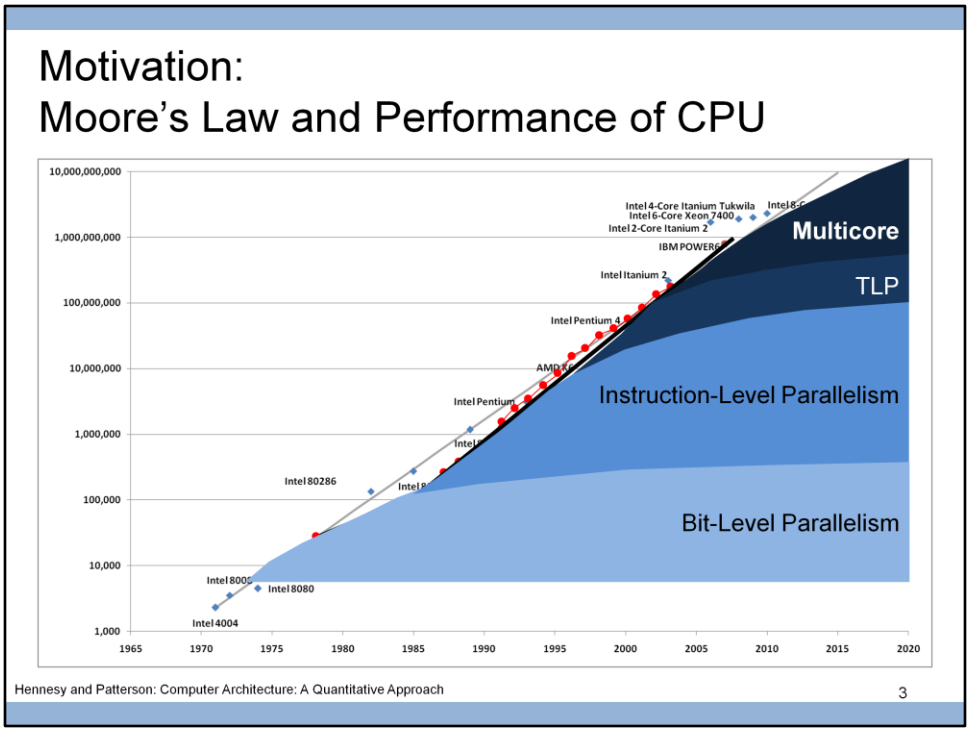
Design and Analysis of Networks-on-Chip in Heterogeneous Multicore Systems

Young Jin Yoon
<youngjin@cs.columbia.edu>

Contents

- Motivation and Applications
- System Drivers
- On-Chip Communication and Networks-on-Chip
- Modeling and Tools



- Moore's Law: The number of transistor has double approximately every two years, still holds.
- Performance graph:
  - First 25% / year, bit-level or data-level.
  - Second 52% / year, RISC / pipelining with frequency scaling ➔ ILP ➔ more than Moore's law
  - Third slowdown, Branch mispred. penalty, delay for register is even slower than c.c., memory delay ➔ TLP
  - TLP is still sharing pipelines ➔ limited performance, use global wires ➔ Multicore

Let's first get into the historical perspective of generalized microprocessor. The figure is the number of transistor on a die to implement generalized microprocessor, called the Moore's law. As you see, the Moore's law still holds today. If we draw the normalized performance graph with respect to the VAX machine around 1970s, which is pretty common figure used in computer architecture class, we notice that there is only 25% / year improvement at the beginning of the microprocessor, which is the half of the improvement compared to the Moore's law.

Thanks to the RISC architecture, pipelining, and instruction-level Parallelism, 52% / year improvement has been achieved since late 1980s. This is even better scaling than the Moore's law (Deeper pipelining, voltage scaling, frequency scaling).

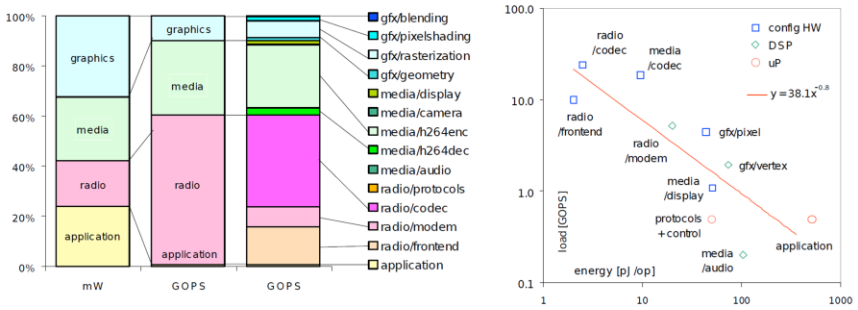
With ILP people observer that the latency between memory and CPUs getting bigger and bigger, in order to extract or "squeeze" more performance, people start thinking about Thread-Level Parallelism from that moment. However, TLP still shares the pipeline, does not prevent global wires, and shows poor power and thermal behavior.

However, we saw that those great scalability gets slow down, around 2000 where Pentium 4 has been announced. There are many things prevent the scaling, such as excessive power, un-scalable global wires, ILP diminishing return (The delay of registers that are put between pipeline is about to equal to the clock frequency, and branch prediction gets more and more penalties as having deeper pipeline stages.)

That was also the moment that we meet the power wall. Due to the excessive power, having a deeper pipeline with high frequency no longer works. As a result, people start talking about Multicores to maintain the scalability of the performance improvement with reasonable power budget.

Motivation:
System-on-Chip with Mobile Phones

- 100 Giga-Operation-Per-Second (GOPS) within 1W
  - 1 core running at 100GHz?
  - 1000 cores running at 100MHz?
- Performance-power vs. flexibility: 3.5G Mobile Phones



1.[2]. Berkel, Multi-Core for Mobile Phones

4

• Embedded market have tighter power budget.
• Mobile Phones:
  • 100GOPS within 1W power budget ➔ impossible to implement only with generalized microprocessors.
  • Optimized with Hardware ➔ performance-power vs. flexibility trade-off.
  • Example ➔ small dissipation but large GOPS for radio, opposite for application
• People want to have more functionality or better performance ➔ the number of cores for both keeps increasing

While the generalized microprocessor is finally arrived to multicore, it was already common on embedded or consumer electronic market because of tighter power budget.

The Berkel's paper explains why we already have multicores in embedded devices. If we consider all functionalities of 3.5G mobile phones and translate them into Giga-Operations Per Second (GOPS), we need 100 GOPS within 1W. To implement this functionalities purely with generalized microprocessors, we need 1 core running at 100Ghz, or 1000 cores running at 100Mhz: Intuitively any combinations of the number of cores and clock frequency is infeasible to meet the power budget of mobile phones.

As an alternative, we can sacrifice the flexibility of generalized microprocessor to get more performance with less power by implementing some functionalities into hardware cores. There is two big trade-offs here: Performance-power versus flexibility.

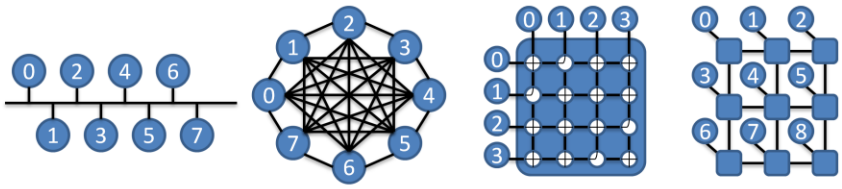
These two graphs show the result of those mappings. For example, on the left, there is small amount of GOPS required for application, but it dissipate more power. The trend is exactly opposite for radio.

If we draw the relationship between workload and power into two dimension, That's the figure on the right. If all application is implemented with microprocessors, those spots will me draw a proportional line: More GOPS means more power. However, application mapping tend to fit into a opposite way due to the heavy hardware optimization to meet the constraints. This is not a coincidence: This curve is the result of 15 years of mobile phone evolution.

As people demand more functionality or high performance for a system, the total number of cores tend to keep increasing in both System-on-Chip and generalized microprocessors.

Motivation: Networks-on-Chip (NoC)

- How do we connect all cores?
  - Bus vs. Point-to-Point vs. Crossbar and Mesh



- Difference between NoC and other Networks
  - Less non-determinism
  - Local, High-performance networks
  - Design-time Specialization
  - Energy-constraints

1.[6]. Benini, Networks on Chips: A New SoC Paradigm

5

So, we know that there are multicores in both generalized microprocessors and systems-on-chip. Moreover, the number of cores tend to increasing. As core numbers are increasing, how do we connect all cores?

First, we can use a bus just like the people used to do. It is shared-medium by all cores and other components. With the bus, only one component can send a message at a time, and the message is broadcasted to all other components. Bus shows poor scalability due to the conflict among components, and inefficient in terms of power because every message is broadcasted.

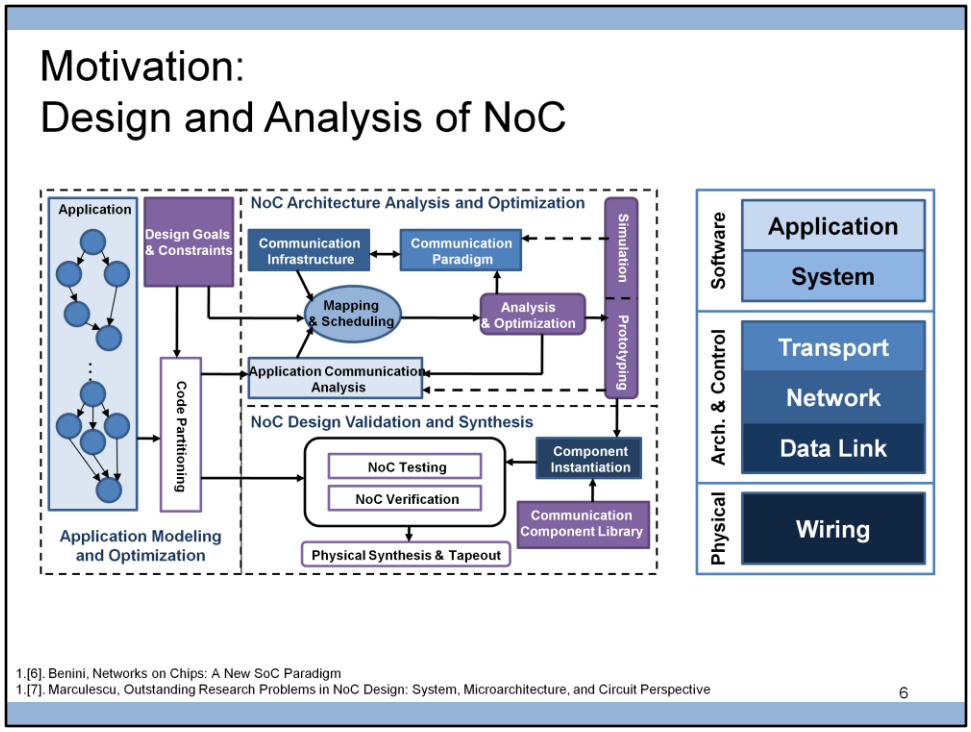
Instead, by connecting all components with point-to-point , each components can send a message to all the other components simultaneously without any conflict. However, the excessive number of wires required to connect all component as the number of cores scales up.

Thus, we need something between, such as crossbar and mesh. Crossbar is a reasonable non-blocking network that has a centralized controller. Based on the request of each components, the controller dynamically change the small switch in the crossbar to establish connections. It shows a reasonable performance. However, as the number of cores increases, the number and the length of wires to implement crossbar becomes large, which is also infeasible. (Even though scalability is better than point-to-point, it is still substantial.)

What if we can just use a small crossbar dedicated to a component, and link all crossbars each other, such as a mesh network explained in the figure? We have the reasonable number and length of wires per crossbar, and the controller to establish connections doesn't need to be centralized.

Thus, having networks on chip provides scalability of communications between multiple components. Now, we know that having networks on chip provides scalability, but the concept of network has been studied and analyzed more than 50 years. What do NoCs make differences from other well-know network implementations such as the Internet?

First, NoCs are less non-deterministic so far. The error rate of NoCs is far below than that of the Internet or System Area Network. Thus, a typical NoCs implementations do not allow dropping packets. Second, the distance between two node of NoCs is very close, and wire is relatively cheap to place. Instead of having one wire and serialize packets, NoCs are designed to send multiple bits at the same time due to the relatively cheap price of wires. Third, NoCs can be specialized at the design time. In wide area network such as Internet, the node can be dynamically entered and exited from and to the network. For NoCs, once the system is designed and fabricated, there is no need to dynamically adjust the network. Last, NoCs has energy and area constraints because they are the part of the chip.



Now, one may ask that how one can design an efficient Networks-on-Chip? There are two interesting papers that discuss it.

First, the Benini's paper introduce the term "Networks-on-Chip," and explain NoCs with the layered approach similar to the network stack used for WAN. Second, Marculescu's paper categorizes research challenges based on the NoC design flow.

In general, in order to design networks on a system, we first need to know what do we want to do with the system, the applications.
It could be either a software on a generalized microprocessor or a HW core. We also have design goals and constraints, such as the best performance and a power budget.

Given design goals and apps, we can partition the application into multiple parts and analyze communication patterns. Based on the communication, we can map the parts to various components of the system. If there are two or more app fragments that run on the same component, we also need to schedule them to meet the design goals or constraints. These mapping and scheduling is generally provided by the system layer.

Once we place all application fragment into components, we need to build the network that supports the communications. The way to communicate, how to control congestion, the logical and physical connections among components can be selected iteratively.

We also want to evaluate how good our network is in order to optimize performance. The performance can be measured throughout simulations, or analytically done by simple physical model.

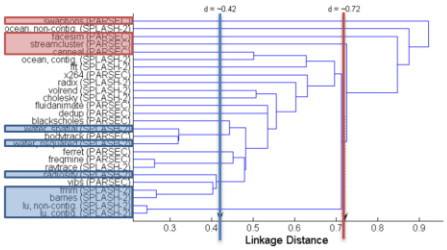
Once you meet the design goals, you can instantiate communication components from the library that contains physical property. The selected design
may or may not meet the requirement of physical constraints. If not, further optimizations can be done in this stage.

Finally, after all components meet the requirement and provide a good performance, we can test, verify and tapeout the system.

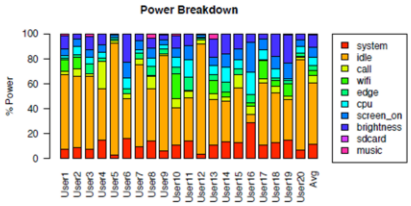
These two figures summarize the main theme of this talk. Using Both a top-down approach and the design flow I just described, I want to describe some of recent research trends related to the NoCs throughout the talk.

Applications:
PARSEC Benchmarks and User Interactions

- PARSEC benchmarks
  - Multithreaded
  - Emerging Workload
  - Diverse
  - State-of-art Techniques
  - Support Research



- Mobile architecture
  - Restricted power constraints
  - Users determine the power consumption



1.[4]. Biena, PARSEC vs. SPLASH-2: A Quantitative Comparison of Two Multithreaded Benchmark Suites on Chip-Multiprocessors Barrow-Williams, A Communication Characterization of SPLASH-2 and PARSEC
1.[5]. Shye, Into the Wild: Studying Real User Activity Patterns to Guide Power Optimizations for Mobile Architectures

7

In order to measure performance, people typically use benchmarks, which are representative applications for the target system. In generalized microprocessor domain, there is an interesting benchmark called PARSEC. The main motivation of PARSEC was that we have not had any benchmarks specifically designed for CMPs. Before PARSEC, researchers used to run experiment with SPLASH-2, which was released at the beginning of 90's. At that time, parallel machines were still relatively uncommon and most of them were owned by well funded organizations to majorly solve scientific applications.

There is 5 features of PARSEC that differentiates from the other parallelized benchmarks:

Multithreaded applications, supporting various comm. paradigm (MPI, pthread, and Intel TBB)
Emerging Workload: Include recently interesting apps such as computer vision.
Diverse: wide range of application domains, not just for scientific apps. Which is proved as a clustering algorithm.
Employ State-of-Art Techniques: enhanced algorithms and techniques
Support Research: allow instrumentation, workload manipulations, etc. …

Examples of each characteristic groups

Instruction Mix: floating point operations per instruction
Working Sets: Cache misses per memory reference
Sharing: Accesses to shared lines per memory reference

To compare PARSEC to SPLASH-2, Biena uses the principal component analysis(PCA) with 44 characteristics. PCA is a standard way to fairly quantify the distance between two or more objects. He also uses a clustering algorithm to see which benchmark suite gets clustered earlier. As a result, he showed that given 44 characteristics in groups, more than a half of SPLASH-2 benchmarks get merged before merging the first two PARSEC benchmarks, and there are more PARSEC benchmarks that are not clustered than that of SPLASH-2 before the relative distance = 0.72. Thus, he argue that PARSEC is more diverse than SPLASH-2.

For embedded system, there are also some application benchmark suite such as EEMBC or MiBench. However, just running benchmarks to represent applications of some embedded system may not reflect the real system. Shye's paper address the importance of user interactions in case of the applications in mobile architecture. As we already see, mobile architectures heavily constrained by powers. Thus, processing user-interactions may take more power than running applications. The power breakdown figure on the bottom-left represents power for each anonymous users. As you see, we should consider the different behavior of the user when designing a system.

EEMBC: EEMBC, the Embedded Microprocessor Benchmark Consortium (recursive, I guess)

Contents

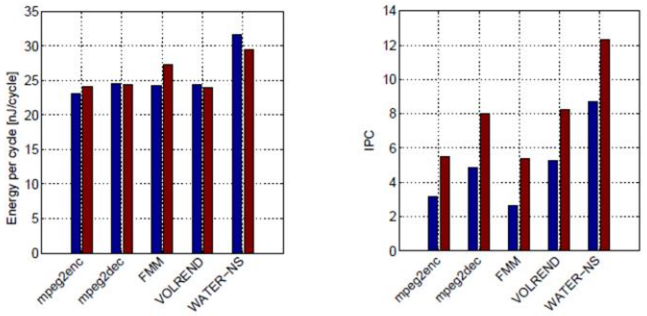
- Motivation and Applications
- System Drivers
- On-Chip Communication and Networks-on-Chip
- Modeling and Tools

8

So, we briefly covered Motivations and Applications. We also need to understand the hardware modules or system applications that affect the behavior of NoC to design better NoC. In this part, I will mostly talk about that system components that produce and consume messages from and to Networks-on-Chip.

Core Parallelism, Power, and Temperature

- Performance and Power
  - Same total parallelism (4P-8W vs. 8P-4W)
  - Same power but better throughput on 8P than 4P
  - Energy-Delay Product (EDP) and Energy-Delay^2 Product (ED2P)



2.[2]. Monchiero, Design Space Exploration for Multicore Architectures: A Power/Performance/Thermal View

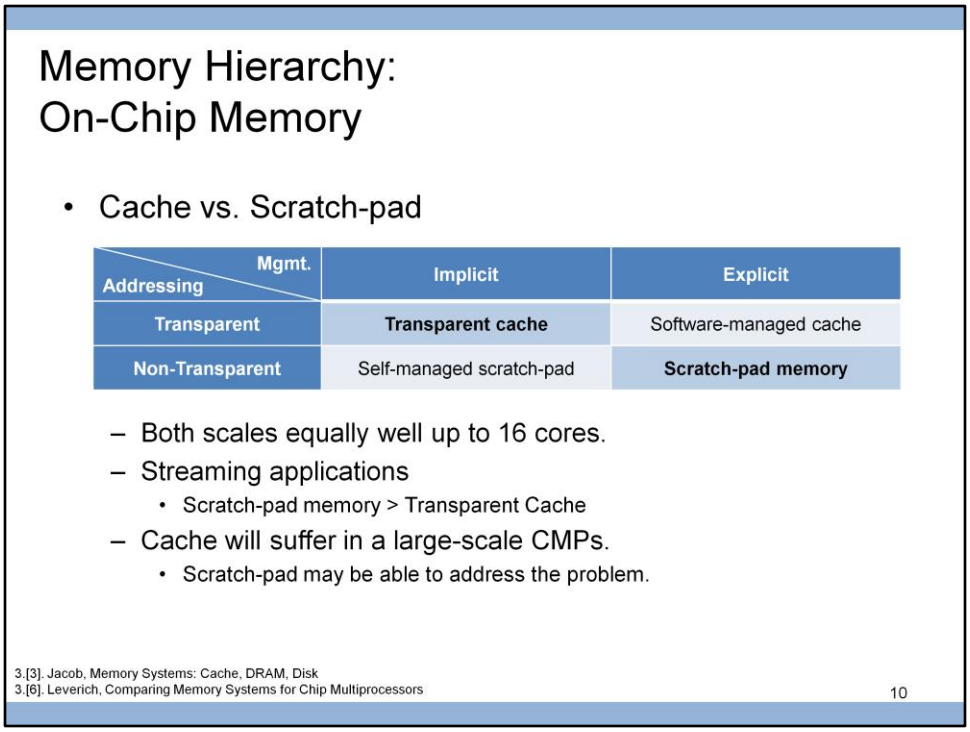
9

A generalized microprocessor, core, or processing element is the first component that produces the traffics for NoCs. We have various design spaces for cores such as total number of cores, the number of threads, and the number of issue-width for each core. Since these design spaces determine the injection rate and the number of components in a system, it is worth to know as a NoC designer.

The power-performance trade-off is one of the important consideration to design cores. Monchiero explores these design spaces and analyzed them.

For example, the figures below compares the performance, power, and temperature of two different core configuration. Blue and red is based on
The same total parallelism (4-processor with 8-issue vs. 8-processor with 4-issue). As you see, the total energy dissipated by both architectures
are very similar, while the performance of 8-core with 4-issue archs. outperforms 4-core 8-issue archs.

Monchiero uses the energy-delay and the energy-delay square products to quantify the trade-off between power and performance. Typically EDP is used for Embedded computing, while ED2P is for High-performance computing. With ED2P, the authors argue that large Number of fairly narrow cores are preferred.



The round-trip time from the HW cores or generalized microprocessor to the off-chip DRAM is fairly slow, and getting worse as technology evolves. It's about couple hundred c.c.. In order to reduce the latency, there are multiple on-chip memory components in a system. Those on-chip memory can be categorized into 4 types depend on the visibility of memory management and addressing scheme to programmers.

The traditional implementation of these on-chip memory is using caches, which is completely invisible and uncontrollable to users. There are other implementations that can be controlled by a program, but the address is still invisible, or vice versa. As an extreme, we can also provide on-chip memory that is completely visible to the program, called a scratch-pad memory. For example of the scratch-pad memory, you can see it in the SPE element of IBM CELL processor.

Leverich compares the performance of two-commonly used on-chip memory implementations: a transparent cache and a scratch-pad memory. With the optimized program and compiler optimization for each on-chip memory implementations, it turns out that both scales equally well up to 16 cores.
However for streaming applications, the scratch-pad performs better than cache, but the gap can be mitigated by having small optimization to the cache, such as a pre-fetching module.

However, Leverich also discusses the scalability of the cache. He argue that the cache will suffer in a large-scale Chip-Multiprocessors due to the inefficiency of auto-management, while scratch-pad may be able to the address the scalability problem well. As a programmers perspective, This is somewhat consistent to the projection of the "multikernel" paper.

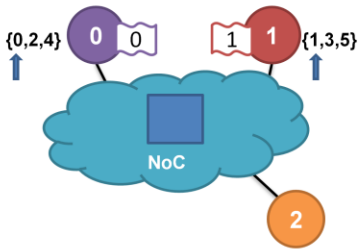
Intelligent NoCs for Cache Coherence: INSO and INCF

- Cache Coherence Protocol
  - Snoop-based vs. Directory-based
- Snoop-based Coherence in unordered NoCs:

1. Incorrect

In-Network Snoop Ordering (INSO)

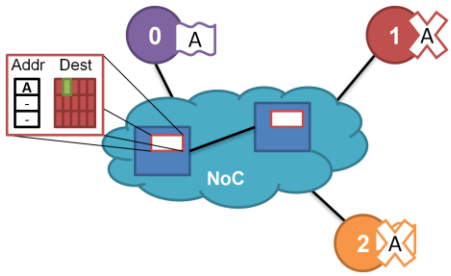
Route messages as ordered



2. Broadcast messages.

In-Network Coherence Filtering (INCF)

Filter Unnecessary Broadcasts



Agawal, In-Network Snoop Ordering

2.[4]. Agawal, In-Network Coherence Filtering: Snoopy Coherence without Broadcasts

11

If there are distributed on-chip memory dedicated to a core, they may contain different values for the same address. For example, when core 1 modify the value in address A, the modified value may not be propagated to core 0. Thus, we need to have a protocol that synchronize contents for the same address, called cache coherence protocol. To make cache coherence protocol work, it is important to provide a global ordering point. Based on the location of ordering point, cache coherence protocol can be categorized into Snoop-based and directory-based protocol. Snoop-based protocol uses NoC as the ordering point, while directory-based protocol has a dedicated module to manage orders called directory. Moreover, snoop-based is typically uses broadcast and fast, while directory-based uses unicast and slow.

On top of the broadcast, snoop-based coherence protocols require the ordered delivery of messages in NoC. However, if an unordered NoC has enough intelligence to dynamically route messages as ordered, we can still use the Snoop-based coherence protocol for fast synchronization. In-Network Snoop Ordering is an implementation of this intelligent network.

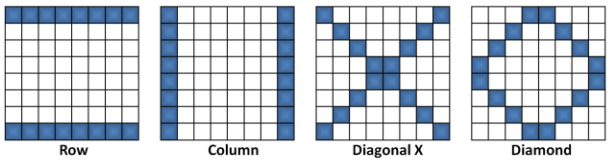
First, each node in the network has some unique order indexes. When a node transmits a message, it will assign an index to the message. If one or more request goes to the same memory controller, 2 in the left figure, there is always a router which is on paths of both messages. By recognizing the index and route messages as ordered, INSO simply solve the correctness problem of unordered NoC with Snoop-based Coherence protocol.

While some messages are essential to broadcast to make snoop-based coherence work, there exists unnecessary broadcast messages. If routers can record the request / reply and filter those broadcasts, we may also reduce the traffic thus improve the performance of NoC by avoiding useless transmission. That's the main concept of In-Network Coherence Filtering routers.

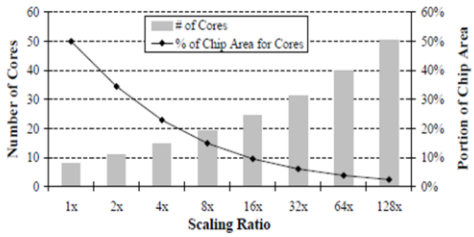
For example, when addr A is requested and broadcasted by routers, there are replies that represents non-existance of addr A. When the reply comes from 1 and 2 to 0, the router on the path record where the actual cache line exists. When it comes to delivering the request from 2 for the same addr A, instead of broadcasting the request, router can simply look up the record, send the message that goes to node 0 only, and filter the message that goes to the other nodes.

Memory Controller & Off-Chip Network

- On-Chip Memory Controller
  - Where to place them?



- Bandwidth Wall
  - Due to pin-limitations, power constraints, and package costs
  - Memory scales only 10% per year



2.[5]. Dennis, Achieving Predictable Performance through Better Memory Controller Placement in Many-Core CMPs
2.[7]. Rogers, Scaling the Bandwidth Wall: Challenges in and Avenues for CMP Scaling

12

One of the recent trend of both SoC and microprocessor is that the chip also includes memory controllers. For the embedded system, some companies start providing memory controller Intellectual Property(IP). For recent CPUs use on-chip memory controller such as Intel Nehalem, IBM POWER7, and IBM Cell processors.

Considering the on-chip memory controllers, one may ask this. where do we need to place the on-chip memory controllers for better performance?
Dennis addresses the memory controller placement problem by using Genetic Algorithm. He found that Diagonal X and Diamond shows the best performance among in 8x8 mesh network. He also argue that Diamond is preferable Diagonal X has some memory controllers in the center of the chip, which may require the long wire to go off-chip, or possibility of hot-spot.

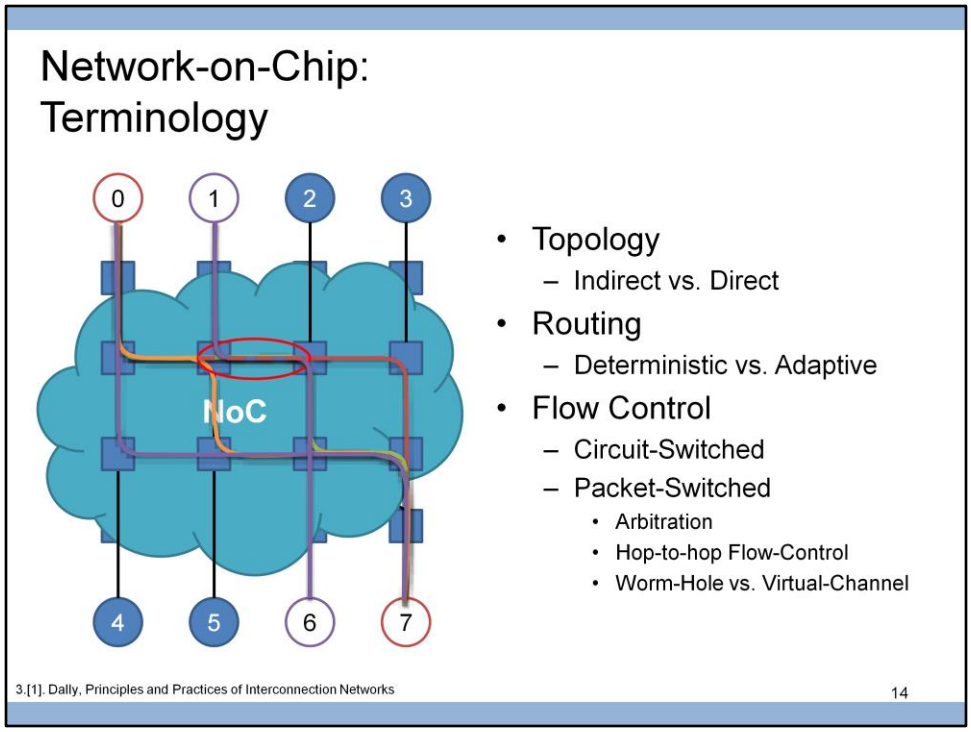
Since on-chip memory controllers is connected to the off-chip memory through the off-chip network, the behavior of off-chip network affect the traffic of Networks-on-chip. Rogers introduces the new challenges called bandwidth wall, which limits the scalability of the number of cores as technology evolves. Due to the pin-limitations, power constraints, package costs, and the low-scaling factor of the off-chip memory, Rogers predict that the bandwidth between on-chip and off-chip will be limited. That is, the chip includes more and more caches rather than cores. The figure on the right shows these trends. After the four generation (16x) from the current technology, you can see that only 10% of area is used to place cores, which is only 25 cores in a system. To reduce the size of caches to support bandwidth, Rogers also argue that various bandwidth saving technique such as link and cache compression will help to increase the total number of cores in a system.

Contents

- Motivation and Applications
- System Drivers
- On-Chip Communication and Networks-on-Chip
- Modeling and Tools

13

So, we go over the research related to the system components that affect the design of NoCs. Now it's a time to see how NoCs look like.



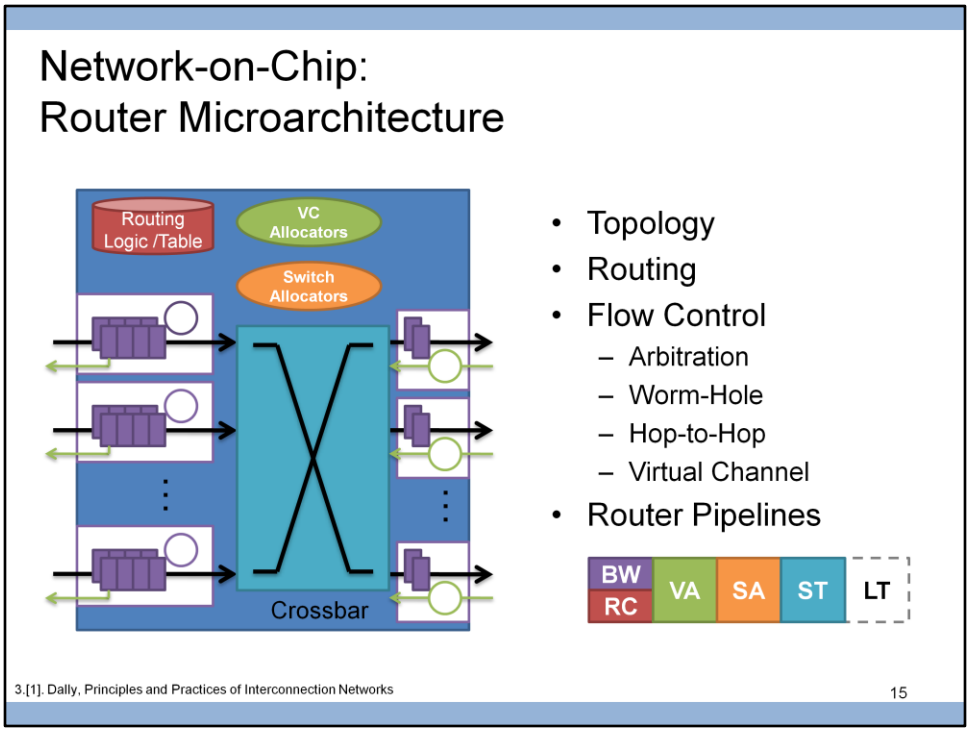
With given components, we want to establish the connections among them. The first question is how do we connect components. We may connect them with multiple intermediate switches, or switches that connect components and the other switches with no intermediate ones. The former one is called indirect network, while the latter one is called direct network. The examples of indirect networks are butterfly and CLOS, and for direct networks are mesh and torus.

With given topology, let's assume that the node 0 wants to communicate with the node 7. There are multiple paths exists in the topology. To determine which paths we want to use, we need to have the routing policy. Deterministic routing means that there is only one path for the same source & destination pair, while adaptive routing represents that multiple paths can be used dynamically based on the network status.

With given topology and routing policy, we can have multiple flows that want to share one or more channels at the same time. To control the sharing, we need a flow control mechanism. First, we can completely avoid conflict between two or more flows by reserving the resources over a path of a flow before sending data, called circuit-switched network. Instead, the conflict of flows can be dynamically resolved by arbitration mechanism. With arbitration, the non-selected flows should be stored in somewhere to avoid dropping data. Thus, we need to have buffers. Because the size of buffers are limited, we also require to have backpressure mechanism to stop the non-selected flows, called Hop-to-hop flow control.

Instead of blocking the non-selected flow until completing transmission of the selected flow, we may want to use time-multiplexing among multiple flow to enhance the performance of the network. The blocking mechanism is called as worm-hole flow control, while the multiplexing one is called virtual-channel flow control.

By going though the terminology, we also saw that the required functionalities of the network. Now, let's see how these functionalities can be implemented in a router, which is represented as a square in the figure.



First, topology determines the radix of a router, which is the number of input and output units. Each input and output units, there is a channel that has multiple wires. In every clock cycle, a fraction of packets, called flow control unit or flit, is delivered from the channel The radix of a router also related to the size of crossbar. Next, routing can be implemented as a routing logic or a routing lookup table. For the dynamic routing, lookup table is preferred. Mostly the routing policy is implemented as a logic in NoCs. Second, to control the flow, the allocator for switch is required to setup the crossbar. In case of worm-hole routing, input and output buffers per each input and output ports are required. To prevent the buffer overflow, some hop-to-hop flow control is also required.

In case of virtual channel instead, you need a buffer per virtual channel because while forwarding packets of one virtual channel we need to save the packet of the other virtual channels. Moreover, to select the output virtual channel, we also need to have virtual channel allocators.

Based on the microarchitecture, we can divide the overall procedure into 6 parts. First we need to save the incoming packet to a buffer, and compute
the designated output port. Then we can allocate the output virtual channel, and switch. Finally we send flits to the switch and they go to the output links. These divided functionality can be pipelined to improve the performance of the overall network. This is the standard five-stage pipeline.

Performance and Cost Metrics

- Performance Metrics

	Delivery Speed	The number of Delivery
Ideal	**Zero-load** Latency	**Bi-section** Bandwidth
Average	**Average** Latency	**Average** Throughput
Worst	**Maximum** Latency	**Peak** Throughput

- Cost Metrics
  - Average or peak energy/power consumption
  - Network area overhead and total area
  - Average or peak temperature

3.[1]. Dally, Principles and Practices of Interconnection Networks
1.[7]. Marculescu, Outstanding Research Problems in NoC Design

16

The performance of network can be measured in two ways. We can first measure the speed of packet delivery in clock cycles, or the number of delivery with a given number of clock cycles. The former one is typically called as latency, and the latter one is called as bandwidth. The latency with no contention is called as zero-load latency. In case of regular topology, the maximum possible number of delivery in a given time is typically represented with bi-section bandwidth. Bi-section bandwidth is the total number of wires that is between any cut of the network. The average and the worst bandwidth also can be measured by the efficient usage of given bi-section bandwidth, called throughput. Thus, the range of through is from 0 to 1.

While the performance metrics are directly related to the functionality of a system, cost metrics are the other metrics that should be considered for designing a system. The cost metrics of NoCs are average or peak energy or power consumption, network area overhead and the total area, and the average or peak temperature of the networks.

Given cost and performance metrics, we can determine whether the NoCs are designed well or not.

Note: Ideal bandwidth is different from bi-section bandwidth: We can derive the throughput equations of the upperbound and lowerbound with bi-section bandwidth. However, for regular topology, such as ring or torus, all bi-section bandwidth is the same. Therefore, those two bounds will be converge to the ideal bandwidth (i.e. In regular topology, bi-section bandwidth = ideal bandwidth.)

Router Microarchitecture: Reducing Pipelines

- Spend 4 c.c. for 1 link traversal

- Speculative Routing
  - Let VA and SA execute simultaneously
  - Rollback if VA depends on SA

- Lookahead Routing
  - Pre-compute and forward routing infos

- Pipeline in SPAROFLO
  - VA: on the critical path
  - SA before VA for the simple VA stage

3.[1]. Dally, Principles and Practices of Interconnection Networks
3.[3]. Kumar, A 4.6Tbits/s 3.6GHz Single-Cycle NoC Router with a Novel Switch Allocator in 65nm CMOS

17

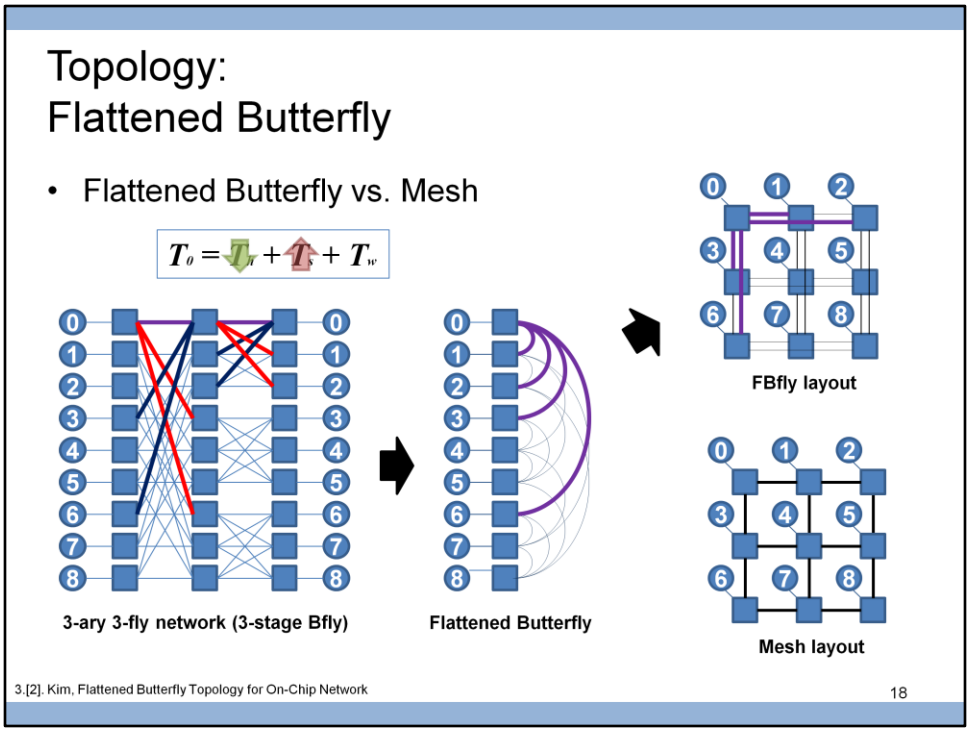
Based on the performance metrics and with a given baseline router, one may want to optimize the latency of the network. For example, with router pipeline, it seems that spending 4 clock cycle just for 1 link traversal seems inefficient in terms of latency. The various approaches to reduce the number of pipeline stages has been proposed.

First, you can let virtual and switch allocations execute simultaneously, especially under the low congestion in a router. If there are mismatched part between the result of virtual and switch allocations, we can rollback and start computing VA and SA stage sequentially. This strategy is called speculative routing. This approach is especially good for the network low injection rate; however, it may suffer with high congestion due to the extra stage introduced by speculation.

Second, under the deterministic routing, it is relatively simple to pre-compute the designated output port for the next router, if we already know what's the output port for the current router. Thus, with the aid of additional wires that sends the pre-computer information, lookahead routing can parallelize the next routing computation stage with the virtual allocation stage.

Lastly, if we see the virtual channel allocation stage, all virtual channels for all output ports need to be searched to find the best for all input virtual channels of all input ports. The result of these allocations are further filtered out by allocating input virtual channel to use crossbar. Thus, the virtual channel allocation stage typically takes more time to be completed than the switch allocation in seconds (not in clock cycles). To eliminate critical path of the virtual channel allocation stage, one may think that allocating switch before the virtual channel allocation may reduce the overall time. The pipeline in SPAROFLO apply this idea and try to allocate switch first. If there is multiple granted output for an input port, the output port with higher priority of virtual channel is selected. With all other various optimizations, both switch and virtual allocation stages can be combined and executed in a clock cycle.

Note: Area – performance (latency) trade-off



As an alternative, we can use different topology to reduce zero-load or average latency. Kim introduces flattened butterfly topology to provide the low average latency in NoCs. The figure is a traditional 3-ary, which means 3 nodes per group or 3x3 routers, 3-fly which means 3-stage butterfly network. Except the connections, if we flatten or combine the all routers in a row into one, we could build flattened butterfly. The layout-friendly figure of flattened butterfly, which is isomorphic to the original, it is represented as a directed network of which all nodes in a row or a column is linked together. We could compare the flattened butterfly to the mesh by having the same bisection bandwidth. For example 16-bit wide channels in the mesh layout has the same bisection bandwidth to the flatten butterfly with 8-bit wide channels. Even though the layout seems unintuitive due to the long-distance channels, Flatten butterfly reduces the average latency.

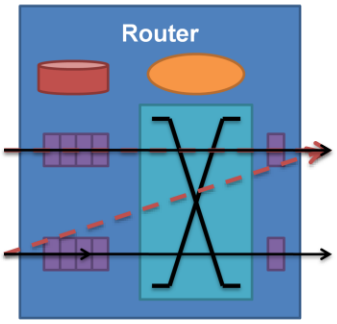
Theoretically, zero-load latency can be divided into three parts, head latency, serialization latency, and time to flight on the wires. Head latency is measured from sending the first flit to the receiving the same flit. Since a packet can be divided into multiple flits, it will introduce additional latency to wait to receive the last flit of the packet. In physical layout, the distance of channels could be different, which is expressed as T_w in this equation.

Flatten butterfly reduces the header latency because of the smaller average hop count, and increases serialization latency due to the narrow channel. Kim argue that the zero-load latency of mesh is imbalanced and have too much penalty to the head latency. By re-adjusting head and serialization latency, flatten butterfly provide more balance with zero-load latency, which eventually reduces the average latency.

Or intuitively, a flit spends multiple clock cycles in a router. If we avoid the number of routers passed by a flit, we may reduce the overall latency even if it takes multiple clock cycle to traverse the long distance channels. If the benefit of avoiding routers is greater than the serialization penalty, flattened butterfly can outperform the mesh topology.

Bufferless Network

- Buffers in NoC
  - Energy, area, complexity
- Can we design a network without buffers?
  - Deflective routing vs. Packet/Flit dropping
- BLESS
  - Deflective bufferless Network
  - Problems: Injection problem & Livelocks
  - Pros: Power consumption
  - Cons: Throughput, Latency …



3.[5]. Moscibroda, A Case for Bufferless Routing in On-Chip Networks

19

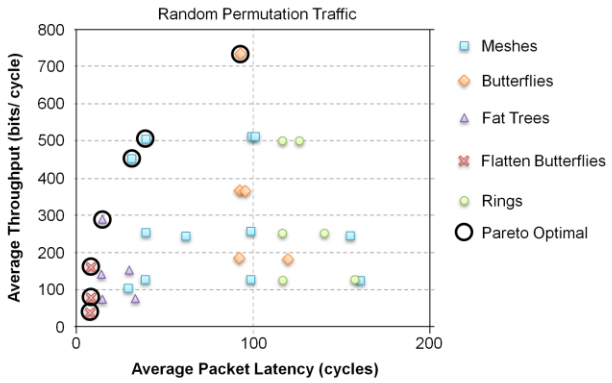
Typically buffers in NoCs consume energy, and area. Implementing routers with buffers also introduces more complexity. Then can we design a packet-switched network without buffers? To eliminate buffers, we could drop flits or packets.

However, there is alternative routing scheme called deflected routing. Deflective routing, or hot-potato routing, intentionally misroutes, or deflects, flits/packets if all productive output ports are used by the other packets. That is, deflective routing use channels as buffers. Moscibroda introduces a deflective bufferless network for Network-on-Chip. Due to the nature of deflection, deflective routing may cause livelock or out-of-channel problem. Moscibroda resolve these problems by prioritizing old packets and sending information of input ports from neighbors. If the use of input ports can be detected 1 clock cycle before, the router send a signal that stops the injection of a component that is attached to the router.

The experimental results report that there is a significant energy saving by eliminating buffers with reasonable performance degradation in the chip-multiprocessors. However, the performance degradation for both latency and bandwidth become worse with high injection rates. In the worst-case performance, there is 3.2% ~ 17.1% of system performance degradation with 28.1% ~ 14.0% of energy saving, while on average there is 0.15% ~ 0.65% of system performance degradation with 46.4% ~ 42.5% of energy saving.

Polymorphic On-Chip Networks

- There is no network to fit all workloads.



3.[4]. Kim, Polymorphic On-Chip Networks

20

As you see, you can improve performance by changing topology, or save more power by eliminating buffers and using deflected routing. One may ask, is there any network configuration that is the best for any performance metrics. In order to answer the question, Kim presents a detailed design space exploration and Pareto analysis of NoC architectures. She finds that there is no network to fit all workloads. For example, in case of Random Permutation Traffic, there are performance results for mesh, butterfly, fat tree, flatten butterfly, and ring topologies. If we draw circles for the best latency configuration for the same bandwidth or the best bandwidth for the same latency, the optimal solutions, called Pareto optimal, will be drawn like this.

We can see that topologies on pareto optimal configurations are not the same. Butterfly topology is preferred to the applications that requires the large bandwidth but less critical to the latency, while flatten butterfly may be the best option for the latency critical applications that shows the low-injection rate.

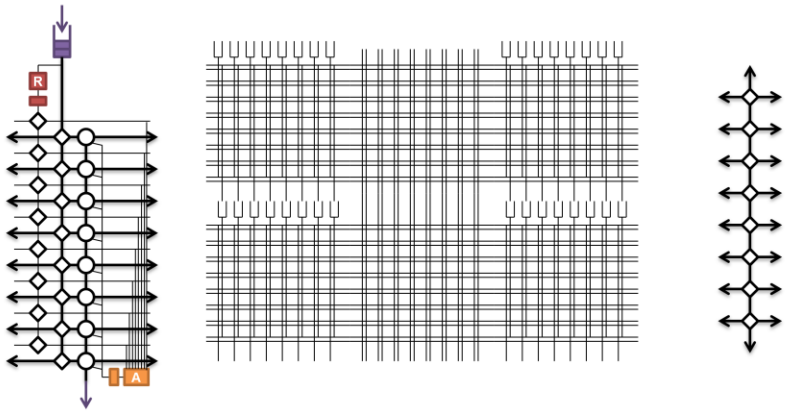
Now the question becomes “can we change the network configurations such as topology, buffers or the bi-section bandwidth before running each applications?” That is, can we change NoC configurations after the design stage?

Pareto optimal curve: various network topology
Uniform Random: fat tree & mesh offer the lowest latency
Random Adversarial: Fat tree
Local Adversarial: mesh & ring
Region:
Collection of Routing unit, input output queues, crossbars, and arbiters.
Each lane in Region can be configured as a router.

Interconnection between router is managed by crossbars btw two regions and CROSSBAR.
Example on the next slide

Polymorphic On-Chip Networks

- Let's provide Network resources
  - Users can statically configure NoC before running applications



3.[4]. Kim, Polymorphic On-Chip Networks

21

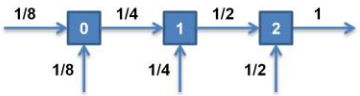
Changing NoC configurations after the design stage can be simply done by providing seas of network resources and configurability instead of providing a complete network itself. For example, we may want to provide a really basic buffer, routing logic, switch arbiter, and wires to control arbiter from routing logic. And, we can also have a simple switch driven by the arbiter. Finally, we need to have a line that drives the data from the input buffer to the switch. The set of resources I just explained is called a slice.

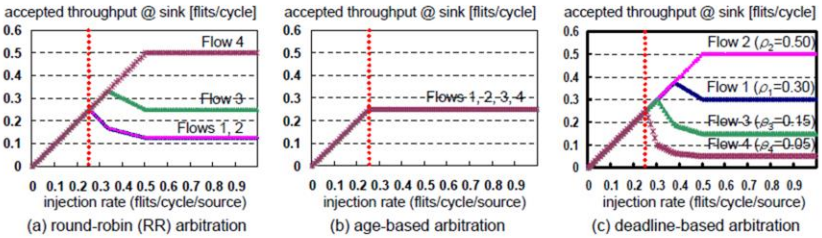
Multiple slices can be clustered into a region by stacking it column-by-column. Since a slice can only forward data down, left, or right, there is a crossbar that takes output of slices and deliver it to the input of the other slices. When users configure the network, multiple slices within or across regions can be combined and act as a router. With these resources, users of a system can freely configure the network before running the application.

Since the polymorphic on-chip networks provide compose-able network resources instead of optimized network, it occupies more area. However, unless the application is fixed or the area constraint of system is tight, polymorphic on-chip network provides the flexibility to satisfy the different performance demands of different applications.

Quality of Service (QoS)

- Quality of Service
  - Local Fairness ≠ Global Fairness
  - Some packets are more important than the others.
- Round-Robin vs. Age-based vs. deadline-based





3.[6]. Lee, Globally-Synchronized Frames for Guaranteed Quality-of-Service in On-Chip Networks

22

So far we talk about the performance in terms of average. However, we know that the average does not explain how fair the traffic is. The topic of fairness and prioritization of flow is called Quality-of-Service in Networks-on-Chip design.

For example, arbiters or allocators in many NoC implementations use round-robin arbitration. Round-robin provides a local fairness because resources attached to the allocator have the same chance to get forwarded on average. However, it does not mean that the entire flow of the network will be fair. For example on the right, each node represented as a square is locally fair. However, if we consider the each input flow, we can see that the flow injected to the latter stage gets more injection rate than the flow from the others. In 2D mesh, this behavior can be translated that the flow from neighbor has more chance to be accepted to the same destination, if the router uses round-robin switch allocator.

By encoding the timestamp of injection in each packet and by looking up the timestamp and route packets based on the oldest-timestamp first, the flows can be globally fair, as shown on the center of the bottom figure. This timestamp encoding with oldest-first strategy is called age-based arbitration.

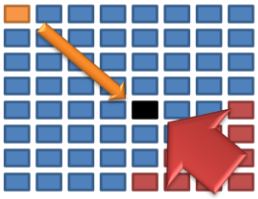
Sometimes it is possible to have some packets that are more important than the others. As long as the importance can be quantified such as weight, we can encode the timestamp of the deadline that a packet should be delivered to the destination. It is called a deadline-based arbitration. Again, the arbitration is based on the oldest-timestamp first to meet the deadline.

QoS: Three approaches

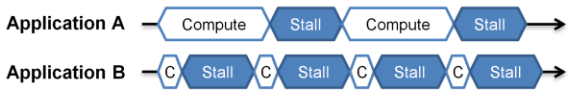
- Source-Based QoS
  - Global Synchronous Frame (GSF)

    : Source-based static deadline-based allocation
- Router-Based QoS
  - Preemptive Virtual Channel (PVC)

    : Router-based dynamic bandwidth allocation



- Application-Aware QoS
  - Performance of applications, not the performance of NoCs.
  - Stall-Time-Criticality (STC): Application sensitivity w.r.t. latency



3.[6]. Lee, Globally-Synchronized Frames for Guaranteed Quality-of-Service in On-Chip Networks
Grot, Preemptive Virtual Clock: A Flexible, Efficient and Cost-effective QoS scheme for Networks-on-Chip
Das, Application-Aware Prioritization Mechanisms for On-Chip Networks

23

There are three major approaches for the Quality of Service in Networks-on-chip.

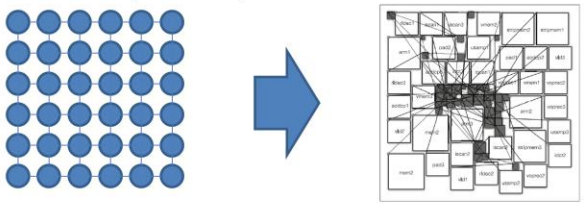
The first one is Global Synchronous Frame proposed by Lee, called GSF. It is a slight variation of the deadline-based arbitration. Instead of encoding the deadline, GSF uses the deadline for each virtual channel of the network. Because GSF is source-throttled Quality of service, it is possible that GSF may reduce the overall throughput of the system. For example, if both a traffic from the top and the bottom expressed as arrows want to go to the destination expressed as a black node. With GSF, a node on the top left will not inject many traffic even though there a fluent network resources that are not used, while, all nodes on the bottom right tries to use the network resources as much as possible.

If the flow fairness is controlled by routers, the packets from top left will get forwarded to the black node as close as possible, and the finally scheduled based on the priority. Thus, Preemtive Virtual Channel is proposed by Grot to achieve both fairness and the bandwidth utilization. Instead of sending deadline, PVC directly encodes the flow weight to each packet so that each router can dynamically compare the priority of the packet.

Lastly, the Application-Aware QoS by Das comes from a different concept. What we really care is the performance of applications, not the performance of networks. We optimize the network because we hope that network performance improvement will lead the overall performance improvement. For QoS, we optimize the fairness and the bandwidth of the network because we can achieve the better performance. However, there are some counter intuitive result revealed by Das. He argue that each application has different demands for network, and should be treated differently to improve throughput. Das also quantifies the application sensitivity with respect to the network latency called stall-time criticality. For example, because the application B stalls more than A waiting for the memory, we can say that the application B is more critical to the latency than the application A. Based on the criticality, Das also proposes various scheduling approaches for a router.

Hop-to-hop: wires and interconnects

- Network-on-Chip: Floorplans



- Interconnect Requirement from ITRS

Cu Interconnect (ITRS)		2011	2012	2013	2014	2015	2016
Gate Length (nm)		16	14	13	11	10	9
Intermediate	RC Delay (ps)	1291	1455	1842	2406	2670	3341
	Line length (um)	16	15	12	9	8	7
Global	RC Delay (ps)	487	557	705	921	1004	1297
	Line length (um)	26	23	19	15	13	11
FITs /m /cm^2		2	1.6	1.6	1.4	1.3	1.1

4.[3]. Pinto, COSI: A Framework for the Design of Interconnection Networks
1.[1]. International Technology Roadmap for Semiconductors (ITRS): 2009 edition

24

Now, let's try to get into a bit lower level. Let's say based on the performance and cost metrics, we decide routers, topology, and flow-control mechanism. It looks nice, but ff we place them into the layout, it will become ugly again. The figure on the right is the floorplan of video object plane decoder with mesh network after the synthesis. You can see that almost all routers gets centered, so that there exists a long channel that deliver the signal from the side to the center of the chip.

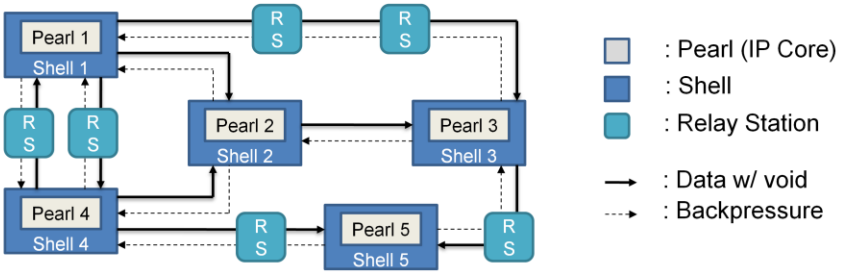
These long distance wire may produce problems because it may not be delivered in a clock cycle. This table is the part of projections for interconnections presented by ITRS 2007 (2009 version is still under the review). RC Delay reported in picosecond represents the delay to deliver the signal for 1mm without considering the scattering effect, while the line length is the minimum length that does not show the of that RC delay. For example, in 2011 the RC Delay of a intermediate router is optimistically 1291 ps, which is almost 3 clock cycles in case of 2Ghz Clock frequency. This table basically shows that we may not deliver the data from a router to a router in a clock cycle. Then the question becomes, what approach should we deal with to solve this problem?

One of interconnection challenges described in Interconnect section of ITRS 2009:

1. Achieving necessary reliability: (Failure in Time per m length wire per cm^2 of active area)
2. No scattering RC delay exceeds 2Ghz c.c. requirements (2Ghz == 500ps)
3. Maximum Line length with no scattering.
4. If we combining 3 and 4, problem gets worse.

Latency Insensitive Design

- Time-to-market constraints
- Intellectual-Property design modules (IP Cores, Pearls)
- Interconnect latency
  - Hard to estimate in early design stage
  - Conservative estimation: suboptimal design
- Latency Insensitive Design(LID)



4.[2]. Carloni, Coping with Latency in SOC Design

25

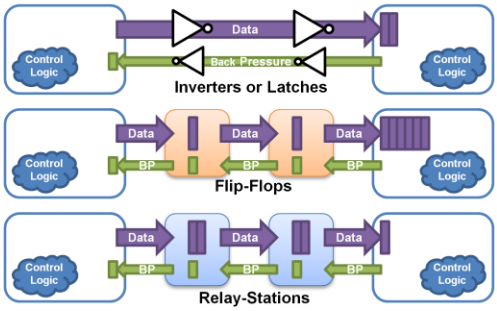
Before introducing any Network-on-Chip related approach, Let me introduce the Latency Insensitive Design presented by Carloni, which also deals with communication among components of a patient system – a synchronous system whose functionality depends only on the order of each signal's events and not on their exact timing.

Due to the time-to-market constraints for the embedded system, designers want to reuse their modules. And, some companies such as ARM sells its design as an intellectual property cores with given constraints. On the other hand, it is very hard to estimate the interconnect latency in early design stage because of process variations, crosstalk, and power supply drop variations. Since, the conservative estimation of interconnect latency will lead to suboptimal designs, we need another approach to combine and consider connecting different cores.

For instance, let's assume a system contains 5 Intellectual Property cores, called pearls, and the communication graph can be drawn as follows. If all IP cores produce outputs or consume inputs in different number of clock cycles, such as pearl 1 consumes one input in every clock cycle while pearl 4 can produce output in every two clock cycle, we need to control the speed of each cores to provide the correctness of the system. We can first encapsulate shells to control communications of each pearls. In every clock cycle, shells produce either void or actual data, and consume actual data by enable the clock of their own pearls or filter out the void signal. If there is a long distance link between two shell, it is simply handled by placing synchronous repeaters called relay stations.

Hop-to-Hop Flow Control

- Channels between two routers
  - Longer is the wire, slower are delivered the messages.
- Put some intelligence on the channel!
  - Link pipelining with distributed buffers



ON/OFF	Credit	Ack/Nack
-	-	-
2+5K	2+3K	1+3K
2+2K	2+2K	1+2K

3.[1]. Dally, Principles and Practices of Interconnection Networks
3.[7]. Petracca, Distributed Flit-Buffer Flow Control for Networks-on-Chip

26

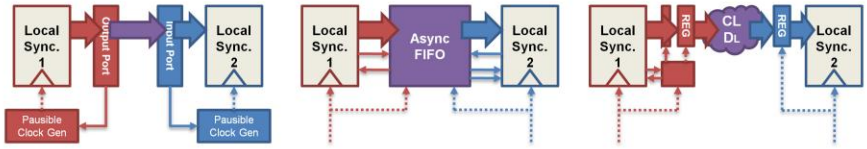
Let's go back to the NoC. There is a problem that wire signals cannot be delivered in a clock cycle. First we can put inverters or latches as an asynchronous repeater. However, these should be placed carefully to meet the clock cycle requirement under the global synchronous system.

As an alternative, you can place multiple flip-flops, called wire pipelining. When there is backpressure that stops the transmission, data in the flip-flops should completely delivered to the destination. Thus, there is a minimum number of input buffers to provide the correctness, as you see it on the table of the left side. These are the number of registers required to provide the correctness with k-stage flip-flops.

We can also use relay stations in LID. Because RS contains two buffers for data and some small intelligence, the backpressure is detected per relay-stations and will hold the data in the relay-station instead of send them all the way to the receiver. Thus, we only need one data register for both sender and receiver independent from the number of stage between the sender and the receiver. If we compare the number of storage required for k-stage, we can see that in any case, placing relay-station provides more scalability than flip-flops because RS contains an intelligence to stop the transmission by understanding the backpressure.

Globally Asynchronous, Locally Synchronous (GALS) Circuit

- The problems of Clock Distribution
  - Design Complexity, Noise, and Power
- Local clock w/ asynchronous communication



Property	Pausible Clocking	FIFO-based	Boundary Synchronization
Area Overhead	Small	Med to Big	Small
Latency	Low	High	Med
Throughput	Depend on clock pause rate	High	Med
Power Consumption	Low	High	Med

3.[9]. Krstic, Globally Asynchronous, Locally Synchronous Circuits: Overview and Outlook

27

The previous slides assume that everything happens on synchronous domain. However, if we allow asynchronous communication, things can be changed. Let's see what is the problem of globally synchronized system with deep-submicron technology. First. It becomes harder and harder to distribute the clock signal in 1 clock cycle. Designers use the carefully tuned H tree to distribute clock without skew. However, it suffers from noise due to electro-magnetic interference of other signals, and contributes 10~20% of overall powers.

Since the computational model has been evolved with synchronous domain, it is hard to ignore the clock cycle and go completely back to the asynchronous. However, with multicore architecture, the intermediate solution may exists. First we can somehow generate the local clock using Ring oscillators or programmable delay lines and let local component work synchronously. Next when it come to communicate with components that has different clock frequency or clock skew, they can communicate asynchronously. The overall system is called Globally-Asynchronous-Locally -Synchronous, GALS system.

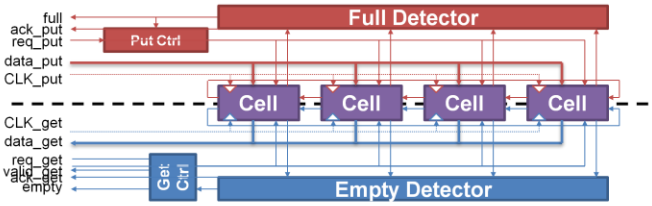
There are three well-known patterns of communications in GALS system. First one is pausible clocking. When it comes to communicating from module 1 to 2, first 1 saves the result to the output port and pause the clock for 1 to transmit the data. Once it is received by the input port of 2, the clock cycle for 2 is stopped to resolve the clock skew caused by async. transmission. Pausible clocking provides small area, low latency and low power consumption, the throughput really depend on the clock pause rate and it is typically slower than the other approach.

Next, by using asynchronous FIFOs that receives clocks from both modules and synchronize the clock timing inside, modules with two different clocks can be communicated. Even though async FIFOs typically have bigger area, high latency and high power consumption, they provide high throughput.

Lately, if we know how different the clock cycles is between two modules, we can manually place clockless delay between two modules. The approach is called boundary synchronization. Since the communication depend on the clockless delay, it is hard to implement, but provide relatively good performance for all criteria I listed.

Some issues:
1. How to generate local clock?
   Ring oscillators need careful calibration.
   Using programmable delay lines
2. GALS-based solutions don't automatically offer performance gains
3. How to test?
   functional test of async circuit is very difficult.
4. async-sync design flow is required in CAD tools.
5. Combining DVFS with GALS may provide great power reduction.
6. Advantages: Great EMI reduction

Robust Interfaces for Mixed-Timing Systems

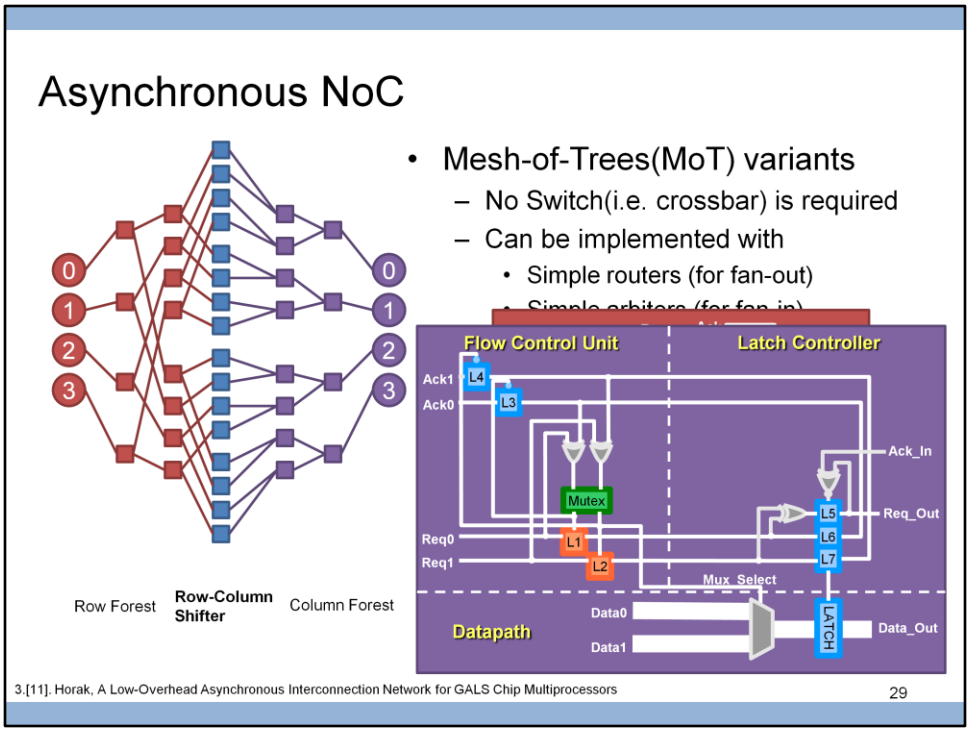


- Partition FIFOs into reusable components
  - Reusable Put and Get Cell sub-modules
  - Only Data Validity Controller sub-module needs to be modified
- Implement Relay Stations with Mixed Timing FIFO

3.[10]. Chelcea, Robust Interfaces for Mixed-Timing Systems

28

As a good example, Chelcea propose robust FIFOs for Mixed-timing system. The overall FIFO structure can be described as follows. Each cell maintains the head and tail information for FIFO implemented as a circular buffer. In the cell, the tail information and the register is maintained by putter (sender), which is clock cycle from the top part. On getter side (receiver), there is a clock-driven delays to synchronize the value from the output of the register, and the head information. In case of full or empty detection is carefully designed to avoid meta-stability. As I explained even the cell part can be almost separated into the components driven by the putter and the getter. Thus, those sub-modules can be reused to design, async-sync FIFO, Async-Async FIFO, or Sync-Async FIFO. Only Data Validity contolling sub-module in the cell need to be designed for each case. Chelcea also propose the async-sync or sync-async relay stations based on the FIFO implementations. The first challenge of designing system, having a long wire delays is solved by placing Relay Stations in Latency-Insensitive Design, this proposal try to solve the second question, the different local timing with mixed-timing relay stations.

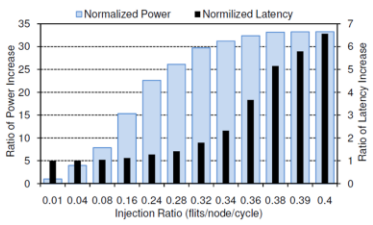


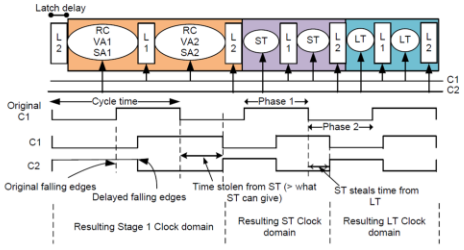
With the knowledge of GALS system we have seen so far, one may think that using a completely asynchronous NoC is also feasible. The asynchronous network on chip presented by Horak explains how we can build asynchronous network based on what we have seen. Basically we can use mixed-timing FIFOs as an interface between locally synchronous components and the asynchronous network.

Since designing asynchronous routers are quite complex procedure, we can simply use the topology that simplifies the design of routers. Horak uses one of Mesh-of-Trees(MoT) variants. Instead of having a dedicated node for the leave of tree, by having each node attached to a root of each tree, we only need to have a simple router that chooses where to go, and simple arbiters to select packets. This topology requires no switch or standard router microarchitecture we saw.

Dynamic Voltage-Frequency Scaling (DVFS)

- Power Consumption of NoC
  - up to 28% total power
  - Router frequency: critical design parameter
- Network power vs. network latency
- Dynamic power management for routers
  - Clock Scaling and Time Stealing





Mishra, A Case for Dynamic Frequency Tuning in On-Chip Networks

30

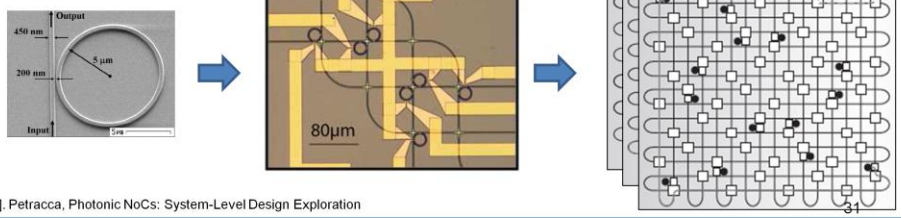
We can also use the dynamic voltage-frequency scaling for a router because the power consumption of NoC becomes significant. The recent implementation of NoCs already consumes 28% of total power and it gets worse as the injection ration increases. Since one of criticial design parameter for power is the router frequency, Mishra propose two different dynamic frequency scaling in his paper. The first one is clock scaling applied to a router. When a router is congested, it sends the signal back to the congested node to scaling down the frequency of the flow. When no other flows exists, to boost the traffic, a router also sends the signal to the current flow to scale up the frequency.

The second proposal of Mishra is Time Stealing. Since the critical path of router pipeline is virtual allocation, a router intentionally feed the clock with low frequency to the virtual allocation stage. However, for the other stage, a router also feed the faster clock. Thus, the figure looks like VA stage steals some parts of clocks from the other stages. This clock delaying can be implemented as inverter delay chain connected to a mux.

With the experiments, Mishra argues that their router design achieves 36% reductions in average latency and 13% of power savings.

Photonic NoCs

- The benefits of Photonic communication
  - Bandwidth
  - Power Dissipation
- Hybrid Photonic vs. electronic NoCs
  - Same execution time: 7.6W vs 244W
  - Same power dissipation: 960Gbps vs 100Gbps



3.[12]. Petracca, Photonic NoCs: System-Level Design Exploration

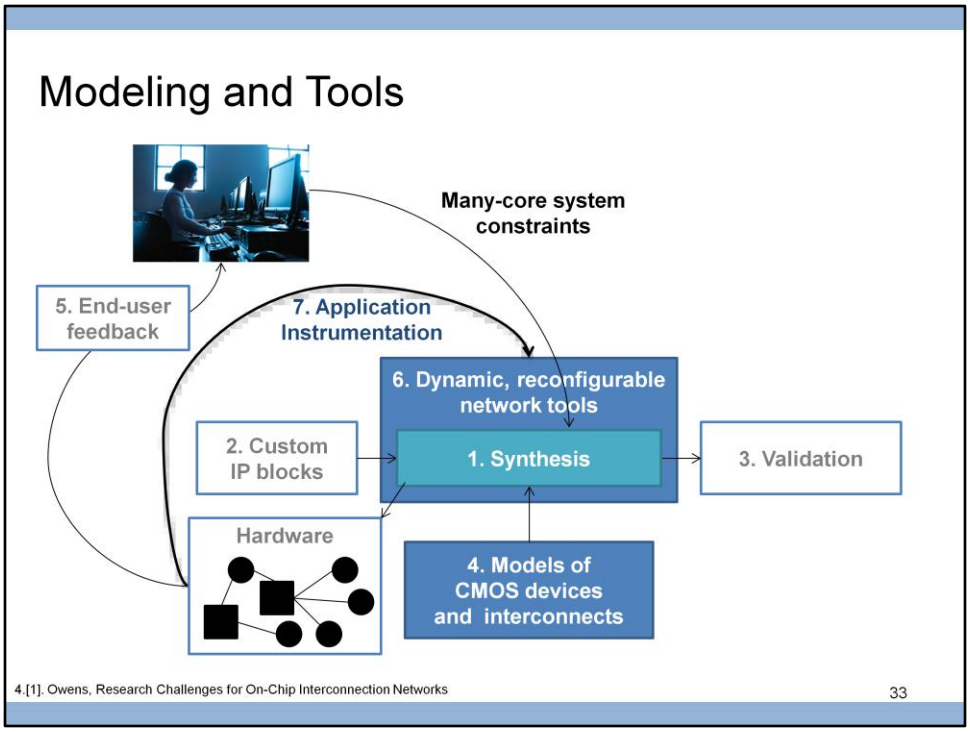
31

As new technology is evolved, sometimes there is a new opportunity. Photonic NoCs are one of good examples of it. There are benefits that differentiate the photonic communication from the electronic counterparts. First, using wavelength-division multiplexing, photonic communications provide large bandwidth. Second, by the physical property of light, it also dissipate less power, almost independent from the traveling distance.
However, there are disadvantages. First, due to the physical property again, it is hard to broadcast. And second, there is no optical RAM. The recent emergence of photonic NoCs come from the ring resonator implemented in Princeton. This device acts as a switching element with a relatively small circumference. Based on the resonator, Bergman's group in columbia shows the feasibility of building 2x2 non-blocking switch, and finally start talking about communication with those devices.

Since there is no optical RAM, Petracca proposed a hybrid Photonic NoCs based on circuit-switch network: the circuit is established by a small electronic network, and the massive data are transferred through established photonic circuits. He also argue that with the same execution time, photonic NoC saves large amount of power 7.6W vs. 244W, or for the same power dissipation, provides large bi-section bandwidth, 960Gbps vs 100Gbps.

Contents

- Motivation and Applications
- System Drivers
- On-Chip Communication and Networks-on-Chip
- Modeling and Tools

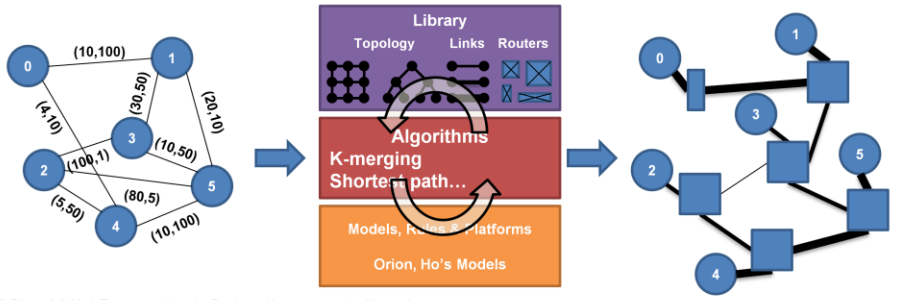


Before we go over the modeling and tools, I want to point out that there is another interesting paper that talks about the research issues related to NoCs. Owens present the paper called Research Challenges for On-Chip Interconnection Networks to summarize the discussion of the NSF workshop held in 2006. This paper emphasizes three issues: First, The large Power dissipation of NoC, Second, high Latency of NoC, and Third, difference between modern design flows and NoC Design. Even though all other papers address the problem, this paper put more emphasis on CAD tool integration. As a figure, the paper briefly explains the role of automation tools for NoCs. First we need synthesis framework, and IP blocks that is fed into the synthesis module. We also need a validation model. To measure the power or performance, we also need to have relatively accurate CMOS device and interconnect models. To measure the performance, we could run application directly, and the synthesis module should iterate to adjust the performance.

In this presentation, we will briefly go over the shaded box with bold arrows, which is simply put as: synthesis framework, power model, and possible simulation tools for designing NoCs.

COSI: NoC Design Automation

- Can we automate to design NoC?
- Communication Synthesis Infrastructure (COSI)
  - Network specification
  - Library of building blocks
  - Quantified performance and cost models
  - Optimization Algorithms



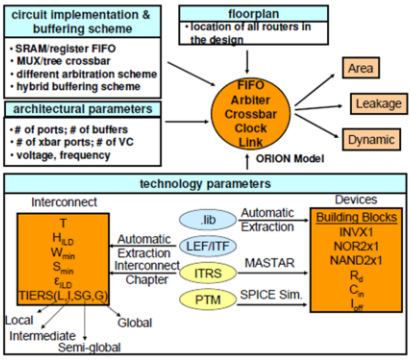
4.[3]. Pinto, COSI: A Framework for the Design of Interconnection Networks

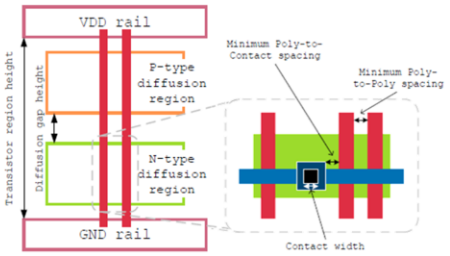
34

Pinto proposes Communication Synthesis Infrastructure, COSI, as a synthesis framework of NoCs. The input of framework is represented as a communication graph with design constraints such as latency and bandwidth. With a given library that contains topology, links, and router models, and the power or other constraint models such as Orion, the algorithm like K-merging iteratively compares the performance of different configurations and finds the best solution. The result is the actual implementation based on the communication graph.

ORION: NoC Power and Area Model

- Power: the most critical design constraint.
  - Power of NoC will also be substantial
  - How to estimate NoC power in the early-design stage?





4.[4]. Kahng, ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration

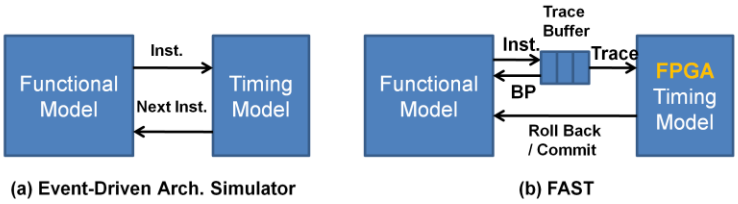
35

Kahng announced the Orion 2.0 for the NoC power and area model for early-stage design space exploration. By extracting the parameter from the layout of device building blocks such as inverter and nor gates, The model of basic building blocks of NoCs such as MUX or Tree crossbar is built. Based on the architectural parameters and abstract floorplans, Orion calculate area and both leakage and dynamic powers. Moreover, due to the simplicity, Orion can be easily intergrated into the architectural simulator or synthesis frameworks.

Dynamic Power consumption : P = E \dot f_{clk}, E = \frac{1}{2} \alpha C V_{dd}^{2}
Leakage Current consumption: I_{leak}(i,s) = W(i,s) \cdot (I'_{sub}(i,s) + I'_{gate}(i,s)),
W(i,s): the effective transistor width of component i at state s. I's can be measured by HSPICE simulations.

FAST: Architectural Simulation

- Good simulators
  - speed, accuracy, completeness, transparency
  - inexpensiveness, up-to-date, and easy-to-use, …
- The functional model of FAST
  - Keep generating instruction stream
  - Roll back when mis-speculations occur



(a) Event-Driven Arch. Simulator (b) FAST

4.[5]. Chiou, FPGA-Accelerated Simulation Technologies (FAST): Fast, Full-System, Cycle-Accurate Simulators 36

Sometimes it is necessary to run simulations to compare the performance of different NoC configurations in the early design stage. The good simulator should provide good speed, cycle-accurateness, complete in terms of including all system components, and should produce useful data as much as possible, called transparency. However, there is a trade-off between accuracy/completeness/transparency and speed.

To boost up the speed without sacrificing the latency, Chiou proposes FAST to place the timing model into the FPGA for better parallelization. Compare to the timing model-driven simulator, FAST decouples the dependency and let functional model execute without the control of timing model. When there is a mismatch between two models, timing models will invoke a rollback signal to regenerate the functional model. With detailed information, FAST could generate simulation result almost comparable to SESC. Since FAST has not been optimized yet, Chiou anticipates that FAST will provide further improvement in terms of speed without sacrificing the accuracy.

Speed vs cycle-accuracy:
SESC does not simulate wrong path, no backpressure in cache hierarchy (More speed, less accuracy)
GEMS is very slow. (More accuracy, less speed)

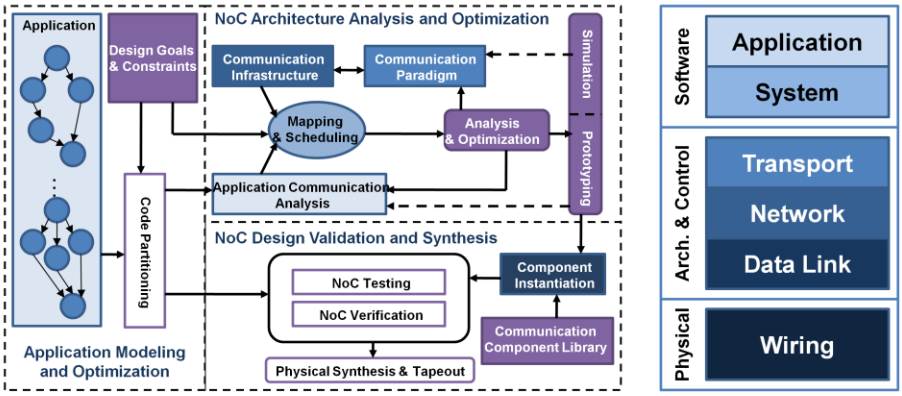
Complete vs. speedup:
SESC requires to modify applications and no OS support (More speed, less complete)
GEMS is very slow. (More complete, less speed)

Transparent vs. speedup:
GEMS produce more data than SESC, but it's slow (More transparent, less speed)

Why the accuracy is important? You can just use benchmarking or sampling (traces)
They do not reflect complex interactions between apps and the os.
They do not reflect the number of external events.
They do not reflect the potential performance impact of a rare event.
e.g. small differences on DRAM access ➔ different execution path caused by operating system ➔ Different performance result (5~10%)

Simulations of multi-core system: Get suffered due to the sequential timing model to be accurate.
Multi-threaded event-driven simulator: Only achieved 2~6x Speedup. It sometimes sacrifices accuracy.
FPGA-Based simulation: Exploit the parallelized nature of logics.
RAMP: Full-RTL level architectural simulator. Require many FPGAs and RTLs to simulate
FAST: Separate Functional and Timing models, and Timing model is in RTL.

Conclusion



1.[6]. Benini, Networks on Chips: A New SoC Paradigm
1.[7]. Marculescu, Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspective

37

I want to conclude the presentation by reusing two important figures. We covered what could be the application for multicore, and how the system driver of NoCs look like, and different research directions of NoCs with the mapping of system drivers. I also briefly covers the modeling and tools used to design NoCs. With these various research directions, I believe that the NoCs becomes more and more interesting topics, especially in multicore era for both generalized microprocessor and systems-on-chip.

Questions?


SO. HOW DID YOUR QUALS GO?
...
THAT BAD, HUH?
HOW? HOW CAN THEY JUDGE US CAPABLE OF 5 YEARS OF RESEARCH IN 12 MINUTES? WHAT NEXT??
I KNOW! NEXT YEAR, THEY ALSO HAVE TO PAT THEIR HEADS AND RUB THEIR STOMACHS WHILE JUMPING THROUGH HOOPS!
FACULTY MEETING DO NOT DISTURB
OR, WE COULD INCORPORATE A FITNESS TEST! THAT WAY, NOBODY WILL PASS!!
heh, heh...
JORGE CHAM ©THE STANFORD DAILY