Universidad Politécnica de Valencia

Redes en Chip

The Spidergon Network-on-Chip

Author:
Andrew Fecheyr Lippens

June 18, 2010

Abstract

As chip designers increasingly move to multicore solutions to improve performace and modularity of designs, self-timed packet-switched networks are poised to take a major role in addressing the complex system design and timing closure problems of future Systems-on-Chip. This paper highlights the benefits of Networks-on-Chip as a natural evolution of common bus-based communication and introduces the Spidergon STNoC interconnect technology by STMicroelectronics.

Contents

1 Introduction

A network-on-chip or NoC has become one of the key components in digital chips. As the number of components and transistors per chip grows continuously, most chips have become multi-core. These chips consist of a number of separate cores, often with fixed-function accelerators, memory arrays and input/output units.

Historically buses or crossbar switches were used to connect the functional components of a chip. Above four cores, however, the limited bandwidth of a bus becomes a major limitation, and above 8 or 16 cores, crossbar switches become very expensive. Only a NoC provides the scalability needed for emerging multicore chips. The NoC bounds the performance potential of these chips and accounts for a substantial fraction of the power and area consumption.

STMicroelectronics, a leading manufacturer in System-on-Chip (SoC) solutions, has developed a flexible and modular Network-on-Chip (NoC) called the Spidergon STNoC. This paper explores the features of this technology and focusses especially on the network implementation. Although it should only serve as a thorough introduction.

First a few technical terms are discussed in section 2.1. The evolution from common bus-based communication to network-on-chips is revealed in section 2.2. Finally, chapter 3 contains the elementary review of the Spidergon STNoC technology. For a more thorough analysis of the platform, Coppola's *"Design of Cost-Efficient Interconnect Processing Units: Spidergon STNoC"* [1] comes highly recommended. It also served as the basis for this paper.

2 Background

2.1 Technical Terms

This paper uses technical terms from the domain of computer networks. Some of them are explained here to help the reader.

System on Chip (SoC) refers to the integration of all components of a computer or electronic system into a single integrated circuit (chip). SoCs are often used in space and power constrained devices.

A Network on Chip (NoC) is a packet-switched on-chip micronetwork. It is the natural evolution of traditional circuit-switched bus solutions.

An Interconnect Processing Unit (IPU) is a configurable and extensible communication component that implements system services and core communication. It is essentially an on-chip communication network with hardware and software elements which jointly implement key functions of different SoC programming models through a set of communication and synchronization primitives and provide low-level platform services to enable advanced features in modern heterogeneous multi-core SoCs.[1]

A Chip multiprocessor (CMP) is a multi-core processor in which the distinct cores are integrated onto a single integrated circuit die.

IP block: In electronic design a semiconductor intellectual property core or IP block is a reusable unit of logic, cell, or chip layout design that is the intellectual property of one party. They are often used as building blocks in SoC designs.

2.2 Bus vs. Network-on-Chip

SoC communication has traditionally been done using on-chip busses. A diagram of an early bus architecture is depicted in figure 1. This allowed designers of early SoCs to select IP blocks, place them onto the silicon, and connect them together with a standard on-chip bus.

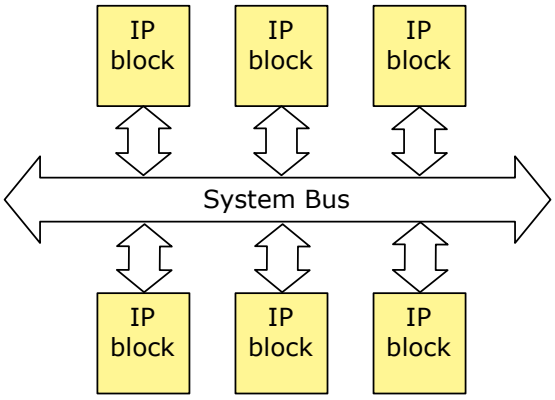

Figure 1: Early SoC structure based on a single shared bus

Buses are well understood and have been successfully implemented in many complex SoCs focusing on both application-based and processor-specific features, thus providing compatibility with most available IP and processor cores, and SoC communication requirements. Examples of more modern on-chip buses include ST Microelectronics' STBus [8], ARM's AMBA AHB and IBM's Core-connect [7].

However, buses do not scale well. With the rapid rise in the number of blocks to be connected and the increase in performance demands, today's SoCs cannot be built around a single bus. Instead, complex hierarchies of buses are sometimes used, with sophisticated protocols and multiple bridges between them. Communication between two remote blocks can go via several buses, and every section of every path must be carefully verified [5].

Moreover, the increase in the number of connected IP and processor cores causes the bus to become a communication bottleneck due to sharing of aggregate bandwidth by all attached units, and worsens arbiter delay, thus introducing variation, limiting scalability and making the arbiter instance-specific. Additional concerns regarding reachable clock frequency and time closure arise, since every bus transaction involves all connected cores and unnecessarily consumes significant power [1].

Where bus-based solutions reach their limit, packet-switched networks are expected to take over [4]. A packet switched network offers flexibility in topology (see Figure 2) and trade-offs in the allocation of resources to clients [5].

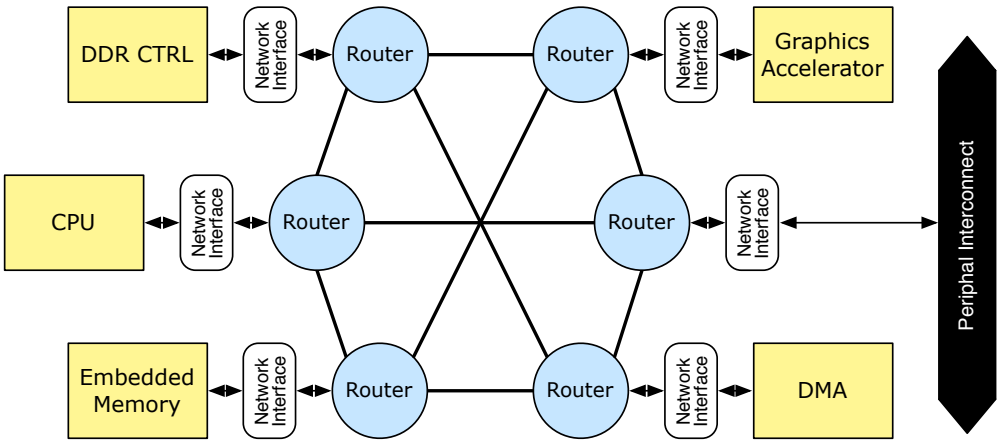

Figure 2: SoC equipped with a Network-on-Chip

As shown in Figure 2, on-chip interconnects based on point-to-point network topologies can provide scalability and power efficiency. In contrast to the bus view, on-chip interconnects are no longer passive components for plugging onto it several cores, but rather active components that provide connectivity and services to cores. Socket interfaces are no longer included in the cores, but are implemented in new components, called network interface units (NIs). This provides a clear de-coupling between the external core protocol and internal interconnect communication, while also strengthening modularity, composition and multi-protocol interoperability of connected cores.

Packet-switched networks-on-chip are the clear solution to the problem of complex SoC interconnects, and future developments will see advances in these networks to improve their performance, flexibility, power-efficiency and functionality [5]. Support for Quality-of-Service (QoS) protocols, fault-tolerance, secure communication and other similar high-level functions will emerge to establish the NoC as the defacto on-chip interconnect technology.

3 Spidergon STNoC

As a first attempt at designing a configurable Interconnect Processing Unit IPU, the Spidergon STNoC IPU technology consists of a flexible, pseudo-regular on-chip communication network implementing a set of customizable low-level platform services and a set of communication primitives.

The Spidergon STNoC IPU is a software programmable on-chip communication network that enables system designers to extend communication primitives. These communication primitive extensions are then automatically synthesized, placed, and routed into the Spidergon STNoC communication network. The communication network leverages graph properties of the Spidergon regular topology, while matching heterogeneity of Multicore SoCs through software programmability concepts, application-specific hardware configurability and extensibility.

Other NoC architectures are often based on a fixed regular network topology, or they are topology-independent, i.e. they can be customized to a particular application-specific communication graph. The Spidergon STNoC topology fills the gap between these two approaches, trading off regularity with customizability; for this reason, it is sometimes called a pseudo-regular topology [1]. Using this concept, NoC topology becomes an architectural parameter that can be configured depending on the communication patterns exhibited by the application. Moreover, in order to address Multicore application requirements for feature-rich devices, e.g. speech processing, video, GPS, security, and mobility, Spidergon STNoC technology provides a set of low-level platform services. The most important services defined in Spidergon STNoC are security, power management, and QoS. These services can be instantiated depending on the real target application and may be augmented by customer-specified services.

3.1 Spidergon Topology

The Spidergon topology provides an interesting price/performance trade-off for SoC devices. In the Spidergon topology, all of the IP blocks are arranged in a ring where each IP block is connected to its clockwise and its counter-clockwise neighbour and directly to its diagonal counterpart in the network. This allows the routing algorithm to minimise the number of nodes that a data packet has to traverse before reaching its destination. A particularly important advantage is that the functional diagram (shown below on the left of figure 3 for a network of 16 nodes) corresponds to a simple planar implementation (shown on the right side of figure 3) in which the wiring only needs to cross itself at one point. This is a key benefit in delivering high price/performance [9].

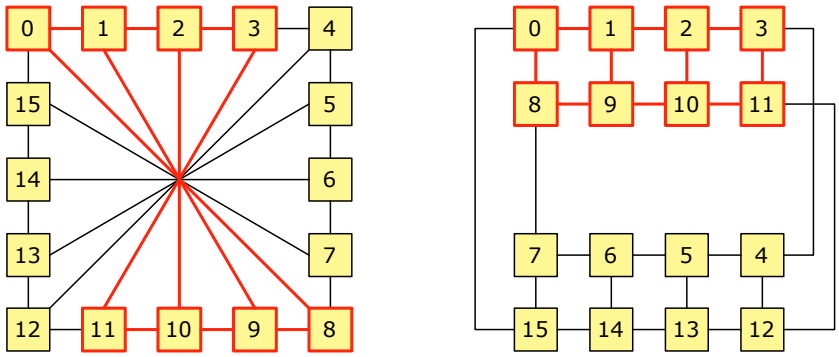

Figure 3: Equivalent representations of the Spidergon topology for $N = 16$

Topologies with increased connectivity, such as 2D mesh, provide very good theoretical metrics, but in most cases, these features cannot be fully exploited due to the nature of communication traffic in Multicore SoC applications. On the other hand, simple topologies, such as rings, are cost-effective in terms of manufacturing cost, but deliver relatively poor performance, especially as the number of connected cores increases. Spidergon tries to strike a balance in this tradeoff.

Compared to complex topologies, Spidergon offers a small number of links and simple implementation. For current, realistic NoC configurations with up to 60 nodes, the proposed Spidergon graph has a smaller number of edges and a competitive network diameter with respect to fat-tree or 2D mesh topologies.

However, Spidergon is flexible enough to support different topology configurations. Figure 4 illustrates different families of topologies supported by the Spidergon STNoC. These topologies are essentially degree 2 or 3 Spidergon subgraphs that range from rings and simple spanning trees to irregular chordal rings. Depending on application traffic requirements and especially mapping of master and

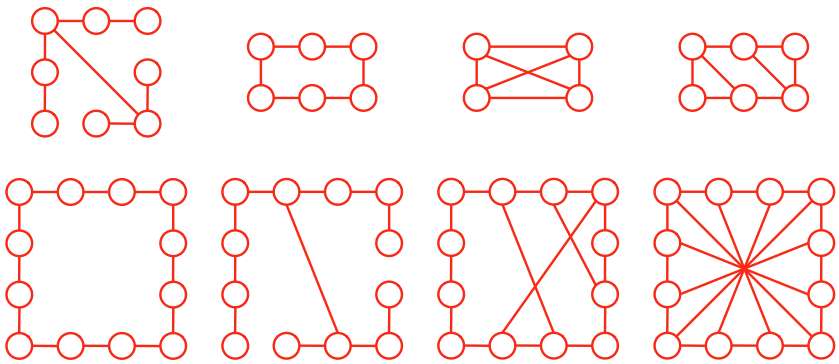

Figure 4: Different topologies supported by Spidergon STNoC

slave agents, connection paths can be removed if never used and cross connections customized to provide shortcuts between any pair of nodes in the ring.

3.2 Comparison with other Topologies

Each NoC topology offers a different set of tradeoffs in terms of metrics, such as:

- Vertex symmetry, which affects routing or scheduling cost and performance,
- Network degree, which affects the operating frequency and the complexity of the routers,
- Network extendibility, which should be as low as possible to maximize flexibility in Multicore designs, and
- Average distance between nodes.

Table 1 presents a summary of static topological metrics for three typical NoC topologies: Ring, Spidergon and $m \times n$ Mesh.

The data shows that for $N < 60$, the Spidergon topology, despite its constant bisection, is very competitive. Due to its higher connectivity, the Spidergon topology always outperforms Rings in terms of diameter and average distance. Spidergon also competes favorably or outperforms 2D Mesh, although outcomes depend on network size. In fact, while Spidergon properties scale linearly with the network size, 2D Mesh behavior is quite irregular. This irregularity is a severe bottleneck in Multicore SoC design, since it complicates design space exploration in terms of cost-performance tradeoffs. For example, a 22 node 2D mesh has smaller diameters and average distance metrics than a 24 node 2D mesh.

Property	**Ring**	**Spidergon**	**2D $n \times m$ Mesh**
Symmetry	both	vertex	no
Degree	2	3	4
Links	$2N$	$3N$	$4mn - 2m - 2n$
Extendibility	1	2	$\min(m, n) \geq 2$
Node connectivity	2	3	2
Diameter	$N/2$	$N/4$	$m + n - 2$
Average distance	if N even: $N/4$ if N odd: $(N^2 - 1)/4N$	if $N = 4n$: $(2n^2 + 2n - 1)/N$ if $N = 4n + 2$: $(2n^2 + 4n + 1)/N$	$\frac{(m+n)(mn-1)}{3mn}$
Bisection width	4	if $N = 4n$: 8 if $N = 4n + 2$: 10	if max(m,n) even: 2min(m,n) if max(m,n) odd: 2min(m,n) + 2

Table 1: Theoretical metrics for different topologies [1]

3.3 Switching Strategy

The switching strategy refers to how a packet traverses the route. More specifically, the switching technique determines how a message is fragmented and transmitted from an input of the on-chip network to an output resource by defining how channels and possible buffers along the path are allocated. There are four popular types of switching techniques:

- Circuit switching,
- Store-and-forward,
- Wormhole, and
- Virtual Cut-through.

Spidergon STNoC adopts wormhole routing, which is commonly used in NoC design. Due to this switching technique, the Spidergon STNoC router has a simple architecture, occupies a small area, and is extremely fast in terms of operating frequency.

In wormhole routing packets are split into flits (flow control units) which are sent along the path opened by the header flit in a pipelined way [3]; at the physical layer, each flit may also consist of several smaller transmission units (usually a few bytes long), called physical units, or phits. With wormhole routing, router resources are normally dedicated to the packet until all flits have been transferred. While both store-and-forward switching strategy and virtual cut-through use per packet flow control, i.e. only the header carries flow control information, with wormhole routing each flit has its unique flow control [3]. This concept drastically reduces the amount of network buffering to a few flits, since routers do not have to store entire packets. The minimum buffer depth actually depends on link-level flow control and the depth of packet pipelining. Furthermore, similar to virtual cut-through, transmission latency with low network congestion is almost independent of the distance between source and destination [2]. However, heavy network contention causes blocked packet flits to remain stored at the network interface and internal routers, which makes wormhole routing more susceptible to deadlock and congestion than virtual cut-through.

Transmission of different packets over a physical link cannot be interleaved or multiplexed freely without additional architecture support. In fact, if the header flit cannot proceed due to link contention, the whole worm of flits is stalled, occupying flit buffers in different routers on the currently constructed network path, while blocking other possible communications. Spidergon STNoC resolves this issue using virtual channels [6]. This requires extra control logic and separate buffers that allow independent packet flows sharing the same physical link to be time multiplexed at flit granularity.

In terms of link-level flow control, the system used in Spidergon STNoC is based on simplified credit mechanisms.

3.4 Layered Architecture

As shown in Figure 5, the Spidergon STNoC IPU has been developed using a vertical layered approach based on a simplified version of the ISO-OSI reference model composed of only 4 layers. All these layers represent independent components that can be examined either separately or in relation to each other.

- **The physical layer** provides electrical media definitions that establish connection at the bit level among on-chip network resources. This layer determines the number and thickness of electrical wires for connecting together resources, and depending on technology establishes a type of synchronization. It also defines the number of repeater circuits to cope with long wires

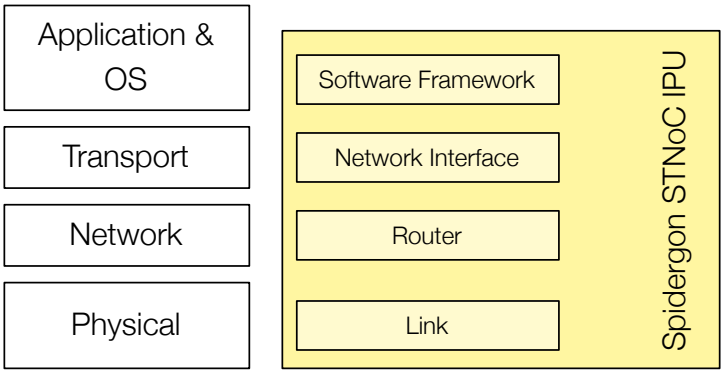

Figure 5: Spidergon STNoC Architectural Layers

and corresponding control signals. On top of physical wires and circuits, the Spidergon STNoC link establishes a logical connection between on-chip network resources, through a robust data transmission protocol, through flow control.

- **The network layer** is concerned with point-to-point packet routing from an arbitrary sender to an arbitrary receiver through the network topology. This layer implements wormhole routing and manages flit- and packet-level resource allocation and virtual channel and link arbitration.

- **The transport layer** hides network topology and link implementation issues, thus establishing efficient end-to-end flow control and source-to-target message communication services to higher layers, including operating system, system software, and application layers. In this layer, data are split or merged into NoC packets, and different processors or hardware blocks can be plugged as an open socket.

- Operating system, programming models, system software, middleware and application software run on interface wrappers of higher NoC communication layers built upon the transport layer as abstractions of the lower-level communication framework.

With a vertical layered stack approach, decisions taken at different layers are not independent and interact with each other through well-defined interfaces. In Spidergon STNoC design a lot of attention has been placed on how the four architectural layers fit together and which type of issues must be addressed at each layer for designing an efficient and unique IPU. To simplify design, each layer of the stack is implemented by a specific component. Thus, Spidergon STNoC IPU consists of three hardware modules called link, router and network interface, and

a software component (called software framework in Figure 5) that provides a set of communication primitives and low-level platform services.

3.5 Packet Format

The Spidergon STNoC network interface is defined so that multiple split transaction bus protocols could be encapsulated by the same network layer. Thus, the network layer payload consists of the overall transport layer packet, i.e. its header together with the payload. The network layer packet header contains an end-to-end QoS field and routing information. More specifically, routing information encodes a complete end-to-end path to arrive to the destination. In Spidergon STNoC, routing information is based on a destination address identifier and directional information. Exploiting the topological properties of Spidergon, the address identifier is 8 bits and the directional information is encoded in only 2 bits, independent of the number of nodes.

According to wormhole switching, Spidergon STNoC packets are split into one or more flits. The flit size is selected according to bandwidth requirements in the range of 16 to 512 bits; notice that flit sizes between 32 and 128 bits are the most appropriate for mapping requirements of most existing architectures. Depending on the packet and flit size, the network layer packet is decomposed into a variable number of flits, each characterized by a tag specifying if the flit is at the beginning, middle, or end of a packet.

3.6 Routing Algorithms

Spidergon STNoC adopts oblivious routing. Several alternative routing algorithms are proposed, so that the application user can select the best solution for a specific network topology instance and application traffic scenario.

The communication path is completely determined by the source and destination address and the route is fixed when packets enter in the network. The choice for deterministic routing guarantees in-order end-to-end communication at network level, hence avoiding costly flit reordering at packet reception.

Although Spidergon STNoC routing is not adaptive, i.e. selection of the routing path is fixed upon packet injection at the network boundary, it is programmable by supporting efficient software control of packet routes. A routing function executed at packet injection time associates a particular network path to the packet depending on the source and destination address. This function can be changed

during runtime through software reconfiguration, fully exploiting topological path redundancy.

Packet routing decisions are carried out using only local information available at each network node. Each router has a unique address i in the network, $0 \leq i < N$, where N is the network size. Since the routing algorithm is local, and the Spidergon topology is vertex- and edge-transitive, the routing algorithm is equivalent at any node.

When a router receives the first flit, i.e. the network header of a new packet, the forwarding path is computed. The Spidergon STNoC routing algorithm compares the network address of the current router to that of the destination router whos address is encoded in the packet header. If the two addresses match, flits are routed to the local port of the router. Otherwise, an attempt is made to forward the flit towards a clockwise or counter-clockwise direction along the ring, an across link, or in a hierarchical way to another instance of the Spidergon topology family.

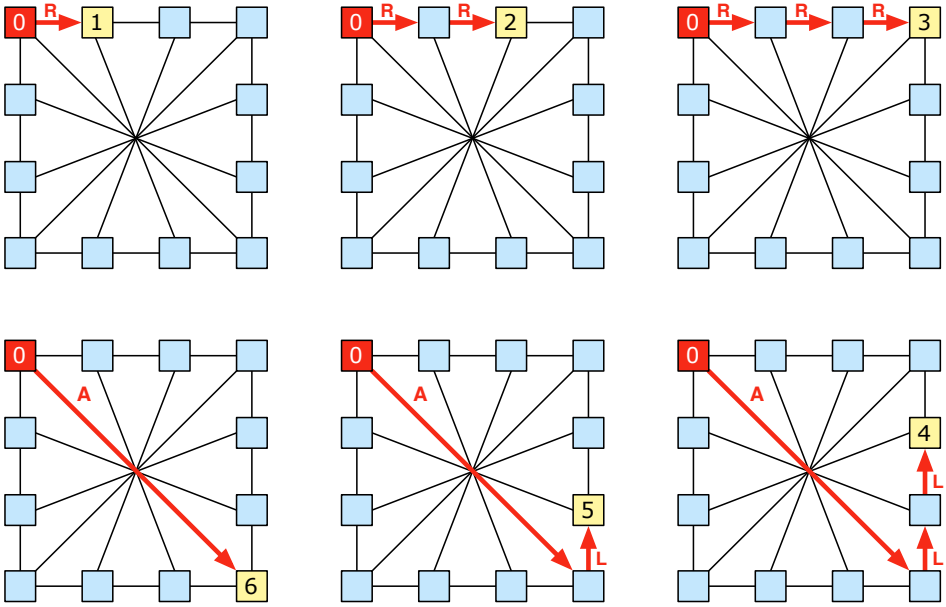

Figure 6: Across-First routing paths in the Spidergon NoC for $N = 12$

The first algorithm, called Across-First, moves packets along the ring, in the proper direction, to reach nodes which are closer to the source node, and uses the across link only once at the beginning for destinations that are far away. As an example, Figure 6 shows routing paths with the Across-First deterministic shortest-path Spidergon routing algorithm on a 12-node Spidergon STNoC. The clockwise direction is indicated with R (right), the counter-clockwise is L (left)

and the across link connection is A (across).

The Across-First routing relies on a simplified form of source-based routing, since the entire path can be encoded in the packet header using the following two observations. First, the Across communication link is selected at most once, always at the beginning of each packet route. Second, when the packet starts moving along the ring, it follows the same direction (right or left) for the entire path. According to these properties, only packets arriving from the local resource or from the across port need to be considered for routing. All other packets continue to move in the same direction until they reach their destination.

Across-First routing in Spidergon requires a fixed size of directional information (2 bits) that specifies the packet's path through the network: right, left or exploiting the across direction. For instance, in Figure 6, when the destination is 5, the first hop uses the across link and the direction field is set to "across-then-left" value, that indicates the cross as first hop and then left as ring direction. When destination is 6, direction is still "across-then-left", but left is not used since the packet arrives at destination before moving on the ring.

A obvious alternative to the Across-First routing is Across-Last routing, which is similar to Across-First but where the Across link is taken on the last hop. Both routing algorithms are shortest-path. However, Spidergon STNoC also allows routing in other ways. For instance, packets may progress along the ring without using the cross connections. In this case, the path is not optimal, but it could help distribute the load of request/response flows depending on the specific application requirements.

3.7 The Router

The Router component is responsible for implementing the network layer of the Spidergon STNoC protocol stack. It must ensure a reliable packet transfer through the on-chip network, according to a proper QoS policy. From a very high-level perspective, a router is based on a crossbar switch with a given number of input and output ports, and proper control logic. The core functionality of the router consists of deciding the output port where to forward the flit (after arbitrating possible conflict situations) and managing link-level flow control.

As shown in Figure 7, each Spidergon STNoC router implements wormhole routing and communicates through bidirectional links to a local NI, three neighbor routers in the right, left and across directions, and possibly another Spidergon STNoC network through a hierarchical connection.

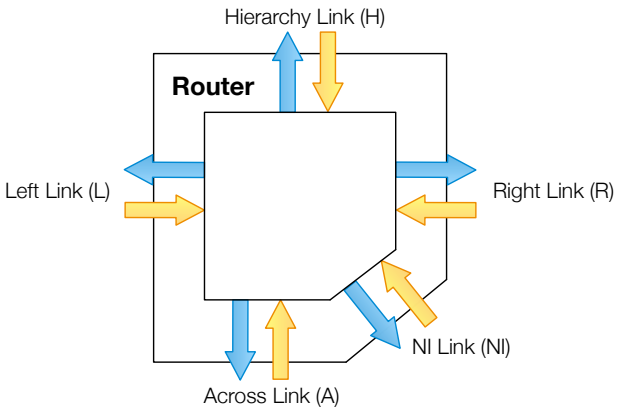

Figure 7: Spidergon STNoC router with upstream (blue) and downstream (yellow) interfaces

The Spidergon STNoC router is highly configurable, exposing its internal modular micro-architecture and allowing great flexibility in overall NoC design space exploration. A set of router parameters can be tuned at instantiation time to match functionality, performance and cost requirements.

A first parametric option in the router is the flit size for each input/output link. These flit sizes must be checked with respect to other connected routers and NIs, to guarantee correct configuration. When a link is not used at all it can be removed. The size of all buffer resources in the router is also configurable. This is especially important for low cost implementations when buffering can be selectively reduced depending on traffic type and performance requirements. Moreover, alternative queue techniques can be selected to increase global network performance, e.g. when lots of traffic is directed to a specific resource.

The Spidergon STNoC router has no externally accessible control registers. Its operation depends only on network configuration parameters defined upon instantiation and cannot be changed by software during runtime. This saves router area and programming complexity by avoiding switch initialization logic. Moreover, Spidergon STNoC services do not require any additional topological information since on-the-fly reconfiguration is performed efficiently at the boundary of the network. This implies simpler on-chip network design, while still enabling efficient on-the-fly tuning capability.

A low latency router is mandatory for ensuring fast communication. Spidergon STNoC router implementation proved to be a very costeffective solution, allowing clock frequencies up to 1GHz in 65nm ST technology, with a 2-cycle latency, and a per link bandwidth of 8GB/sec, assuming a 64-bit data path. Thus, the overall

bandwidth for the router is 40GB/sec. [1]

3.8 Buffer Management

The Spidergon STNoC router can use input and output FIFOs with the primary goal of having independent control over efficient link transmission and global network throughput, while at the same time minimizing buffer requirements. Since buffer capacity required for flow control is independent of that for absorbing suspended traffic at each router, different buffer management techniques are provided. An output FIFO can be allocated for each output port, reducing head-of-line blocking. These queues are shared among input flows to avoid costly time/space speedup factors. Multiple queues for each incoming flow can be eventually instantiated in case of high traffic to a local node to avoid many conflicts for packets that must be ejected from the network. FIFOs have bypass capabilities to reduce the pipeline, while an advanced output scheme supports virtual channels and at the same time offers a retiming capability.

3.9 Quality of Service

As far as bandwidth is concerned, Spidergon STNoC supports the Fair Bandwidth Allocator (FBA) QoS mechanism. It is an end-to-end service that guarantees fair and programmable weighted bandwidth allocation on top of a distributed network, just by tuning the injection points. Similar to routing, FBA is a two-step QoS scheme that provides all necessary information in the network header, limiting router behavior to simple operations. Configuration is provided through NI registers that are programmable by the software framework (see Figure 5) without impacting network routers. Thus, for example, routing can change without any effort by re-computing the path followed by a flow and corresponding QoS parameters along this new path. Moreover, FBA is cost efficient, i.e. the router implements a simple arbiter, without complex and slow logic. Finally, the adopted QoS scheme follows a unique structured approach, whose flexibility is achieved by simple software control, without changing the hardware implementation; the same scheme can degenerate into simple schemes, such as round-robin, LRU, or priority-based.

4 Conclusion

The Spidergon STNoC on-chip interconnect technology is a first attempt at designing an IPU for an advanced Multicore platform. At a first glance, the Spidergon STNoC IPU is a flexible on-chip communication platform based on a pseudo-regular network infrastructure implementing a set of communication primitives and programmable services for different applications.

We have briefly described the cornerstones of the network architecture, namely topology, switching strategy, packet structure, routing algorithm, buffer management, flow control and QoS support.

The design of the low-cost Spidergon STNoC architecture follows simple, clean and well-defined communication layering based on three hardware components: a physical communication link, a wormhole router, and a network interface to plug cores, hardware IPs and memories. The fourth module of the Spidergon STNoC platform is the software component which implements IPU services.

Spidergon STNoC leverages the regularity of Spidergon topology, while featuring practical topological customizations. Hence, Spidergon topology is a cost-efficient solution for the embedded SoC market, between a simple and low performance ring and a highly connected mesh.

References

[1] M. Coppola, M.D. Grammatikakis, R. Locatelli, G. Maruccia, and L. Pieralisi. *Design of Cost-Efficient Interconnect Processing Units: Spidergon STNoC*. CRC, 2008.

[2] WJ Dally. Performance analysis of k-ary n-cube interconnection networks. *IEEE Transactions on Computers*, pages 775–785, 1990.

[3] WJ Dally and CL Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on computers*, 100(36):547–553, 1987.

[4] WJ Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *Design Automation Conference, 2001. Proceedings*, pages 684–689, 2001.

[5] S. Furber and J. Bainbridge. Future trends in soc interconnect. In *System-on-Chip, 2005. Proceedings. 2005 International Symposium on*, pages 183–186, 2005.

[6] M. Moadeli, A. Shahrabi, W. Vanderbauwhede, and P. Maji. An analytical performance model for the spidergon noc with virtual channels. *Journal of Systems Architecture*, 2009.

[7] K.K. Ryu, E. Shin, and V.J. Mooney. A comparison of five different multiprocessor soc bus architectures. In *dsd*, page 0202. Published by the IEEE Computer Society, 2001.

[8] A. Scandurra, G. Falconeri, and B. Jego. Stbus communication system: concepts and definitions, internal document. *STMicroelectronics*, 2:2, 2002.

[9] STMicroelectronics. Stmicroelectronics unveils innovative network-on-chip technology for new system-on-chip interconnect paradigm. December 2005.