A GALS Infrastructure for a Massively Parallel Multiprocessor

Luis A. Plana, Steve B. Furber, Steve Temple, Mukaram Khan, Yebin Shi, Jian Wu, and Shufan Yang
University of Manchester

Editor's note
This case study focuses on a massively parallel multiprocessor for real-time simulation of billions of neurons. Every node of the design comprises 20 ARM9 cores, a memory interface, a multicast router, and two NoC structures for communicating between internal cores and the environment. The NoCs are asynchronous; the cores and RAM interfaces are synchronous. This GALS approach decouples clocking concerns for different parts of the die, leading to greater power efficiency.

—Michael Kishinevsky, Intel

THE SPINNAKER (SPIKING Neural Network Architecture) project at the University of Manchester aims at simulating a billion spiking neurons in real time. Fortunately, such an application is an ideal candidate for massive parallelism, and unlike some forms of parallel processing, it needn't maintain consistency in shared memories. Neural models running in such an environment communicate by means of spike events, which occur when a neuron is stimulated beyond a given threshold. The spike events must be communicated to all connected neurons, with typical fan-outs on the order of 1,000.

Figure 1 illustrates the basic Spinnaker architecture. Every node comprises a Spinnaker chip multiprocessor (CMP) and a memory chip. Six bidirectional links connect the nodes. The Spinnaker chip, which constitutes the basis of the system, houses several synchronous ARM9 processor cores, chosen primarily for their high power efficiency. Each processor models up to around 1,000 individual neurons, and a packet-switched network carries spike events to other processors on the same or other connected chips. At start-up, the processors perform a self-test; the first to complete the test successfully appoints itself the monitor processor and thereafter performs management tasks.

Each processor core has about 100 Kbytes of local memory on chip. As a supplement, a single external mobile double-data-rate SDRAM device of 128 Mbytes provides a large shared-memory resource used primarily for storing neural weights.

Each chip's six bidirectional links permit chip networks of various topologies. Interchip communication uses self-timed channels, which, although costly in terms of wires, are significantly more power efficient than synchronous links of similar bandwidth. We expect a flat 2D interconnect to suffice for the intended application, and this will allow straightforward layout on PCBs. However, this does not imply that the system can model only 2D neural structures. Spinnaker can model networks in two, three, or more dimensions. The key to this flexibility is that Spinnaker maps each neuron into a virtual address space. Assignments can be arbitrary, though assignments related to physical structure are likely to improve modeling efficiency. Neurons can be allocated to any processor, and the routing tables must be configured to send the neural events accordingly. Further details of the neural simulations are available elsewhere.[1]

Figure 2 shows a simplified block diagram of the Spinnaker chip. The prototype chips contain 20 ARM9 processor cores, each running at around 200 MHz. These cores must all communicate with the external SDRAM chip, clocked at 133 MHz. Another significant chip component is the multicast router, which is responsible for routing packets containing spike events between processor cores spread throughout the network. The router is also synchronous and is clocked at around 200 MHz. Providing an on-chip bus to connect all these devices at high speed is a challenge on a projected die size of 100 mm^2 and using 130-nm process technology. Timing closure would be a significant problem, and conventional synchronous buses would struggle to maintain adequate bandwidth when faced with connecting 20 bus masters. However, a globally asynchronous, locally synchronous (GALS) approach to the on-chip interconnect lets each synchronous block run in its own timing domain. The chip uses two distinct networks on chips (NoCs): The system NoC replaces a conventional on-chip bus for the system interconnect. The communications NoC, which includes an input section and an output section, provides an on- and off-chip packet-switching infrastructure. Both NoCs are based on Chain,[2] a delay-insensitive (DI) communication technology developed at the University of Manchester.

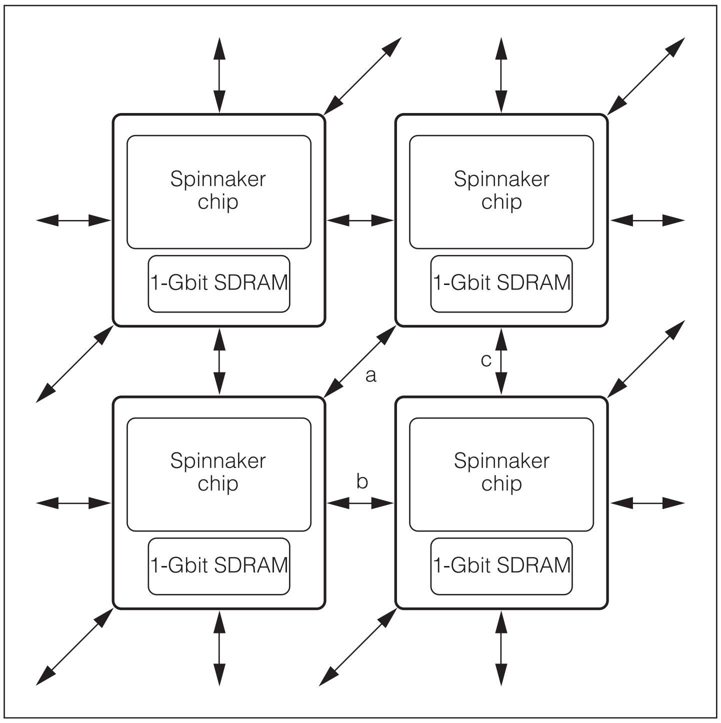


Figure 1. Spinnaker multiprocessor architecture.

Delay-insensitive communication

DI communication assumes nothing about the delays in the wires and gates that form the interconnect fabric except that they are finite and positive. For this reason, DI communication is more robust than styles whose operation is based on worst-case constraints—for example, synchronous communication. Furthermore, interconnect fabrics based on DI communication need no timing validation once they're designed, and they aren't constrained by layout timing issues.

Eliminating the delay assumptions requires extra information to be encoded within the data to communicate such timing issues as data validity. We do this by encoding the data within a DI code.[3]

The simplest practical class of DI code is the 1-of-*N* code. Here, *N* wires are used to encode *N* values, and at most one of the wires can have a value of 1 at any time. Table 1 shows a 1-of-4 code and its equivalent 2-bit binary code. To avoid illegal states, changes from one value to another must always

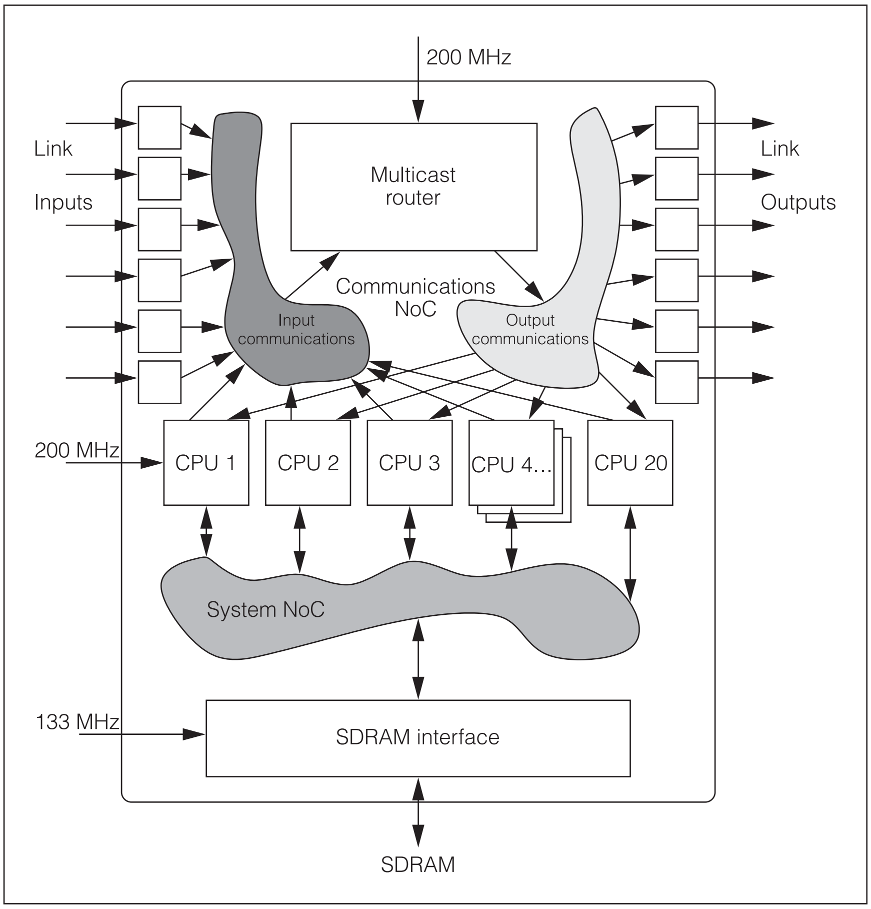


Figure 2. Spinnaker chip organization. (NoC: network on a chip.)

Table 1. Example 1-of-4 delay-insensitive (DI) code.

1-of-4 code				Binary equivalent
0	0	0	0	Null
0	0	0	1	00
0	0	1	0	01
0	1	0	0	10
1	0	0	0	11

go through the null code. This is known as a return-to-zero (RTZ) protocol and serves to guarantee that the receiver can always detect valid data correctly.

To complete a DI communication, the receiver of the data must be able to control how long the sender keeps the data stable. This is usually done with handshaking: The receiver uses an acknowledge signal to indicate that data has been accepted. The acknowledge signal also follows an RTZ protocol, as Figure 3a shows.

The null data tokens that alternate with valid tokens in the RTZ protocol can limit the maximum data rate achievable. These null tokens also affect power consumption, given that signal transitions are responsible for a large percentage of the power consumption of CMOS circuits. For these reasons, a non-return-to-zero (NRZ) protocol, shown in Figure 3b, can also be used. In this protocol, the code is represented by transitions in the wires and not by the actual state. Because NRZ DI codes are represented by signal transitions, performing logical operations and storing data can be very expensive. For this reason, NRZ codes serve mostly for data communication.

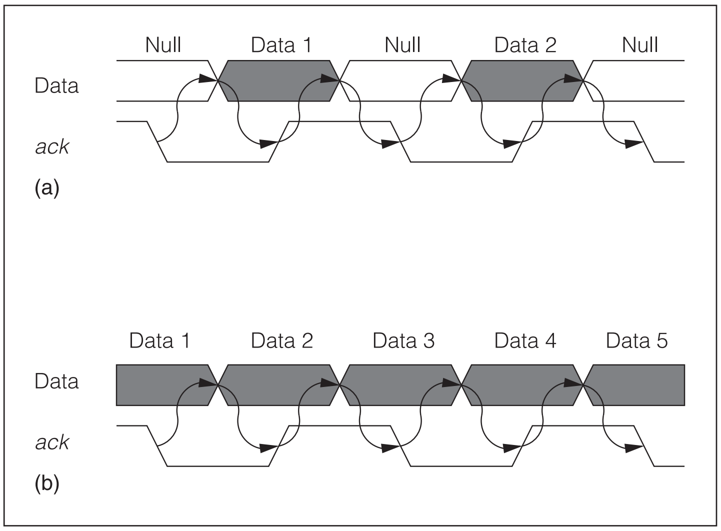


Figure 3. Delay-insensitive (DI) communication protocols: return-to-zero (RTZ) protocol (a) and non-return-to-zero (NRZ) protocol (b).

System NoC

The system NoC replaces a conventional on-chip bus, although in this case with more bus masters than usual. This NoC connects the 20 ARM9 cores and the router (the system masters that can initiate transactions on the NoC) to several slave devices, the most significant being the off-chip SDRAM.

We are implementing the system NoC using Chainworks,[4] a tool developed by Silistix to generate the self-timed on-chip interconnect. This tool generates standard Verilog netlists that can be integrated with the rest of the system and processed with standard CAD tools.

Figure 4 (with only four ARM cores, for simplification) shows how the system NoC appears in the Chainworks GUI. The master network adapters are on the left, and the slave network adapters are on the right. The adapters can provide a range of industry-standard interfaces to the external devices, facilitating the use of available IP blocks. The system NoC implementation uses standard AMBA[5] AXI interfaces, allowing seamless integration of the ARM cores, the SDRAM controller, and the rest of the system components.

Figure 4 shows that although routers are used for packet switching in the communications NoC, they are present in the system NoC as both system master and slave. The on-chip processor cores use the slave interface to configure the router—for example, to set routing tables. In principle, the router needn't start any transactions on the system NoC and should be a slave only. However, in the Spinnaker chip, it is a system master and lets processors in neighboring chips act as system masters. Those processors can send specially formatted messages through the communications NoC, and the router interprets those messages as requests to start transactions in its system NoC. The router automatically returns network responses to the requesting processor, also through the communications NoC. This mechanism serves as a system verification and debugging tool.

The interconnect fabric generated by Chainworks uses 1-of-5 Chain technology, based on 1-of-4 RTZ DI communication links, as described earlier, with an additional wire to encode an end-of-packet symbol. There are two parallel interconnect fabrics: one

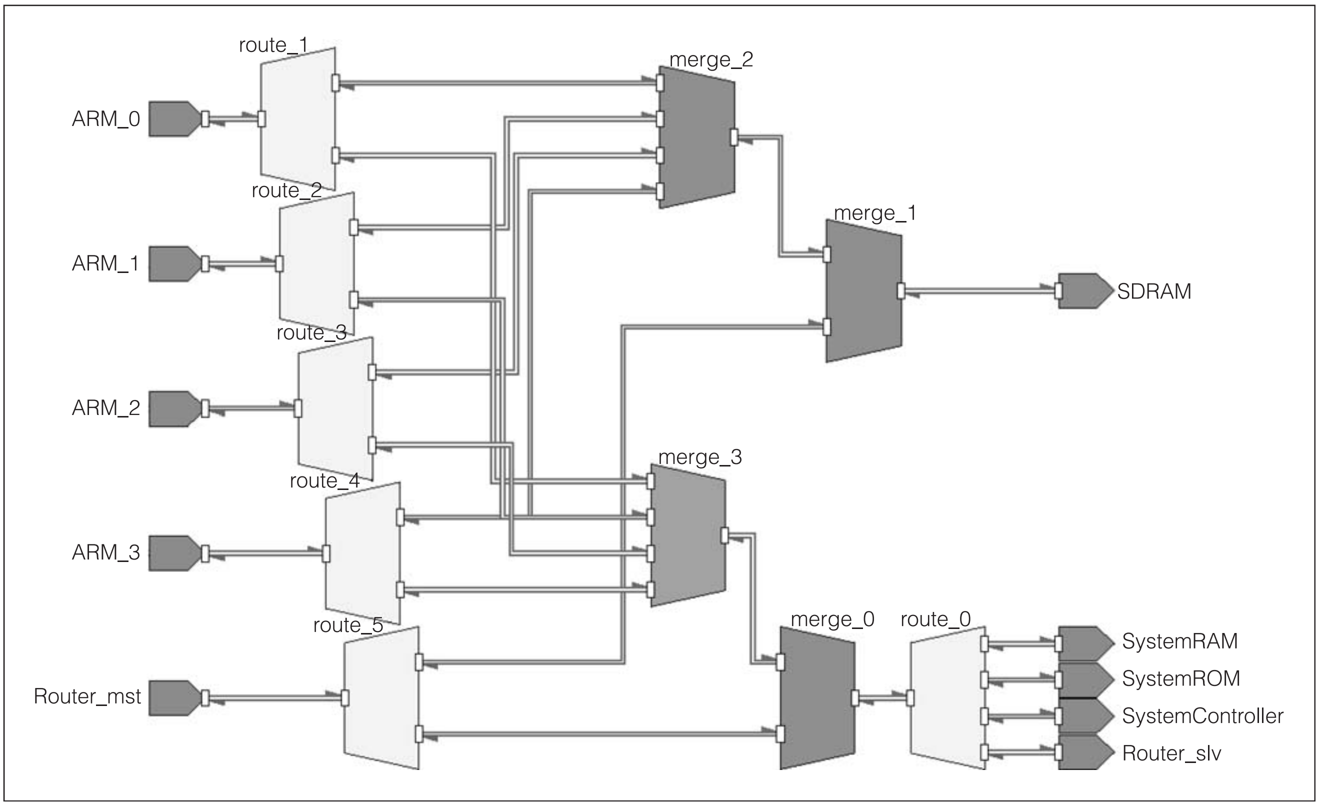


Figure 4. System NoC. The devices on the left (ARM_0 through Router_mst) are the master network adapters. The devices on the right (SDRAM, SystemRAM, SystemROM, SystemController, and Router_slv) are the slave network adapters.

transmits commands from the masters to the slaves, and the other transmits responses back from the slaves to the masters. Multiple DI links deployed in parallel deliver the throughput required by the devices in every part of the fabric, and long interconnects can be pipelined by inserting repeaters.

Using a NoC instead of a conventional synchronous bus offers additional benefits. Combining Chain route and merge components provides the desired fabric topology. The bandwidth available at the SDRAM interface is around 1 Gbps and must be fully utilized to achieve maximum efficiency in the neural modeling. To offload the processing task of transferring data to and from this memory, each processor core has a direct memory access controller dedicated to moving blocks of data to and from the SDRAM. The topology selected for the system NoC, although somewhat more expensive in area than a direct bus replacement, lets any system master communicate with the SDRAM while a different master communicates with any of the other system slaves. This is particularly relevant in our system, which has many system masters.

Communications NoC

The second network on a chip is the communications NoC, which provides the packet-switching fabric for the system. Its primary role is to carry neural-event packets between processors that can be located on the same or different chips. This network also transports system configuration and monitoring information.

The on-chip communications NoC, shown in Figure 5, divides into input and output sections. The former receives packets either from the off-chip links (the receivers, RX) or from the on-chip processors in the top of the figure and passes them to the router. The router determines each packet's destination and sends it via the output section of the communications NoC to the link outputs (the transmitters, TX) or the on-chip processors in the bottom of the figure. The router can replicate packets when necessary to implement the multicast function associated with sending the same neural-event packet to several destination neurons.

The communications NoC operates in a GALS fashion, with the synchronous router and local processor nodes interconnected through a 1-of-5

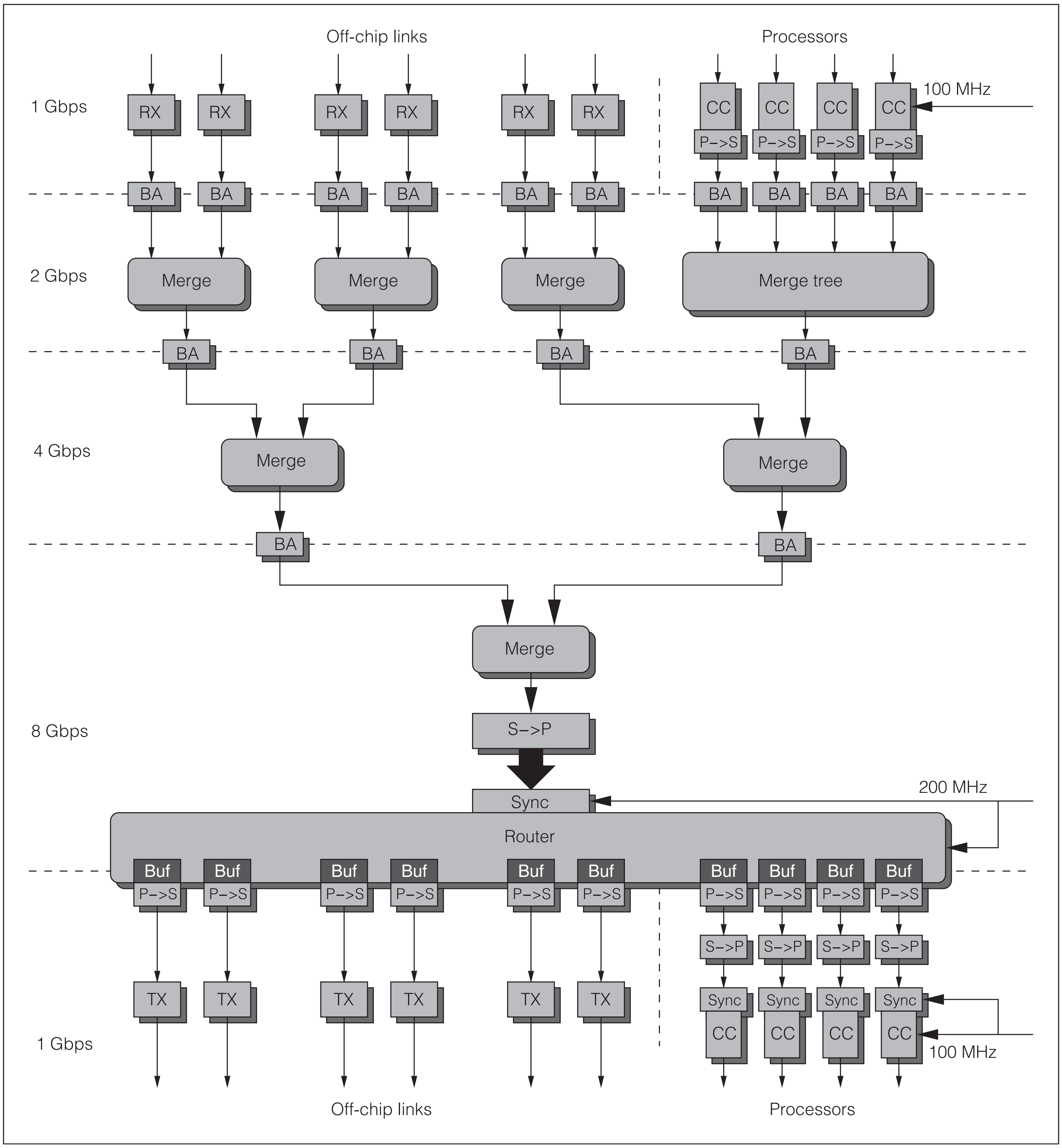


Figure 5. The communications NoC carries neural-event packets between processors. (BA: bandwidth aggregator; Buf: buffer; CC: communication controller; P –> S: parallel-to-serial conversion; RX: receiver; S –>P: serial-to-parallel conversion; Sync: synchronizer; TX: transmitter.)

Chain-link RTZ protocol fabric. The processors access the NoC through their communications controllers (CC), which operate at 100 MHz. The CCs are similar to universal asynchronous receiver-transmitters (UARTS) and serve to serialize (P –> S) and deserialize (S –> P) packets. As the input links converge on the router, they merge through two-way Chain arbiters, and the Chain-link width must increase to absorb the bandwidth. The processor links merge through a single-link arbiter tree, as the local bandwidth requirement is low—for example, 20 processors × 1,000 neurons/processor × 100 packets/(neurons · s) × 40 bits/packet = 80 Mbps. Each RX interface can carry up to 1 Gbps, about half the on-chip single-link bandwidth, so the first layer of the NoC can be a single Chain link. The second layer operates at 2 Gbps and can also be a single Chain link, whereas the third layer must be a dual link and the fourth layer a quad link (8 bits wide). Placing bandwidth aggregators (BAs) wherever the link width increases ensures that the full bandwidth capacity is used. The BA implementations use buffers that are at least half a packet long, and the output data width is twice that of the input. The buffers accumulate data until they fill up and only then trigger their output, thus guaranteeing that they can provide data at the maximum rate that the following merge can accept.

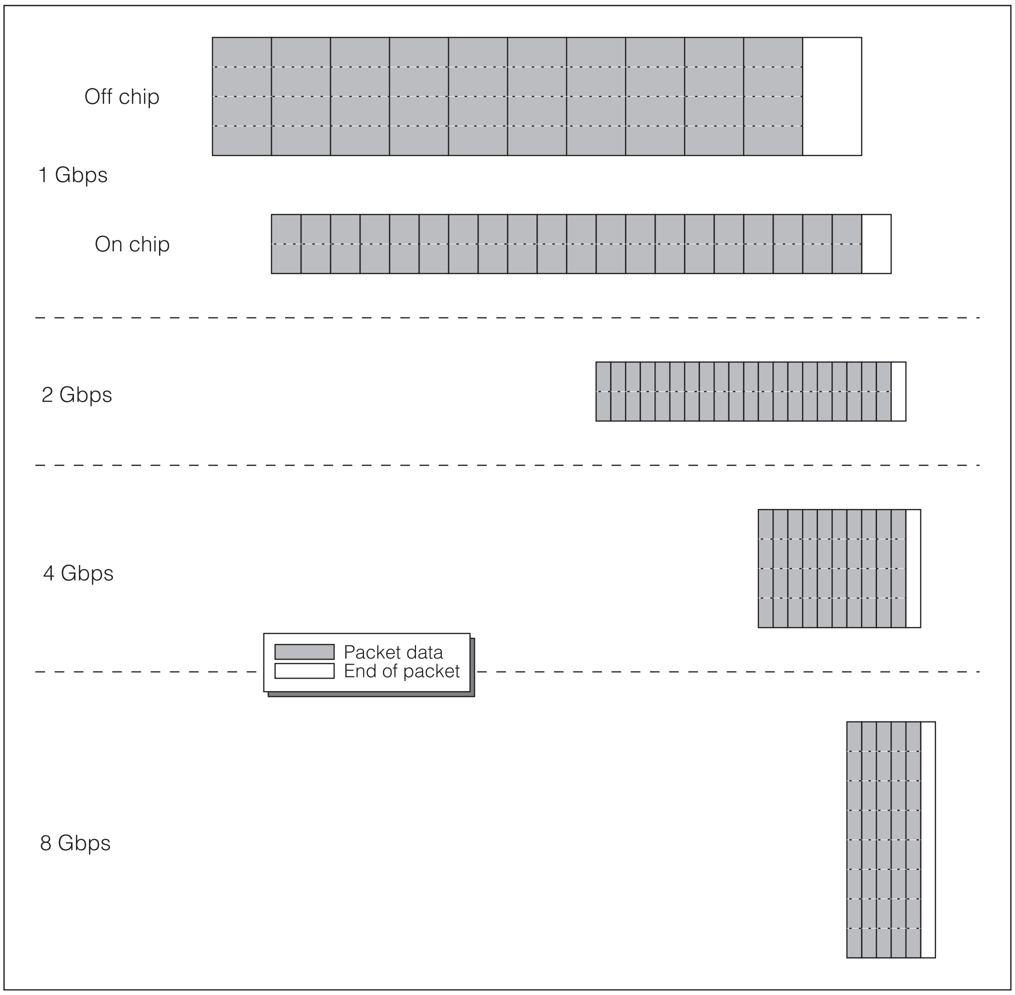


Figure 6. Changes in packet shape during communications NoC traversal.

Figure 6 shows how packets are transformed while traversing the communications NoC from an off-chip input link to the router. The first layer operates at about 1 Gbps. External links transmit packets in 4-bit flits, and RX interfaces transform each 4-bit flit into two successive 2-bit flits. BAs adapt the packets to the 2-Gbps bandwidth available in the second layer. The BAs buffer the first half of the packet and then send out the packet in 2-bit flits, twice as fast as the input flits arrive; this effectively doubles the bandwidth. This first layer achieves the maximum bandwidth provided by a single on-chip Chain link. Successive BA layers achieve the bandwidth-doubling effect by doubling the number of Chain links at the output of the BAs. Interestingly, the buffering in the BAs introduces very little latency. Although the front of the packets is effectively delayed at each layer, the packets are sent out through double-width links, so they take only half as long at each successive layer, and the end-of-packet symbol arrives at the router without a noticeable delay.

The last arbiter's output is deserialized (S –> P), so that a complete packet is presented in parallel to the router. The sync module synchronizes the asynchronous packet with the router clock before delivery. The router can process one 40-bit packet per (200 MHz) clock cycle, achieving maximum utilization of the 8-Gbps bandwidth provided by the input section of the communications NoC.

Router

The router is responsible for routing all packets that arrive at its input to one or more of its outputs. Primarily, it routes multicast neural-event packets,

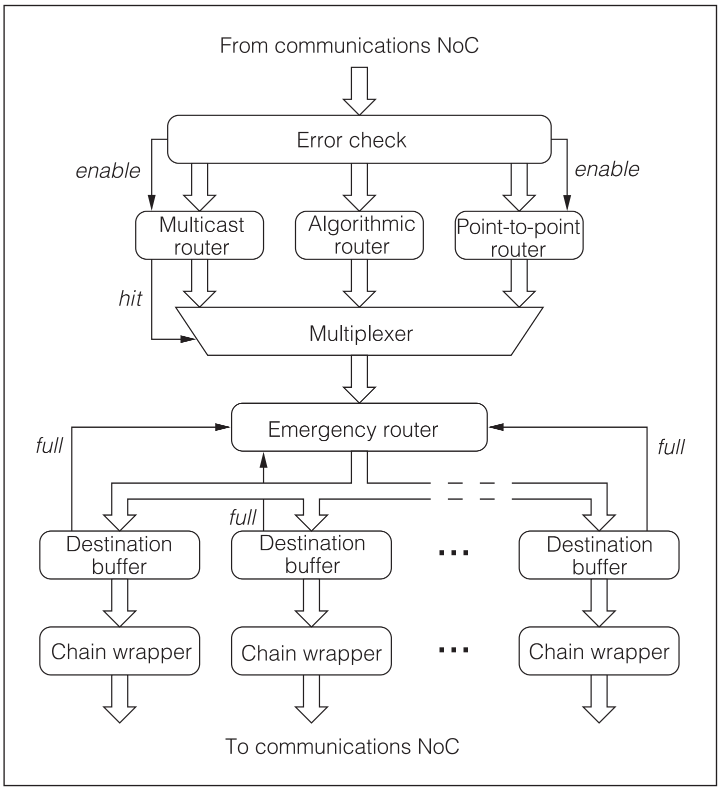


Figure 7. Router's three-stage pipeline organization.

using an associative routing table. It is also responsible for point-to-point packet routing, nearest-neighbor routing (a simple algorithmic process), default routing (when a multicast packet doesn't match any entry in the multicast router), and emergency routing (when an output link is blocked because of congestion or hardware failure). The router identifies and handles various error conditions, such as packet parity errors, time-out, and output link failure.

Figure 7 shows the router's internal organization. Packets arrive as single units from the input section of the communications NoC. In the synchronous, three-stage pipeline implementation, the first stage identifies any errors and steers the packet to an appropriate routing engine, depending on its type.

The second stage comprises three routing engines: multicast for neural-event routing, point-to-point for configuration and monitoring packets, and algorithmic for destinations that can be computed in flight (for example, multicast default routes or neighboring chips). The activated engine determines the destinations of the packets.

The third stage delivers the necessary number of copies of a packet to the destination outputs. When a failed or congested link results in a copy not being delivered, an emergency route automatically activates. As an example, assume that the link labeled "a" in Figure 1 is congested. Traffic that would normally use this failing link would be redirected, in hardware, to the two adjacent links, labeled "b" and "c" in the figure; these form a triangle with the failed link. This emergency routing is intended to be temporary, and if the problem persists, the operating system will identify a more permanent solution, which may involve changing the routing tables. The router informs the monitor processor of all uses of emergency routing.

The outgoing packets are serialized and sent to their destinations using single-width Chain links.

Link interfaces

DI communication is even more attractive for interchip interconnection. It lets data transfer occur at different speeds, which permits very flexible physical organization of the chips.

The Chain RTZ DI protocol is efficient for on-chip communication; however, the energy costs of an off-chip transition are high, and I/O pins are at a premium. As the communication system is extended to include interchip links, the trade-off between simplicity and power efficiency compels us to choose a different interchip protocol. Self-timed RTZ signaling incurs four chip-to-chip delays per symbol (the rising data transition, the rising acknowledge response, the falling data transition, and the falling acknowledge response), whereas an NRZ protocol incurs only two chip-to-chip delays per symbol. In addition, the code-mapping method can largely determine the complexity of the encoding, decoding, and completion detection circuits, so this method should be selected carefully. In the Spinnaker system, the interchip links use an 8-wire, DI 2-of-7 NRZ code with an NRZ acknowledge.[6] In this code, 16 of the 21 possible 2-of-7 combinations are used to encode four bits of data, and a 17th combination represents the end-of-packet symbol. When two CMPs are connected on the same circuit board, each link has half the data bandwidth of an on-chip link. When the CMPs are on different circuit boards, the self-timed protocol guarantees correct operation (albeit at a lower data rate). The communication links automatically adapt to the additional delays incurred by any signal buffering that may be required.

Figure 8 shows a block diagram of the chip interfaces.

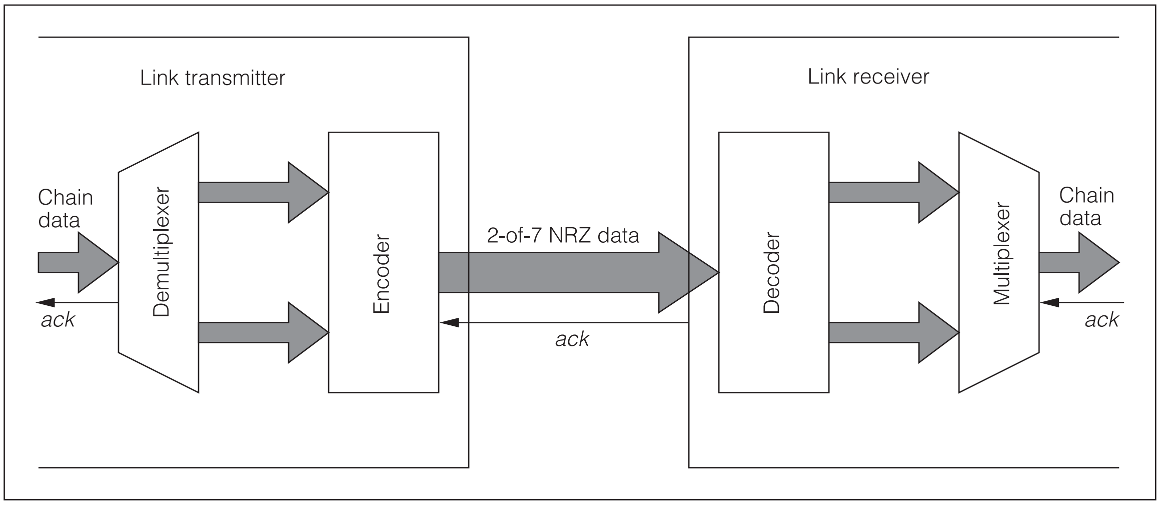


Figure 8. Interchip links. The link transmitter interface merges two successive Chain on-chip (1-of-5 RTZ) data flits into a single 2-of-7 NRZ interchip data flit and sends it to a neighbor chip's link receiver interface, which does the inverse conversion.

Simulation results using a 130-nm UMC cell library and chip-to-chip wires modeled with a 1.5-ns delay and 5-pF capacitance show that the 2-of-7 NRZ protocol's throughput exceeds 600 Mbps,[7] which is 3.6 times the maximum possible throughput when using the on-chip protocol for interchip communication. At the same time, every bit transferred by the 2-of-7 interfaces consumes only about a third of the energy of the Chain links. These results illustrate that the 2-of-7 NRZ links are more power and time efficient than the interfaces using the Chain protocol.

Fault tolerance

Any system designed on the scale of the Spinnaker billion-neuron simulator must incorporate some level of fault tolerance. The basic approach of the Spinnaker chip design is a combination of redundancy and reconfigurability.

The Spinnaker organization is clearly redundant. Each chip's 20 ARM cores are tested at start-up, and any failing processor can be disabled. Even if several processors fail, a Spinnaker chip is still highly functional. All processors have the same capabilities, and any specialized functions, such as system monitoring, are assigned after the processors' functionality has been established.

The system architecture allows easy reconfiguration if a chip fails. The neighboring chips' routing tables can be reconfigured to avoid the faulty chip, and the six interchip links provide the required redundancy for rerouting. This emergency rerouting is an automatic mechanism the routers use to avoid temporarily congested links. The mechanism is transparent to the application, but monitoring permits detection of permanently blocked links and implementation of corrective measures at the system level.

As noted earlier, the router can be driven from neighboring chips to act as a system NoC master, thereby permitting configuration of the chip devices from outside. A faulty chip can be probed, and corrective action can be taken accordingly.

Of particular interest is the behavior of the system interconnect in the presence of transient errors. The scaling down of feature sizes and processor technologies has made ICs more susceptible to factors such as alpha particles, cosmic radiation, crosstalk, and power bounce. This increased vulnerability usually manifests itself as undesired transient changes on wires, and these changes could break the Chain DI communication protocol. In particular, invalid codes could appear, or the acknowledge or end-of-packet signals could transition unexpectedly, leading to link faults or deadlock. Simulation results show that the Chain on-chip RTZ links will not deadlock in the presence of transient errors.[8] Glitches in the data or end-of-packet wires will generate wrong data symbols and may split a packet in two, but these errors will not stop communication on the link and can be detected by parity or CRC checks on the NoC's data link layer. However, the same simulations indicate that, if not designed carefully, the interchip NRZ links could deadlock in this type of noisy environment. The initial implementation of the pipeline registers used in the transmitter and receiver interfaces was susceptible to transient acknowledges that could stop data sampling. Fortunately, a more robust design of the register control circuit can avoid these deadlocks.

Power requirements

In a system on the scale of Spinnaker, power efficiency must be an engineering concern from the outset. On the basis of published figures of 0.12 mW/MHz to 0.23 mW/MHz consumption for an ARM968 core in 130-nm process technology,[9] each Spinnaker chip will consume 250 mW to 500 mW, enabling the chips to be deployed in low-cost packaging. The power requirement for communication is negligible, with each packet consuming 1 nJ for each router and 1 nJ for each interchip link it passes through. A large-scale system that can simulate a billion spiking neurons in real time will require 50,000 nodes and consume 23 kW to 36 kW.

System modeling and simulation

Modeling and simulating a system the size of Spinnaker proved challenging. We developed a SystemC-based systemwide Spinnaker transaction-level model as an efficient way to explore the design space and provide initial hardware prototypes. We refined the model to provide a platform for early software development. Further refinement let us use the model to generate vectors for verification and testing of the different system modules.

MUCH IS STILL UNKNOWN ABOUT the operation of the human brain. In the quest to understand the dynamics of neural systems, the Spinnaker multiprocessor, based on a highly parallel configuration of small, power-efficient processors and a GALS approach to on-chip and interchip interconnects, provides a new tool for the simulation of large-scale systems of spiking neurons. Researchers in this field proceed with the hope that exploring complex, event-driven systems will yield new insights into the biology of the brain and also into novel computational systems. ■

Acknowledgments

The Spinnaker project is supported by the Engineering and Physical Sciences Research Council, partly through the Advanced Processor Technologies Portfolio Partnership at the University of Manchester, and also by ARM and Silistix. Steve Furber holds a Royal Society-Wolfson Research Merit Award. We appreciate the support of these sponsors and industrial partners and also wish to acknowledge the constructive suggestions for improvement offered by the anonymous referees who reviewed this article prior to publication.

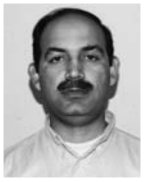
■ References

1. S. Furber and S. Temple, "Neural Systems Engineering," *J. Royal Society Interface*, vol. 4, no. 13, Apr. 2007, pp. 193-206.
2. J. Bainbridge and S. Furber, "Chain: A Delay-Insensitive Chip Area Interconnect," *IEEE Micro*, vol. 22, no. 5, Sept.-Oct. 2002, pp. 16-23.
3. T. Verhoeff, "Delay-Insensitive Codes—An Overview," *Distributed Computing*, vol. 3, no. 1, Mar. 1988, pp. 1-8.
4. *Silistix Self-Timed Interconnect Technology*, Silistix; http://www.silistix.com/technology_silistix.php.
5. *Advanced Microcontroller Bus Architecture (AMBA) Specification*, Rev. 2.0, ARM, May 1999, http://www.arm.com/products/solutions/AMBAHomePage.html.
6. W.J. Bainbridge et al., "Delay-Insensitive, Point-to-Point Interconnect Using M-of-N Codes," *Proc. 9th IEEE Int'l Symp. Asynchronous Circuits and Systems* (ASYNC 03), IEEE CS Press, 2003, pp. 132-140.
7. J. Wu and S. Furber, "Delay Insensitive Chip-to-Chip Interconnect Using Incomplete 2-of-7 NRZ Data Encoding," *Proc. 18th UK Asynchronous Forum*, University of Newcastle upon Tyne, 2006, pp. 16-19, http://async.org.uk/ukasyncforum18/.
8. Y. Shi and S. Furber, "Error Checking and Resetting Mechanisms for Asynchronous Interconnect," *Proc. 18th UK Asynchronous Forum*, University of Newcastle upon Tyne, 2006, pp. 24-27, http://async.org.uk/ukasyncforum18/.
9. *ARM968E-S*, ARM, http://www.arm.com/products/CPUs/ARM968E-S.html.

Luis A. Plana is a research fellow in the School of Computer Science at the University of Manchester, UK. His research interests include the design and synthesis of asynchronous, embedded, and GALS systems. Plana has a PhD in computer Science from Columbia University. He is a member of the IEEE.

Steve B. Furber is the ICL Professor of Computer Engineering in the School of Computer Science at the University of Manchester. His research interests include neural-systems engineering, on-chip interconnects and GALS design, and asynchronous systems. Furber has a PhD in aerodynamics from the University of Cambridge, UK. He is a Fellow of the IEEE, the Royal Society, the Royal Academy of Engineering, the British Computer Society, and the Institution of Engineering and Technology, and he is a Chartered Engineer.

Steve Temple is a research fellow in the School of Computer Science at the University of Manchester. His research interests include self-timed logic, VLSI design, and microprocessor system design. Temple has a PhD in computer science from the University of Cambridge.



Mukaram Khan is a software engineer and a PhD candidate in the School of Computer Science at the University of Manchester. His research interests include the bootstrap procedure, system configuration, and system-level management of the Spinnaker system. Khan has an MPhil in mobile transaction security from the University of Manchester.

Yebin Shi is a PhD candidate in the School of Computer Science at the University of Manchester. His research interests include digital-circuit design, fault tolerance techniques on asynchronous interconnects, and asynchronous SoCs. Shi has an MA in electronics engineering from the University of Science and Technology of China.

Jian Wu is a PhD candidate in the School of Computer Science at the University of Manchester. His research interests include designing communications routers and asynchronous interconnects for large-scale neural-network chip multiprocessors. Wu has an MA in electronics engineering from Beijing Institute of Technology.

Shufan Yang is a PhD candidate in the School of Computer Science at the University of Manchester, and a lecturer in the School of Computer and Communication at Hunan University. Her research interests include the analysis and optimization of the performance of asynchronous NoCs. Yang has an MA in computer science from Hunan University in China.

■ Direct questions and comments about this article to Luis Plana, University of Manchester, School of Computer Science, Advanced Processor Technologies Group, Room IT303, Oxford Road, Manchester M13 9PL, UK; luis.planacabrera@manchester.ac.uk.