An On-Chip and Inter-Chip Communications Network for the SpiNNaker Massively-Parallel Neural Net Simulator

Luis A. Plana[1], John Bainbridge[2], Steve Furber[1], Sean Salisbury[2], Yebin Shi[1] and Jian Wu[1]
[1]The University of Manchester, [2]Silistix Ltd.
{luis.plana, steve.furber, jian.wu, yebin.shi}@manchester.ac.uk, {wjb, sjs}@silistix.com

The real-time modelling of large systems of spiking neurons is very demanding in terms of processing power and communication throughput. SpiNNaker [1] is a scalable, multi-chip system designed for this purpose with an efficient multicast communications infrastructure inspired by neurobiology. SpiNNaker uses a GALS packet-switched network to emulate the very high connectivity of biological systems. The packets represent neural spikes and are source-routed, i.e., they only carry information about the issuing neuron and the network infrastructure is responsible for delivering them to their destinations.

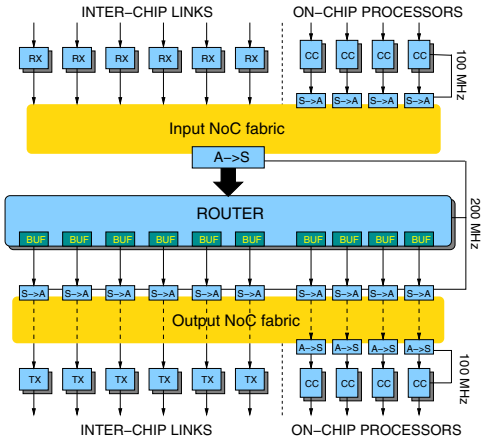

Figure 1. Communications Network-on-Chip.

The network is shown in Figure 1. The input section receives packets from other chips, through the inter-chip input links, and from the on-chip processors, and passes them to the router. The router determines the destination of each packet and sends it via the output section to the inter-chip link outputs and the on-chip processors. The router is able to replicate packets where necessary to implement the multicast function associated with sending the same neural event packet to several destination neurons. The NoC operates in a GALS fashion, with the synchronous router and on-chip processors connected through a self-timed fabric.

Each packet contains an identifier that is used to determine which output(s) the packet should be sent to. In neural applications, the identifier is a number that identifies the neuron that generated the message.

The router operates as a look-up table composed of two parts: an associative (content-addressed) memory responsible for comparing simultaneously the identifier of the packet with the all keys in the table and a conventional look-up RAM that holds the output routing words. If a match is found, the associative memory generates a hit-address for the look-up RAM. To reduce the size of the associative memory, neurons can be associated in groups which are routed using the same table entry. This is accomplished by masking some of the identifier bits in the look-up process, as shown in Figure 2.

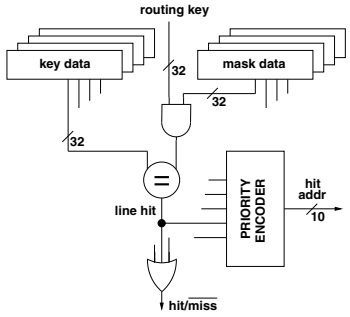

Figure 2. Masked Associative Memory.

In addition to the look-up table, a default routing mechanism is used. If no match is found in the table the packet is transferred from the input link to the output link directly opposite. In neural network applications, default routing can significantly reduce the size of the routing tables.

The router, shown in Figure 3, is implemented as a pipeline to sustain the input bandwidth. A back-pressure mechanism is used to selectively stop pipeline stages when a required output link is blocked or congested. This is also used as a power-saving mechanism. To minimize the im-

978-0-7695-3098-7/08 $25.00 © 2008 IEEE
DOI 10.1109/NOCS.2008.30

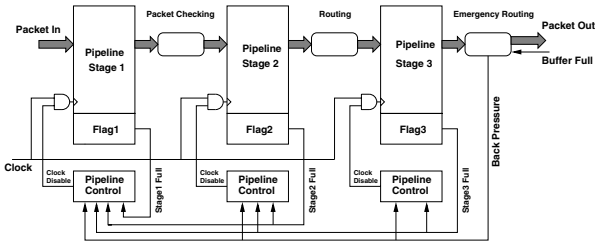

Figure 3. Router Back-Pressure.

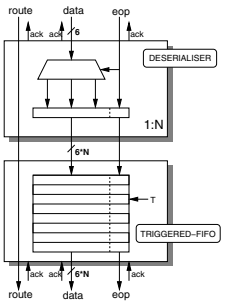

Figure 5. Bandwidth Aggregator.

pact of temporary network congestion as well as to simulate the fault tolerance of real neural networks, an emergency routing mechanism is implemented. When an output link is congested or broken the router will stop sending packets through that link and automatically redirect them through predefined alternative links. The packet is flagged as emergency-routed so that the receiving routers can route it appropriately.

The network fabric is implemented using delay-insensitive CHAIN technology [2]. Communication between the synchronous islands and the asynchronous fabric requires synchronization. Figure 4 shows a block diagram of a sync-to-async parallel path synchronizer, based on the well-known two flop synchronizer.

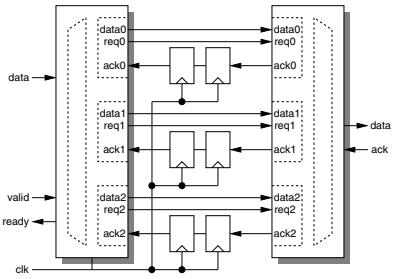

Figure 4. Parallel Path Synchronizer.

The synchronizer latches data internally and operates as a pipeline. The parallel paths (data0, data1 and data2) are used to allow the sending of data on every clock cycle of the sender, without incurring the 2-cycle latency of the synchronizer. An async-to-sync parallel path synchronizer is implemented in a similar way.

The traffic provided by the inter-chip links and on-chip processor must be delivered to the router without excessive latency and with a fair distribution of the bandwidth. The input fabric uses a 'bandwidth aggregation' scheme to allow full-bandwidth traffic to flow from every input and reach the router without any noticeable delay.

Bandwidth aggregators are implemented as shown in Figure 5. The first stage consists of a 1:N deserialiser. This stage transforms an input link of width 1 into an output link of width N buy steering successive flits to different output sub-links and issuing them as a single output flit. The second stage consists of a triggered FIFO. The flits are stored in the FIFO until a threshold level is reached. At this point, the FIFO starts to send the flits through its output link. If the threshold is chosen correctly, the output flits will be issued at the maximum rate possible in the output link, without pause and without holding the fabric components longer than required.

Delay-insensitive communication is also attractive for inter-chip interconnect. With a suitable protocol, each inter-chip link matches the data bandwidth of an on-chip link when connecting two CMPs on the same circuit board, and the self-timed protocol guarantees correct operation (albeit at a lower data rate) when the CMPs are on different circuit boards, automatically adapting to the additional delays incurred by any signal buffering that may be required.

Detailed Verilog simulations show that the network-on-chip design effectively optimises the use of the available bandwidth and delivers one packet per clock cycle to the router. The simulations also show that, after an initial setup period during which input packets may be delayed, packets can also be input without pause and at the maximum input rate.

Acknowledgements

The Spinnaker project is supported by the Engineering and Physical Sciences Research Council, UK, and also by ARM and Silistix.

References

[1] L.A. Plana, S.B. Furber, S. Temple, M. Khan, Y. Shi, J. Wu, and S. Yang. A GALS Infrastructure for a Massively Parallel Multiprocessor. *IEEE Design & Test of Computers*, 24(5):454–463, Sept.-Oct. 2007.

[2] John Bainbridge and Steve Furber. CHAIN: A Delay-Insensitive Chip Area Interconnect. *IEEE Micro*, 22(5):16–23, Sept.-Oct. 2002.