RecoNoC: a Reconfigurable Network-on-Chip

Robbe Vancayseele, Brahim Al Farisi, Wim Heirman, Karel Bruneel and Dirk Stroobandt
Ghent University, ELIS Department
Sint-Pietersnieuwstraat 41, 9000 Gent, Belgium
Brahim.AlFarisi@UGent.be

***Abstract*—This article presents the design of RecoNoC: a compact, highly flexible FPGA-based network-on-chip (NoC), that can be easily adapted for various experiments. In this work, we enhanced this NoC with dynamically reconfigurable shortcuts. These can be used to alter the NoC's topology to adapt to the system's communication needs. The design has been implemented and tested on a Xilinx Virtex-2 Pro FPGA, using the TMAP dynamic datafolding toolflow to automatically generate the reconfigurable hardware and the software reconfiguration procedures. The results show that, using dynamic datafolding, the overhead of introducing this shortcut mechanism is limited.**

***Keywords*-FPGA; Run-time Reconfiguration; Network-on-Chip; TMAP; dynamic data folding;**

I. Introduction

Advances in VLSI technology allow us to integrate more processing elements on one chip, creating ever more powerful and diverse systems-on-chip (SoC). Most of these elements are available as IP cores, so the main problem for many designers today is connecting them in an efficient way, maximizing performance and minimizing area and power costs. Since the concept has been introduced in [1], network-on-chip (NoC) architectures have emerged as the most promising solutions to this problem. The distributed nature of a NoC allows it to transport many parallel data streams, making it very scalable and performant.

If an application's network usage patterns are known a priori, the physical topology of the network can be designed to match the logical topology of the application and meet all delay and bandwidth requirements, without using more resources than strictly necessary. In many cases however, the communication patterns are highly dependent on the system's inputs or the system must be suited to run many diverse applications. In these cases a network using a generic topology and very wide links to provide sufficient bandwidth for the worst-case scenario is the only option. This takes up a lot of resources, most of which are rarely used.

In this work, we attempt to use the dynamic reconfigurability of SRAM-based FPGAs to make more efficient use of the available resources. We use a generic NoC design, but with a quick dynamic reconfiguration procedure, we can add *shortcuts* to our network. These shortcuts allow data streams to travel between distant nodes, using their own dedicated link and not passing through intermediate buffers or routers. Once the shortcut is no longer needed, it can be torn down and it's resources can be used to create another shortcut. This approach is very similar to the one used in [2] to reconfigure optical networks connecting processors at the printed circuit board (PCB) level and showed clear performance improvements for realistic network loads. These improvements were bigger for larger networks and we expect similar results.

II. RecoNoC Design

A first goal of this work is to create a compact, flexible and easily extendable NoC on a FPGA. This will allow exploration of the NoC's performance for different network configurations. The design of RecoNoC therefore consists of nodes with an arbitrary number of in- and outputs, which can be connected in a flexible way, with the possibility to add shift registers between nodes to pipeline long links. Arbitrary topologies can be created, but currently we will limit ourselves to rectangular meshes. With additional buffering to avoid metastability issues, the design can be used in Globally Asynchronous, Locally Synchronous (GALS) systems, where every node in the network uses a different clock signal. Virtual channels (VCs) are also implemented, as these can increase performance and are necessary to achieve deadlock-free routing with some dynamic routing protocols. RecoNoC currently uses wormhole routing and a simple ACK/NACK handshake link protocol. Table-based static routing is used, because it is necessary to be able to change the routing tables as we change the topology. Turn restrictions are used to avoid deadlocks. Arbitration takes 3 clock cycles and node buffering takes 2 cycles so the minimum node delay is 5 cycles. The links between nodes can transfer 1 byte of packet data every 4 cycles. T_{Wait} is the time that a packet waits in the buffers if some links are occupied. Total packet delay in cycles is thus calculated using Equation (1).

$$T_{delay} = 5 \cdot hops + 4 \cdot bytesPerPacket + T_{wait} \quad (1)$$

Secondly, we wanted to use the dynamic reconfigurability of SRAM-based FPGAs to create shortcuts that allow data streams to travel between distant nodes, using their own dedicated link and not passing through intermediate buffers or routers. To add the shortcuts to RecoNoC, we increased the radix of each node by 1 and connected the extra in- and output port to a crossbar. This crossbar is connected to each

of the neighbouring nodes' crossbars, as shown in Figure 1. With these crossbars, we can set up point-to-point simplex links from any node to any other node, effectively creating a shortcut. Packets can travel along these shortcuts and get to their destination in a single hop, avoiding additional wait times if links along the regular route are in use by other data streams. For long links, this can greatly improve packet delays [2].

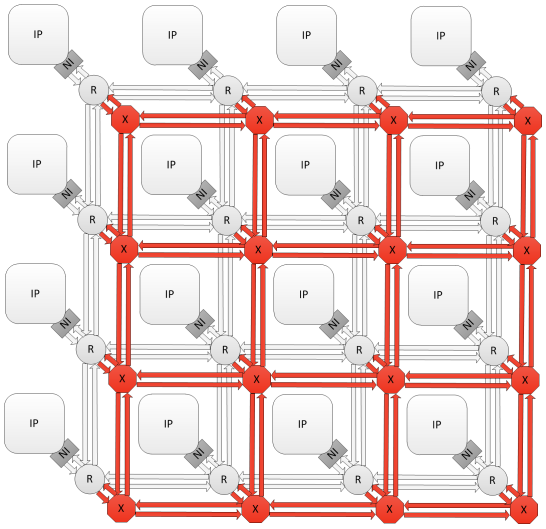

Figure 1. RecoNoC with a regular 4x4 mesh topology. NI = Network Interface, R = Router, X = crossbar.

III. Preliminary results

There has been quite some previous work on dynamically reconfigurable NoCs, for example [3] and [4]. In all cases Xilinx's Modular Design Flow was used. In this work, on the other hand, the TMAP dynamic datafolding toolflow, described in [5], was used to automatically generate the dynamically reconfigurable crossbars of RecoNoC. A unique feature of TMAP is that only some of the FPGA's look-up tables need to be rewritten during run-time, while the routing is kept fixed. This is expected to drastically reduce reconfiguration time. The dynamically reconfigurable crossbars are still much smaller than crossbars that can be adapted using control inputs implemented in the FPGA fabric. This is because the control inputs are considered constant, i.e. connections are hardwired and are changed by reconfiguring the FPGA. The area reduction for one crossbar is shown in Table I.

Radix	LUTs	LUTs (TMAP)	Area reduction
3	45	39	13,3%
4	162	52	67,1%
5	215	65	69,8%

Table I
CROSSBAR AREA REDUCTION USING TMAP.

Radix	Base (slices)	2xDW	2xBD	2 VCs	reco
3	210	44,8%	56,7%	95,2%	40,5%
4	297	41,4%	49,8%	90,2%	30,6%
5	397	39,5%	43,1%	88,2%	22,0%

Table II
COMPARISON OF NODE AREA COSTS. DW = DATA WIDTH, BD = BUFFER DEPTH, VC = VIRTUAL CHANNEL, RECO = WITH RECONFIGURABLE CROSSBAR.

To illustrate the network's flexibility, the size of one network routing node for different network configurations is shown in Table II. This experiment was conducted in ISE 9.1 on a Xilinx Virtex-2 Pro (XC2VP30) device. The base node to which is compared has a data width of 8 bits, a buffer depth of 4 bytes and no VCs. We see that, for a node with radix 5, doubling the data width, doubling the buffer depth and using VCs costs a lot more than the crossbars. We also see that adding the reconfigurable shortcuts only results in a limited overhead of 22 %.

IV. Conclusions and future work

We have presented the design of RecoNoC: a compact, highly flexible NoC, that can be easily adapted for various experiments. Using the TMAP dynamic datafolding tool flow, we also enhanced this NoC with dynamically reconfigurable shortcuts. These extra shortcuts can be used to change the topology of the network to adapt to the network's communication needs. The area cost of adding these shortcuts is modest, but the performance gains are expected to be substantial. In the future we want to finish implementing the tools that measure our NoC's performance for different network configurations and traffic patterns.

References

[1] A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Öberg, M. Millberg, and D. Lindqvist, "Network on a chip: An architecture for billion transistor era," in *Proceedings of the 18th IEEE NorChip Conference*. IEEE, November 2000.

[2] I. Artundo, W. Heirman, C. Debaes, M. Loperena, J. Van Campenhout, and H. Thienpont, "Low-power reconfigurable network architecture for on-chip photonic interconnects," in *17th IEEE Symposium on High Performance Interconnects*, New York, NY, Aug. 2009, pp. 163–169.

[3] T. Pionteck, C. Albrecht, R. Koch, and E. Maehle, "Adaptive communication architectures for runtime reconfigurable system-on-chips." *Parallel Processing Letters*, vol. 18, no. 2, pp. 275–289, 2008.

[4] J. Y. Hur, S. Wong, and S. Vassiliadis, "Partially reconfigurable point-to-point interconnects in Virtex-II pro FPGAs," in *ARC'07*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 49–60.

[5] K. Bruneel, F. Abouelella, and D. Stroobandt, "Automatically mapping applications to a self-reconguring platform," in *DATE*, K. Preas, Ed., Nice, 4 2009, pp. 964–969.