Fault-tolerant Routing Approach for Reconfigurable Networks-on-Chip

Pekka Rantala*, Teijo Lehtonen*†, Jouni Isoaho*† and Juha Plosila *†
*University of Turku, Dept. of Information Technology, Communication Systems Lab.
†Turku Centre for Computer Science (TUCS), Lemminkäisenkatu 14A, FIN-20520 Turku, Finland.
Email: { peaura | tetale | jisoaho | juplos }@utu.fi

***Abstract*— We introduce fault-tolerant on-chip routing philosophy for two-dimensional meshes. It is an extension to the concept of packet connected circuit, PCC. In order to increase reliability we have designed an automatic rerouting property to a single switch node and added return channel to the communication route. An autonomic routing switch node is modeled asynchronously and implemented using Haste language. The logical functionality of routing is illustrated as a single study case in 7*8 mesh. The routing success is further analyzed in congesting and faulty environment.**

I. Introduction

The move towards nanoscale circuits poses new challenges to circuit design. Shrinking dimensions decreases the yield in manufacturing [1]. The yield can be maintained at an acceptable level by admitting some amount of faults in a chip. Electromigration problems can be overcome by the use of built-in redundancy and dynamically reconfigurable circuit structure [9]. Thus, error and fault tolerance issues need to be partly moved to system design and architecture level issue from today's low level testing and testability design. Moving towards reconfigurability, scalability and efficient resource sharing on redundant platforms will increase fault tolerance on system-on-chip design.

The traditional method for communication on chip is to use either point-to-point links or time-division buses such as the AMBA bus from ARM, Inc [2]. These structures have inherent problems with scalability and flexibility. A way to lower the impact of these problems is to merge these two opposites into a network structure. This network-on-chip will consist of a shared set of links and routers that will give higher scalability than the bus and larger flexibility than the point-to-point links [3].

Routing in direct networks, that can be applied to NoCs, is introduced in [4]. A two dimensional mesh network topology (Fig. 1) is the most usual way to interconnect, though there are efficient alternatives researched by e.g. Dally [5].

This research discusses the five lowest layers of the OSI reference model [10]. We sketch a fault-tolerant communication protocol and routing that reaches for a minimal bidirectional communication path between source and destination node. The routing algorithm used in this phase of research is simple and developed especially for this router architecture.

The paper is organized as follows: Section II reviews the concept of fault-tolerant routing on reconfigurable NoCs. In section III the implementation of a routing switch node is introduced. Section IV reviews the results of simulating the routing in network. Finally the future work and conclusions are presented in sections V and VI.

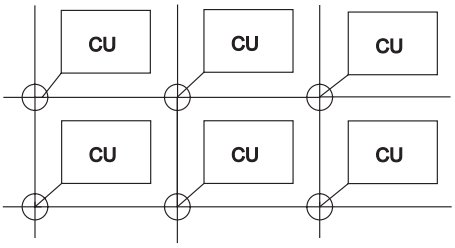

Fig. 1. Section of a typical reconfigurable network-on-chip with switched interconnection

II. Fault-tolerant Approach

A. Fault-tolerant NoC

Fault-tolerance on homogenous and redundant reconfigurable network means small granularity and autonomy [8], [9]. In the case of a fault from manufacturing phase, a normal microchip is totally discarded due to malfunctioning. In contrast, reconfigurable logic may only have an erroneous module that can be switched off and replaced functionally. The smaller the erroneous part is, the more we can replace them.

To achieve fault-tolerant design philosophy we need to support independency between functional units. Consequently, the switch nodes do not have any kind of global control (and clocking) and the nearest neighbors affect on each other only by control signals. The increase of bandwidths and mobility of applications demand low power solutions on circuits. In this design, it is achieved by asynchrony, autonomy and low complexity. There is only few ports logic depth in the simple routing algorithm.

Addressing scheme of the destination is relative, which increases the autonomy of the units. The switch nodes do not have their location data when resetting the circuit. The address is described as rectangular distance to the destination node and it is always modified correspondingly when sended to the next node.

B. PCC-based Routing

The routing method introduced in this paper reminds hybrid packet-circuit switching known as packet connected circuit (PCC) [6]. The PCC uses a small routing packet that traverses the network and sets up a circuit switched route for the payload data to follow. After communication the route resources are released with a command from the sender side. The same routing concept is used in SoCBUS [7]. By using the PCC the need of buffers and thus the latency penalty is eliminated. Also the granularity of the NoC is as small as possible due to low-complex routing algorithm. The smallest redundant unit in this network is the I/O-unit inside the switch node in Fig. 2.

Drawbacks such as rerouting trials decrease due to automatic rerouting between nodes, introduced in the next chapter. This novel property can make a channel bypass a faulty or congesting link in network. In addition, the routed data channel is designed as *bidirectional* which provides feedback (e.g. resending requests) from the destination node.

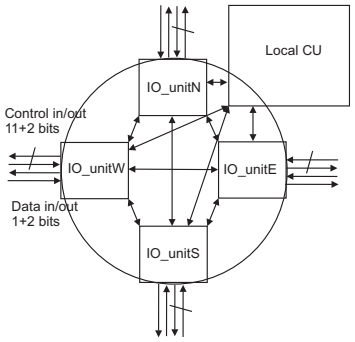

Fig. 2. Switch node with I/O-units inside and local CU

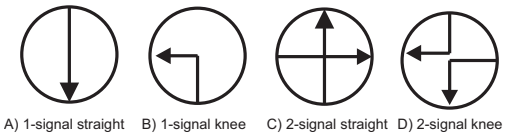

Fig. 3. Routing configuration types

III. Switch Node Implementation

A. Architecture

Fig. 2 presents a single switch node. It is divided into four I/O-units that take care of the traffic in each direction. From outside there are control and data signal channels coming from and going to each I/O-unit. The control signals are 11 bits wide (including 4+4 bits wide address field for 7*7 mesh) and data signals one bit (but easily extendable). The channels inside the node are for control signals between the I/O-units. The I/O-units have also a connection to each data output which is not drawn in the Fig. 2. In addition, each I/O-unit is connected to the local CU that can input and output control and data signals with the I/O-units. All mentioned channels are implemented asynchronous way including wires for request- and acknowledgment signals.

The switch node can route up to two routes in four different ways according to the Fig. 3. A single I/O-unit can be reserved for only one route. Fault-tolerance property is that an I/O-unit can be reserved also for other reasons than routing. That reason might be permanent faultiness of the data channel connected to the I/O-unit (detected by e.g. coding like Hamming), congestion of the data channel or faultiness of the I/O-unit itself (detected by e.g. current monitoring). All these reasons can make the I/O-unit reserved and make routes bypass it.

Used routing strategy decides the best routing direction in each routing case. It is derived from the address field of the *route* command and the shortest path ambition. The shortest path is tried to find with as few turns as possible. Hence, the route tries to travel straight forward until the other address coordinate is zero. This strategy causes fewest knee type configurations (Fig. 3 b and d) that would prevent crossings of the routes. For instance, a *route* command received into the switch node from north with relative address (3, -1) would come out from the I/O-unit south with address field (3, 0) due to positive X-distance and negative Y-distance. In the next node the command would be directed from I/O-unit north to east.

In this design, an asynchronous router node that detects errors in data channel is implemented for logical simulation purposes. It is modeled using Haste and Timeless Design Environment (TiDE), the design language and toolset for asynchronous design by Handshake Solutions [11].

B. I/O-unit Configuration

The configuration state flow of a single I/O-unit is described in Fig. 4. Each I/O-unit has an arbiter that chooses between six signals: control signals from three other I/O-units inside the switch node and outside from the adjacent switch node, data signal from the adjacent switch node and signals from the local CU. Currently, there are four control commands: *route, unroute, reroute* and *false_dir*. Interface to the local CU is not in the scope of this research. However, it controls the sending and receiving of the data payload after creating the route (top left branch of the Fig. 4) unless the I/O-unit is reserved to some other route.

If the local CU does not reserve this I/O-unit it (and some other I/O-unit in same switch node) can be reserved for another route. The procedure of handling a control command from the adjacent node is presented in the bottom left branch of the flow. When the command is: *route* and the I/O-unit is unreserved the command is transferred to one of the three I/O-units according to the best direction for the address carried by the command. Now this I/O-unit is reserved for this target address that is saved into its registers. The rest three commands *unroute, reroute* and *false_dir* can be executed after a route has reserved this I/O-unit. *Unroute* is sent by the original source of the route and it discards the whole route by resetting reservations of all the I/O-units on the route. *Reroute* resets also the I/O-unit and is sent backward along the route to the earlier I/O-unit that

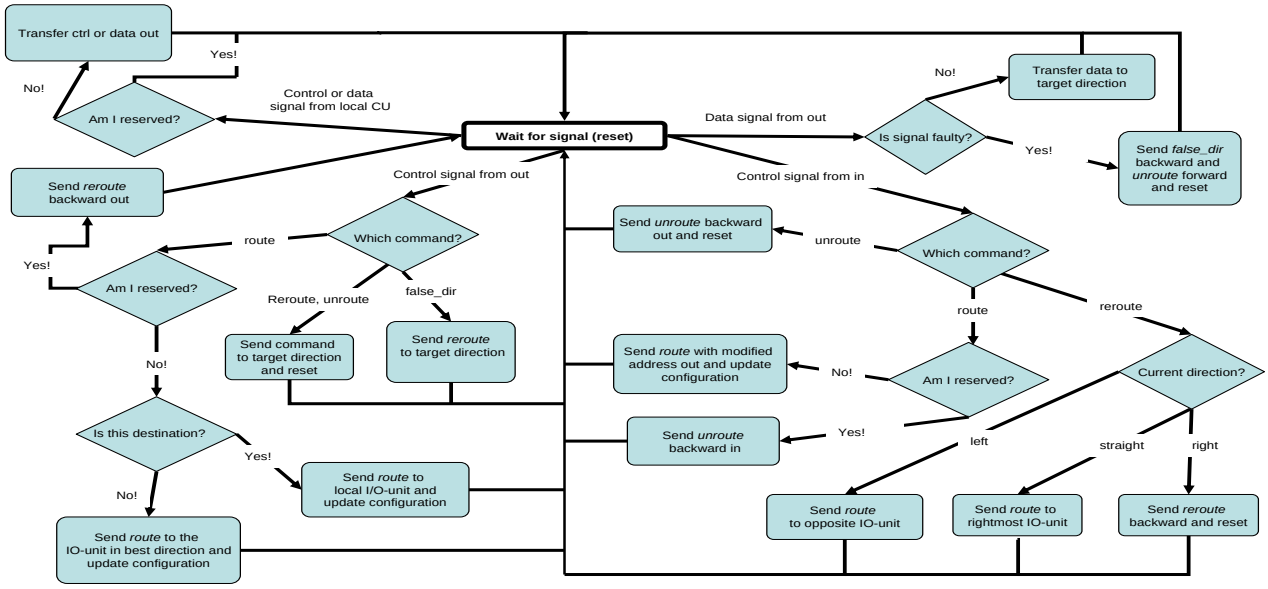

Fig. 4. Configuration state flow of I/O-unit

handles rerouting. *False_dir* indicates (permanent) failure in outgoing data channel. It causes *reroute* command backward but leaves this I/O-unit reserved to prevent further routes via it.

Fault detecting is modeled in the top right branch of the state flow in Fig. 4. The I/O-unit directs the data (bit) to earlier targeted direction out from the node. In the case of detected error the route is discarded in forward direction and the previous I/O-unit in the previous node switch is informed with *false_dir* command.

Control commands from other three I/O-units are handled in bottom right branch of the state flow 4. In the case of unreserved I/O-unit, *route* command from inside the node is mediated forward out to the next switch node and corresponding routing configuration is saved. The X- or Y-coordinate of the relative address field is incremented or decremented depending on the direction of the I/O-unit. Rerouting property is utilized again if the I/O-unit is reserved. *Unroute* command resets the I/O-unit and sends it forward to the next node switch. In the case *reroute* a new routing effort is simply ”right-handed”: the new route is searched from the next I/O-unit in clockwise direction. If this is not possible the route is withdrawn to the previous node by sending *reroute* backward out. This rerouting strategy does not take into consideration the address which is not essential according to simulations results.

IV. Network Simulation

A. Routing Progress

The logical simulation model of a single switch node (without interface to the CU) was realized with TiDE toolset [11] and mapped to a mesh with VHDL testbench. The speed and area of the circuit are not optimal yet but a logical routing simulation demonstrates the functionality well.

In Fig. 5 there is a 7*8 mesh with four letter pairs. The time unit Δ means *an approximate delay of a route command traveling through a node* (i.e. two I/O-units) without any rerouting inside the node.

At time T=0 the other sides of nodes pairs A, B, C and D send *route* command with address of their counterpart. At T=Δ the commands have traversed the first node. Route B has turned towards the target. At T=2Δ the pair E begins also communication. At T=3Δ D has bounced from the route A and C and B have traveled across A. At T=5Δ B and C have completed their route for data transmission. D has turned clock wisely south and E bounces to A. At T=11Δ A and D have completed their routes and E is withdrawing after making a loop route. There are four 2-signal straight -type configurations (Fig. 3 c) and two 2-signal knee -types (Fig. 3 d). Finally after T=24Δ the E connection is completed via lower B-node.

B. Routing Quality Measures

In faulty and congestive network traffic the PCC does not always route. With 2D-mesh topology routing is guaranteed only with one communicating pair and one fault [5].

With Matlab [12] tool the success of routing was simulated. In Fig. 6 the probability of success is plotted as a function of simultaneously contacting node pairs and faulty links between the nodes. The communicating pairs and the faults are uniformly distributed over a 25*25 mesh. With amount of [0, 10, 20 and 30] faults and [5, 8, 10, 12, 15, 18 and 20] communicating pairs, 200 iterations were driven.

From Fig. 6 it can be seen that probability of successful routing for all the node pairs decreases rather linearly from

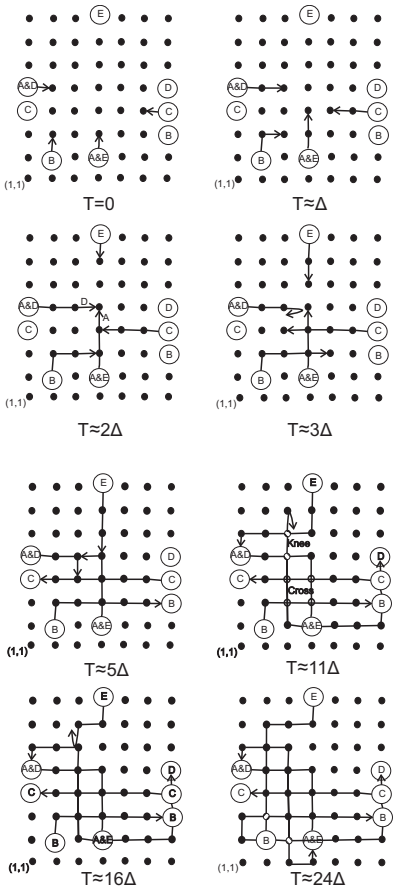

Fig. 5. Routing case in 7*8 mesh

almost 100% to 0% when number of node pairs increases to 20 while number of faulty links increases to 30.

Also the aggregate route path length was calculated compared to the minimum length. The average path length never was over 10% over the minimal routing path, when the routing was successful. Hence, the overhead is not an important issue when faultiness and traffic increases.

V. Future Work

Fault-tolerant routing approach introduced in this paper needs developing on many levels. As mentioned, fault detecting and channel congestion can be performed physically and logically and embeded to a system in numerous ways. The reservation of a communication link could be modeled as permanent or dynamic.

The physical model of the switch node is implemented currently only for logical simulation purposes. The next development phase is to optimize the asynchronous model of the switch node. Its area and delay measures must achieve reasonable level on the given NoC architecture.

The architecture of the node might be improved to achieve smaller granularity and signal delays. Another direction of development is agent technology that embeds more intelligent and specialized properties to the switch nodes. However, the original goal, fault-tolerance, must be retained.

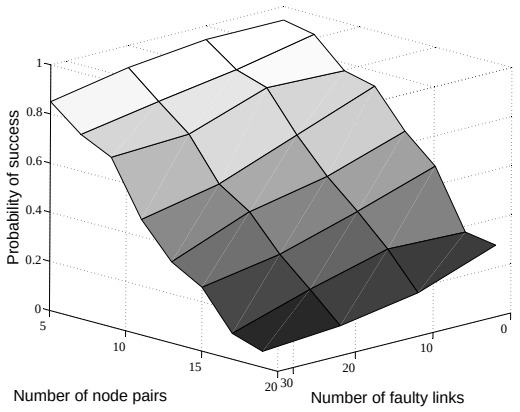

Fig. 6. Routing success

VI. Conclusions

A novel fault-tolerant routing philosophy, an automatic rerouting extension to the packet connected circuit, for NoCs was introduced and analyzed successfully. The routing switch node was implemented and simulated in faulty environment. In the presence of faults, the successful routing paths turned out to be averagely less than 10% longer compared to the minimum paths without faults. This routing concept gives us various options to manage fault and congestion challenges on nano-scale and high bandwidth reconfigurable systems.

References

[1] *International Technology Roadmap for Semiconductors 2005*. (http://public.itrs.net).

[2] *The ARM AMBA protocol*. (http://www.arm.com/products/solutions/AMBAHomePage.html).

[3] Daniel Wiklund, Sumant Sathe, and Dake Liu, *Benchmarking of On-Chip Interconnection Networks*. The 16th International Conference on Microelectronics, 2004. ICM 2004 Proceedings.

[4] L. M. Ni and P. K. McKinley, *A survey of wormhole routing techniques in direct networks.* IEEE Computer, 26(2):62–76, 1993.

[5] W.J. Dally, *Performance analysis of k-ary n-cube interconnection networks*. IEEE Trans. Computers. ,Vol. 39, No. 6, June 1990, 775–785.

[6] Daniel Wiklund *An on-chip network architecture for hard real time systems*. Licenciate thesis, Linkping Studies in Science and Technology, Jan. 2003, ISBN 91-7373-577-9.

[7] Daniel Wiklund and Dake Liu, *SoCBUS: switched network on chip for hard real time embedded systems*. in Proc of the International Parallel and Distributed Processing Symposium, 2003.

[8] Barry W. Johnson, *Design and Analysis of Fault-Tolerant Digital Systems*. Addison-Wesley, 1989.

[9] Teijo Lehtonen, Juha Plosila and Jouni Isoaho, *Fault-tolerance in Nanoscale Circuits*. Technical Report, Turku Centre for Computer Science (TUCS), Aug. 2005.

[10] H. Zimmermann *OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection*. IEEE Transactions on Communication, 28(4):425-432, April, 1980.

[11] Handshake Solutions http://www.handshakesolutions.com.

[12] Matlab http://www.mathworks.com/