Hardware for Multiconnected Networks: A Case Study

G.Campobello (+) , G.Patané (+) , M.Russo (+)

(+)Dept. of Physics, Univ. of Messina and INFN Catania,Italy;
gcampo,gpatane,mrusso@ai.unime.it

Abstract

This paper presents the experimental results of a research project financed by INFN (National Institute for Nuclear Physics). The goal of this project is to realize a cluster of Personal Computers (PCs) suitable for distributed software in the field of nuclear physics. The developed hardware must have a low cost and must be able to interconnect 32 PCs with an expansion capacity of up to 100. Due to the requirements of this research project, it seems natural to look at it as an application of our general hardware architecture presented in [1]. Here, we present the results obtained and compare them with other works in literature. The results show that our architecture is preferable when low cost hardware and high performance are the goals.

Keywords: Computer Communication Networks, Switching, Routing, Multiconnected Networks, Digital Design.

1 Introduction

The case study depicted below is part of a research project of the INFN. The objective of this project is the realization of a cluster of computers for the distributed elaboration of algorithms related to nuclear physics. The performance of a cluster depends greatly on the sub-network of communication [2]. For this purpose research is directed to the development of network interfaces able to minimize the latency and maximize the throughput of communications. In particular, the principal points of the research concern:

- the design of a new transmission medium and of the related transceivers with the aim of obtaining high-bandwidth links;
- the study of new routing algorithms that use techniques of artificial intelligence (neural networks, fuzzy logic, genetic algorithms, etc.).;

- the realization of a network interface for PCI or AGP buses able to implement the routing algorithm in hardware;
- the implementation of a communication driver that can efficiently transfer information between the network interface and the operating system minimizing communication latency.

The first prototype of the system is a cluster of 32 hosts, with the possibility of a gradual increase up to a maximum of 100 hosts. It uses links with 100 Mbps bandwidth (C_i) with the aim of using a commercial transceiver in conformity with standard IEEE802.3u (FastEthernet). It is also desirable that, even in the worst case (when all the hosts transmit contemporarily) a bandwidth of at least 80Mbps (C_{max}/N) is guaranteed to each host. Lastly, in order to avoid problems related to Electro-Magnetic Interference (EMI), we must use a maximum clock frequency for the hardware of 50 MHz (f_{WM}).

The article is structured as follows: In Section 2 the design of the router is presented with the aim of applying our design flow and the proposed architecture to our case study. In Section 3 some comparisions with other router architectures in literature are shown and in Section 4 the conclusions and future prospects of such research are reported.

2 The Design

Here we report our studies and results. On the basis of constraints indexed in the previous section and following the design flow described in the paper [1], we have chosen a network topology and a routing algorithm and sized the different units of the router.

2.1 Choice of the topology

The specifications of our project imply that the topology must be easily scalable; this reduces the choice, among the topologies listed in Table 1, to the Recursive-Cube of Ring (RCR) or to the k-ary n-cube. In fact, in all of the other topologies, if we want to expand the network we need at least to double the hosts. From equation (1) in [1] relative to the maximum capacity for each host, and considering the design constraints, this must be $\frac{2g}{D} \geq 0.8$. For reliability, we impose that, for a network of 100 hosts, it is $\frac{2g}{D} \geq 1$.

With reference to Table 1, fixing the values n and k so as to realize a network with around 100 hosts (N= 100), we can observe (Table 2) that the RCR is the topology that allows the relation $\frac{2g}{D} \geq 1$ to be satisfied with the minimum

value of g. We choose the topology that satisfies the precedent relation for the least value of g with the aim of reducing the costs. In fact, the number of cables and transceivers for each host is directly proportional to g. It is possible to observe that, for this value ($g = 5$), the RCR appears, essentially, as a certain number of cubes connected by 8 rings. Each host is connected to three other hosts of the same cube and with two hosts on the same ring (see Fig. 1).

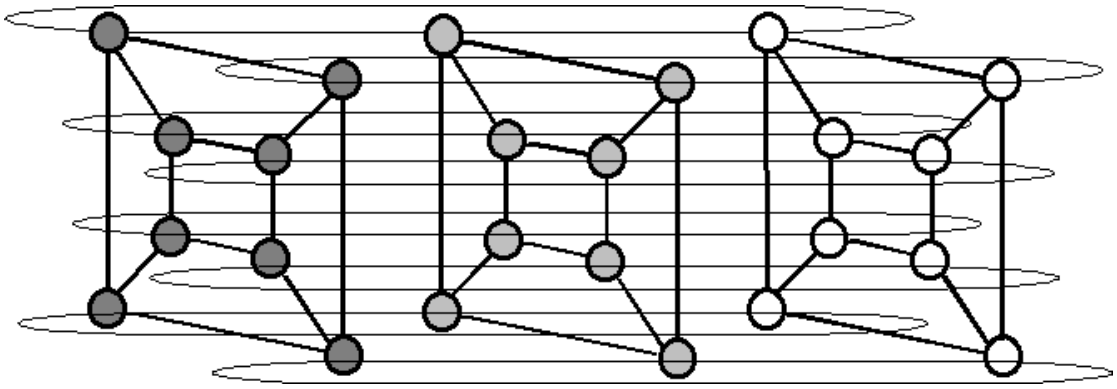

Fig. 1. Topology RCR(3,3,0)

2.2 Input Unit and Output Unit Sizing

The transceivers of the standard IEEE802.3u provide two modalities of data transfer: serially at 100 MHz or in parallel at 25MHz (4 bit at a time). As we want to use a clock frequency less then 50MHz, we have adopted the second solution. Thus we have $f_i = 25$MHz and $b_I = 4$. Since we intend maximizing the performance of the network, according to the results shown in [3], we adopt a Virtual Cut-Through (VCT) switching technique. Such a policy foresees that the packets that temporarily are not routable, because of the occupation of the destination port by another packet, are memorized in a buffer. For this purpose, one of the channels of the switch is connected to the memory of the network card, as a consequence we have $n_{IMUX} = g + 1$, that is to say, the number of the inputs of the MUX is bigger by one unit in comparision to the number of links for each host. So, we can multiplex the data coming from the links in the same way as for those coming from the memory. The same additional channel is used both to transmit the packets generated by the same host at which the router is located and to receive the packets destined to the host. For multiplexing the $n_{IMUX} = g + 1 = 6$ channels with time division, we must impose (see equations (3) in [1])

$$b_A \geq 12 \tag{1}$$

Taking the least value of b_A that satisfies the relation we have $f_B = 50$MHz. The addresses of 100 hosts can be represented with 7 bits (inferior to b_A); therefore all the bits of the destination address of the packet can be on the internal bus of the switch in the same clock period.

Since the memory can be connected directly to the MUX, it does not need IU/OU units ($n_{IU} = n_{IMUX} - 1$).

In order to verify if we can use FPGA of the family FLEX10KA-1, we calculate, by the coefficients in Table I in [1] and the equations given in the previous paragraph, the maximum frequencies of the different units in the case $b_A = b_B = 12, b_I = b_O = 4, n_{IMUX} = 6$. Table 4 shows that all the maximum frequencies (f_{max}) of the different units are greater than the work frequencies (f_W) that we need to use.

2.3 Choice of the Routing Algorithm

As explained in [1], the choice of switching technique and routing algorithm has a great impact on harware complexity and performance. In order to maximize the throughput we have chosen VCT as the switching technique, while, with the aim of reducing hardware complexity, it is preferable to implement a minimal routing algorithm, so that routing decisions are based solely on destination address. Nevertheless, an adaptive algorithm would allow higher performance. Therefore an intermediate approach is adopted that makes use of an adaptive algorithm with a simple hardware implementation. This solution combines *congestion bit* technique with a classical minimal algorithm used in multiconnected networks: the *e-cube routing*.

2.3.1 E-cube routing

This algorithm is a non-adaptive, minimal, deadlock free routing algorithm for n-dimensional hypercubes [4].

A n-degree hypercube consists of 2^n nodes arranged in a n-dimensional cube, where each node is connected to n other nodes. In a n-degree hypercube the address of each node may be encoded with a n-bit string so that the addresses of directly connected nodes differ only by one bit. It is obtained by ossociating a bit for each dimension, disposing two nodes in this dimension and assigning them a different bit value. With this addressing scheme the routing algorithm can proceed as follows.

When a message is received at an intermediate node, the node and destination node addresses are compared using a bitwise exclusive-or (XOR) operation. If,

after the XOR operation, there is one bit sets to one (1) then the message must traverse in that dimension to reach the destination. e-cube routing chooses the least significant bit that is set to 1, and sends the message in the corresponding direction, waiting for the link or node to be free (not busy). Proceeding in this way, e-cube stops routing when there are no more bits that need to be corrected. At this point, the destination has been reached.

2.3.2 Applying the e-cube algorithm to RCR

The chosen topology (RCR) is not a hypercube therefore we cannot apply the e-cube algorithm directly. Nevertheless, few modifications are needed.

The RCR is formed by cubes (hypercubes of order 3) connected by rings (see fig.1). In particular, the address of a host can be seen as a ring part (R_p) followed by a cube part (C_p) expressed, respectively, by an integer and a string of three bits: the integer identifies the position of the host on the ring, while the three bits codify its position inside the cube.

Inside the cube we can apply e-cube routing, as explained before, by doing a XOR operation between the cube part of the addresses, so that we obtain a three bit string (z,y,x) that identifies the directions to follow, as explained before.

An arithmetic difference between the destination address and the address of the host that must route the packet (current host) is sufficient to determine if it is more convenient, in terms of distance, to forward the packet from one side or the other of the ring. More properly, if N_a is the number of hosts on a ring, we must evaluate the difference, modulo N_a, of the integer part (R_p) of the addresses and compare it with $N_a/2$.

Let us observe that XOR operator is equivalent to the modulo-2 sum. This observation will allow us to extend the e-cube also to rings.

Let us imagine the string (z,y,x) extended through two bits (l,r) to obtain the five-bit string PosDir=(l,r,z,y,x) and to evaluate the values of (l,r) by using the difference mod N_a previously computed, as follows: l=1 if $0 < |R_{p,ch} - R_{p,dh}|_{N_a} \leq \lfloor N_a/2 \rfloor$ and r=1 if $|R_{p,ch} - R_{p,dh}|_{N_a} \geq \lceil N_a/2 \rceil$ where $R_{p,ch}$ and $R_{p,dh}$ are, respectively, the ring part of the address of the current host and destination host. Note that:

- (l,r)=(0,0) if current node and destination node are in the same cube;
- (l,r)=(1,0) if it is more convenient to forward the packet on the left side of the ring;
- (l,r)=(0,1) if it is more convenient to forward the packet on the right side of the ring;

- (l,r)=(1,1) if it is indifferent which side is chosen.

With this extension we can apply the e-cube routing using the string PosDir. It is possible to observe that PosDir identifies the dimensions that the packet has to traverse to reach its final destination i.e. the possible directions that can be chosen.

2.3.3 congestion bit

To make the routing adaptive we have used the *congestion bit* technique. Essentially a host in the congestion state advises the nearby hosts by resetting a suitable bit inside the packet. The host getting a packet with the congestion bit set to zero considers the link from which the packet arrived as busy, up to when a subsequent packet advising the end of the congestion state arrives. It is possible to show, by simulations, that, with such a technique, the number of packets sent towards hosts in the congestion state is reduced; this has the effect of reducing the latency of the network.

The congestion bit mechanism is implemented endowing the router with a Congestion-Mask (CongM): a bit of the Congestion-Mask is associated to each output port. When a packet *generated* (and not simply *forwarded*) by a neighbor arrives, its congestion bit is copied into the CongM in the position corresponding to the port from which the packet arrived. For the future we are planning to manage the Congestion-Mask with a smart unit.

2.3.4 Next Host Computation

In this section we will show, with a detailed example, the routing algorithm, i.e. the sequence of steps needed by a host to determine the next-host on the route to the destination.

We assume that each node maintains three strings: CongM and PosDir, explained in the previous section, and a third string, named Mask, of g bits (one bit for each port) that identify if a port is free (1) or busy (0).

When a packet arrives, the host evaluates PosDir and refreshes CongM as shown before. After, a logical AND between the three strings (Pos-dir, Mask, CongM) gets the optimal ports for the routing. Currently, if after the AND operation there are still more possible ports for the routing (in other words if there are many 1s), the algorithm uses a static priority assigned to the ports, that is to say, it selects the port correspondent to the bit most to the left among those set, according to the order left-right-Z-Y-X where left and right are the ports of the left hand and right hand of the ring and Z, Y, X are the ports on the cube in homologous directions. Vice versa, if the resultant

string is null but there is sufficient space in the buffers then the packet is temporarily memorized. Otherwise, if the space in the buffers is not sufficient, it is discarded.

The figure 2 illustrates the steps just explained.

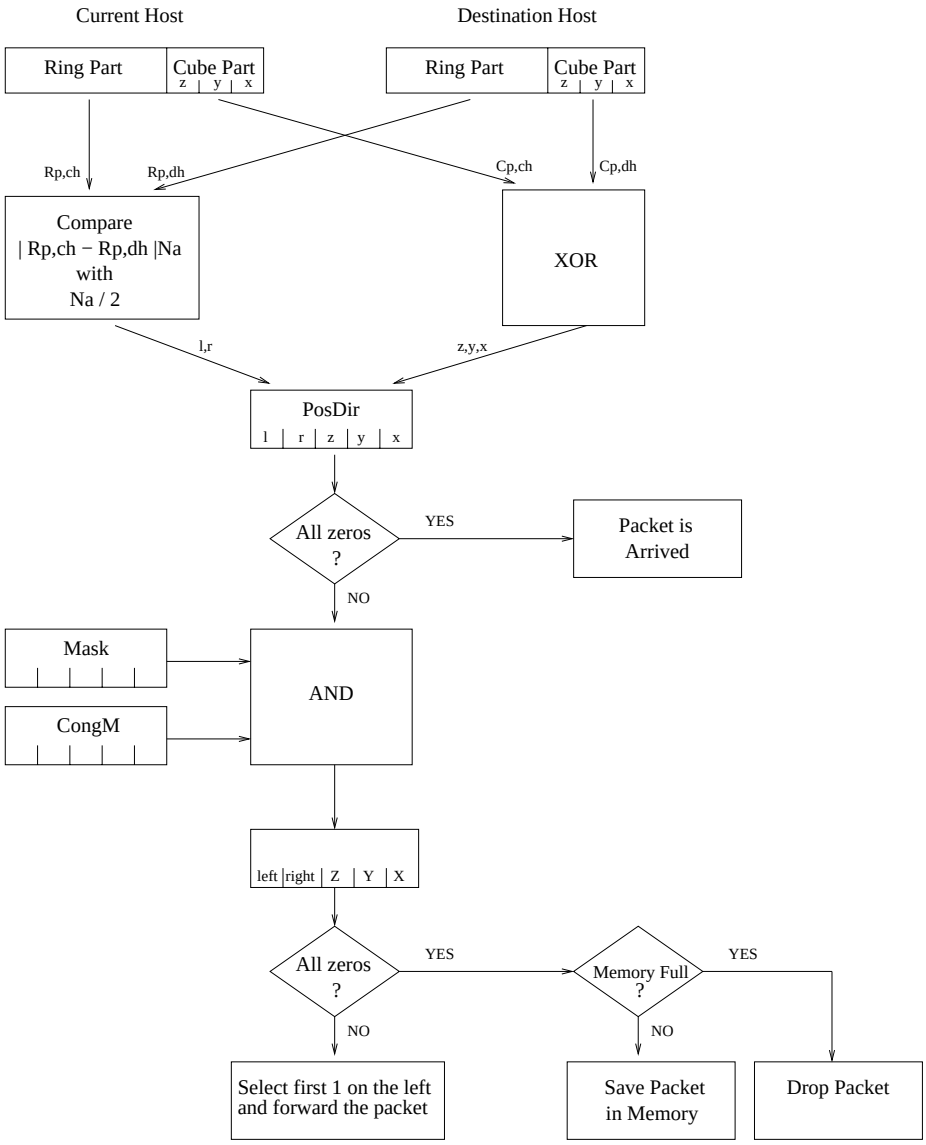

Fig. 2. Functional description of the Routing Algorithm

With the aid of figure 3 we will show a simple example to clarify everything.

In the figure 3, an RCR network with 24 nodes is shown. The nodes are disposed on three cubes ($N_a = 3$) connected by eight rings. Therefore each node may be identified by an address (R_p, C_p) composed by an integer (R_p) in the range 0-2 and a three bit string (C_p). We will assume that the host (0,"000") wants to send a packet to the host (1,"110").

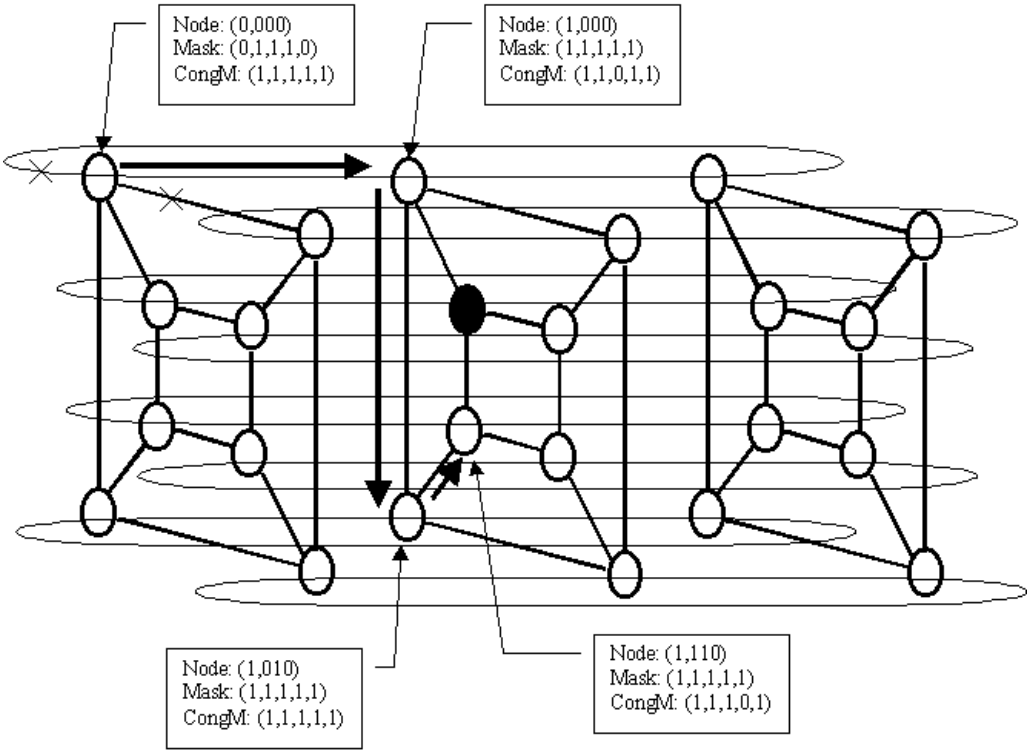

Fig. 3. Example of routing: the host (0,"000") sends a packet to the host (1,"110"). The black host (1,"100") is in congestion state and it has advised its neighbors, then the host (1,"000") avoids forwarding it the packet.

- Step 1: Source node (0,"000") compares its address with the destination address (1,"110") to evaluate the PosDir string. The difference, mod N_a, of the integer parts of the addresses is $|0-1|_3 = 2$ and the XOR operation between cube parts returns "110" so that the PosDir string is (0,1,1,1,0). The congestion mask of the source node is CongM=(1,1,1,1,1) so that none of the neighbor nodes are in congestion state. The Mask string is (0,1,1,1,0) therefore ports in l and x directions are busy. From the AND operation between PosDir, CongM and Mask the string (0,1,1,1,0) is obtained. The 1s in this string identify the directions through which it is possible (free port) and convenient (not congested) to route the packet. The node selects the first one on the left and forwards the packet in the right direction (to the node (1,"000")).
- Step 2: The packet arrives at the node (1,"000").

  The node compares its address with destination address and sees that they are in the same cube (i.e. the difference of the integer parts of the addresses is zero). The XOR opeartion between the cube parts of the addresses permits the evaluation of the PosDir string to be completed (PosDir=(0,0,1,1,0)). The value of the Mask=(1,1,1,1,1) indicates that all ports are free while CongM=(1,1,0,1,1) specifies that the node in z direction is in congestion state. The AND operator returns (0,0,0,1,0) and the packet is routed in the y direction (toward (1,"010")).

- Step 3: The packet arrives at the node (1,"010").
  With the same procedure of previous nodes, PosDir=(0,0,1,0,0) is obtained. Because all ports are free (CongM=(1,1,1,1,1)) and none of the neighbors are in congestion state (CongM=(1,1,1,1,1)), the AND operator returns (0,0,1,0,0). The packet is forwarded in z direction (to the node (1,"110")).
- Step 4: The packet arrives at the node (1,"110"). The node compares its address with destination address and notes that they are the same because the difference of the integer parts of the addresses is zero and the XOR operation between the cube parts also returns zero. It follows that the packet has arrived at the destination and the router sends it to the host through the host interface.

2.3.5 Performace Evaluation

We realized a simple network simulator for analyzing different routing algorithms.

The simulator is a C program that simulates RCR networks at the flit level. A flit transfer between two adjacent nodes is assumed to take place in one clock cycle (a tic). The network is simulated synchronously, moving all flits that have been granted channels in one clock cycle and then advancing time to the next cycle. The simulator can be configured to support different network sizes, degree, number of buffers, routing algorithms, message lengths and message generation rates. The simulator can generate various statistics, such as average message latency, maximum latency and throughput.

All of the experiments are conducted under uniform traffic distribution, assuming message length (LF) of 100 tics and a message generation rate equal to 20% of total node capacity. Total node capacity can be expressed as g/LF because a node can send at most g packets at the same time and it must wait for the complete transmission of the packets (LF tics) before it can begin to transmit.

The simulations show that a buffer with size equal to 3 packets is sufficient to guarantee a very low probability of deadlock, provided that the Congestion-Mask is considered only for hosts of the same cube. Moreover, in simulations we have observed that the throughput is maximized if the host enters the congestion state and stops injecting more packets onto the network when 2 buffers are full and if the host leaves the state of congestion when all the packets in the buffer have been transmitted. Once the maximum dimension of the packet (P Byte) has been chosen we need a memory of $3P$ Byte.

Table 3 shows the results of some simulations for RCR networks of different size (N=32,64,128) with and without the congestion control. As it is possible

to obvserve, the congestion mask allows latency to be reduced. For example, for $N = 64$ latency reduction due to the congestion mask is of approximatively 35%.

2.4 RTL Implementation of the Routing Unit and Mux Control Unit

Relatively to the Mux Control Unit (MCU), a static priority is associated for each input channel: if two words of two packets arrive contemporarily, the word coming from the channel with greater priority is the first transferred onto the internal bus.

In particular, the memory has the greatest priority in order to reduce the latency; in fact, the packets temporarily blocked, waiting for a free output channel, originate from the memory channel. The links of the ring and, after, those of the cube follow in the priority ranking. In order to maximize the peak bandwidth, several packets can be transmitted from the memory to the switch, allowing all the six time slots to be utilized. In particular, the MCU has as output a NextCh-free signal that advises the host interface that the next time-slot is free. By monitoring this signal it is also possible to analyze the traffic on the network and to adapt the traffic generated by the host to the actual traffic in the network.

The complexity and the timings of the Routing Unit (RU) and MCU are evaluated by VHDL codes and their following synthesis (see Table 4). Once the timings τ_{MCU} and τ_{RU} have been extracted from the simulations, we can calculate the time of elaboration as $\tau_e = \tau_{MCU} + \tau_{RU}$; it is equal to 33 ns.

2.5 FIFO Sizing and Router Latency Evaluation

In order to use a 50MHz clock, it is necessary to insert a sufficient number of registers on the critical path MCU-MUX-RU-DEMUX to make the maximum delay less then $\frac{1}{f_B} = 20ns$. Given that $\tau_{io} + \tau_e = 50ns$, two registers dividing the critical path into three parts, each with a propagation time less then 20ns, are sufficient. In particular, it is possible to insert a register after the MUX, and one after the RU. In order to balance the two paths, as seen in the previous paper in the subsection V-A in [1] relative to the pipeline, it is necessary to insert a further register before the DEMUX.

So, the FIFO queue is constituted by 2 registers of 12 bits. The complete switch is represented in Fig. 4.

We can now evaluate the mean router latency in conditions of low traffic.

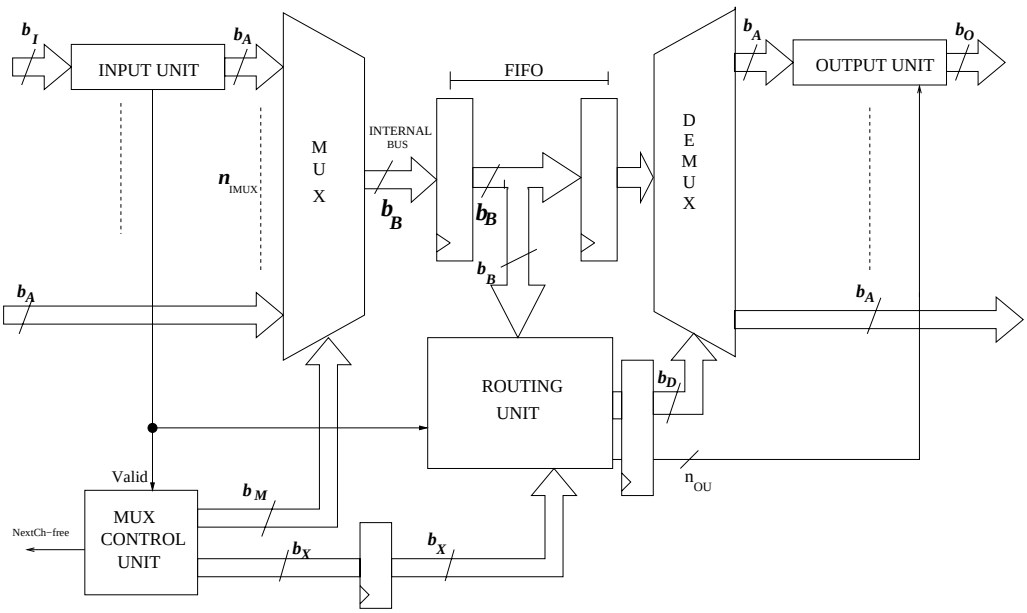

Fig. 4. Final Switch Diagram

Placing the current values into equation (4) in [1] we obtain $\tau_R = 0.32\mu s$. Adding this value to the delay due to the transceiver [5], 60 ns in transmission and 210 ns in reception, we obtain $\tau'_R = 0.59\mu s$. For a cluster of 104 hosts with RCR topology we have $d = 4.5$, from which a network setup latency $lat_{setup} = \tau'_R \cdot d = 2.65\mu s$ follows (in comparison with the 5μs of an ideal 500 port Fast Ethernet Switch).

2.6 Complexity Evaluation and FPGA choice

By the equations given in the preceding paragraph we can evaluate the complexity of the FPGA. As regards the number of LCs and FFs, from equations (6)-(9) in [1], it follows:

$$LC_{tot} = 5(28 + 26) + 56 + 90 + 41 + 96 = 553 \quad (2)$$

$$FF_{tot} = 2 \cdot 5 \cdot 29 + 24 + 25 + 56 = 395 \quad (3)$$

For calculating the number of PINs, we have to consider that the switch is interfaced with 5 transceivers, each with two 4-bit buses (one bus is for reception and one is for transmission) and that an input of the MUX, likewise an output of the DEMUX, are directly connected to a memory with a data bus of 12 bits. Additionally, we need 7 more bits to set the address of the host and 5 bits for the Congestion-Mask, that we assume can also be controlled externally.

Besides, it is opportune to multiply the value obtained above by a factor 1.2 to take into account the presence of control and handshake pins necessary for the exchange of information between the switch, the transceivers and the memory. Finally, we have:

$$PIN_{tot} = 1.2 \cdot [2(5 \cdot 4 + 12) + 7 + 5] = 91 \quad (4)$$

From the datasheet of the family FLEX10KA we have determined that such resources are available in model EPF10K10A TC144-1 (see Fig. 5).

We wish to underline that, after the board is realized, thanks to the reprogramming capability of the FPGA, it will be possible to evaluate routing algorithms different from the one proposed if their complexity, in terms of LCs, FFs and PINs, does not exceed the resources present in the FPGA. It will also be possible to change the topology of the network, provided that the new target topology has a degree less or equal to 5.

3 Related Works and Their Comparison with our Case Study

There are many works in literature concerning router architectures. To the best of our knowledge, not so many of them are directly comparable to our proposal. Here, we briefly review some of the most recent papers in literature that can be approximately related to our work. At the end we summarize the main characteristics of all the works and underline advantages and disadvantages.

An interesting work is reported in [6]. The authors present a router architecture for flexible routing in multihop point-to-point networks. The router consists of a single-chip programmable routing controller (PRC) and an external memory that acts as a buffer. The PRC provides 4 bi-directional physical links each including three virtual channels. There is a separate routing engine for each physical link. A 32-bit time multiplexed bus is used to share data among inputs, outputs and the host. Of course, due to the programmable hardware, the average packet latency is high, above all when there are high network loads. The single-chip implementation employs an internal clock speed of 40 MHz and links at 200 Mbps. The PRC has been designed using the HP CMOS14 process and was packaged by MOSIS. About 500,000 transistors were required.

In [7], a parametrized cost and speed model for router performance is described. The paper analyzes the Dimension Order [8], Planar Adaptive [9], Turn Model [10] and *-channel [11] routing algorithms. The model assumes the use of a 0.8 μm CMOS gate array process. The technology permits the

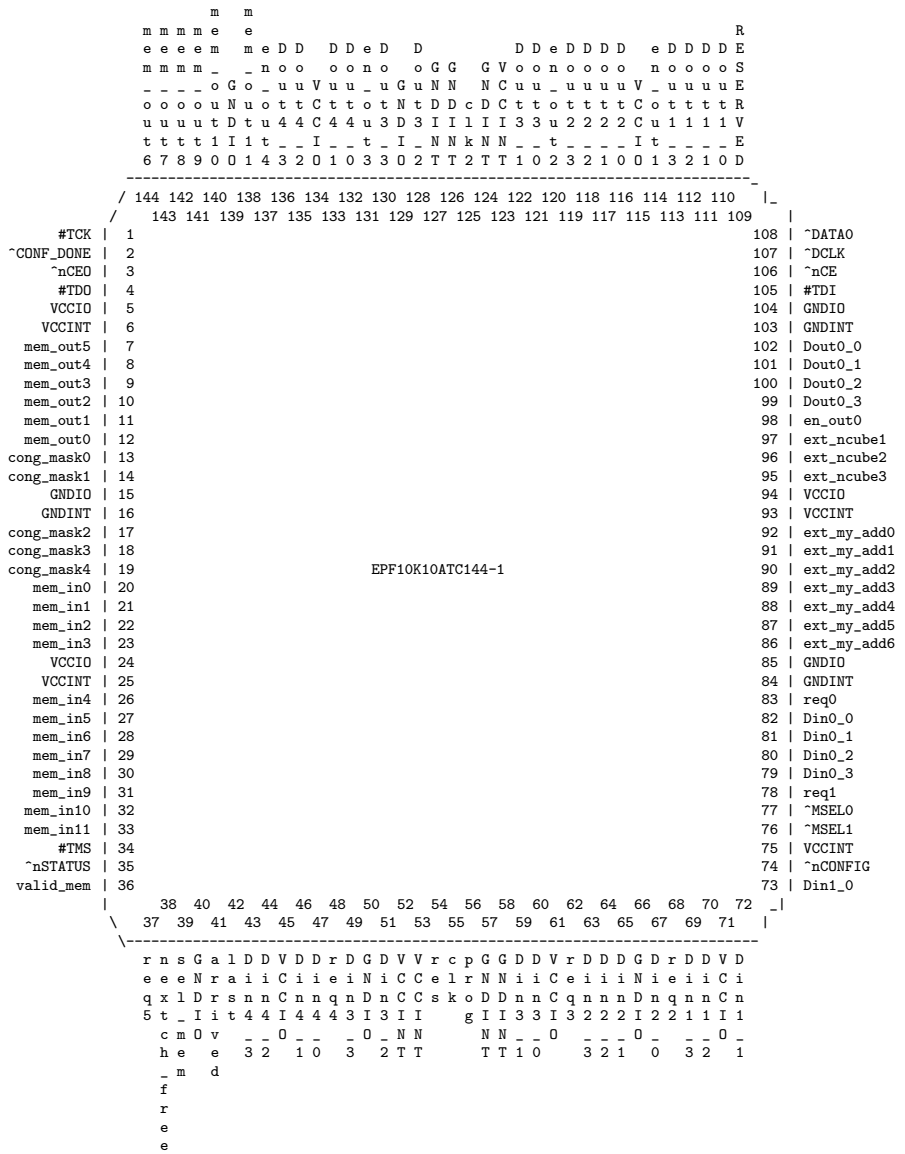

Fig. 5. Pinout

realization of the basic two-input NAND gate with a delay ranging from 250 ps (no load) to 750 ps (five gate load). The aim of the work is to understand the complexity of different routing strategies in terms of realization cost and speed. The essential components of the router proposed are: a crossbar, that provides the basic switching functions; the flow control units, that perform flow control and manage buffers; the address decoders, that examine the packet header to generate the set of possible routes; the routing arbitration logic, that chooses the path and connects and disconnets inputs and outputs and virtual channel controllers, that multiplex the physical channels. The cost model does not include the physical layout and interconnection issues. The reported gate counts assume routers with 16-bit datapaths and channels and they do not include I/O buffering, pads and synchronization. Gate counts for

the analyzed routing algorithms for different numbers of inputs and outputs are reported.

The authors of paper [12] propose another kind of router. It was designed for real-time multicomputer networks. The work presents a novel router architecture that supports end-to-end delay and guarantees throughput by scheduling packets at each network link. The prototype developed was geared toward two-dimensional meshes. Starting from a precise theoretical study the authors develop an architecture whose core is a comparator tree scheduler. The prototype was realized using a 0.5 μm CMOS process, required 900,000 transistors, operates at 50 MHz and introduces a latency of 115 ns.

In [13] the authors present an implementation of a prototype reconfigurable router based on FPGAs. The prototype implements IPv4 routing with a throughput of up to 576Mbps, using a stream-based approach that facilitates dynamic reconfiguration. It consists of three modules in series, each implemented on a Xilinx 4062XL-3 FPGA and connected by a 36-bit systolic bus. The modules have a clock speed of 36MHz and process one packet at a time as 16 data bits per cycle. The proposed router has been implemented with about 320 Configuration Logic Blocks (CLBs). In the Xilinx 4000XL series, CLB contains two four-input look-up tables (LUTs), one three-input LUT, and two flip-flops. As stated in the Xilinx Data Sheet, the LUTs in a CLB are equivalent to 30 gates. The proposed prototype interacts only with memory and does not take into account the physical links. More precisely, the latency reported in the paper, about 100ns, does not include transceiver management and the possible conflicts due to data exchange among transceivers and memory. It is not possible to evaluate latency added by these operations.

Table 5 reports the main characteristics of our work and of the reviewed papers when they use generally 4 or 5 links. As is evident, there are large differences.

- [6] and our work are both programmable. The main difference is that our programmability depends on the FPGA implementation, whereas in the other work it is due to a specialized programmable microprocessor. The main differences are, of course, the latency and the size; in both cases our proposal is better.
- The other two approaches are not programmable, but they are both interesting because they represent two different alternatives: [12] is an architecture for real-time communications while [7] is an accurate study based on an architecture whose core is a crossbar switch. Taking into account our target technology, our solution is, obviously, slower. Instead, the design cost is lower. As regards the number of transistors, our solution is comparable to the Dimension Order Algorithm in [7]. Instead, work [12] requires the highest number of transistors. This is a consequence of the hardware required to guarantee real-time communications.

- In comparison with our work, the router proposed in [13] needs more transistors. This is a consequence of a larger data bus. It seems to have less latency but it processes one packet at a time and the value reported does not take into account latency introduced by transceivers.

4 Conclusions and Future Works

This paper accurately presents the design steps of a network interface for a cluster of PCs. Compared with previous works, our solution gives the best tradeoff between costs and performance. Further studies will be carried out to develop a gigabit network interface card.

References

[1] G.Campobello, G.Patané,M.Russo, Hardware for Multiconnected Networks: The Design Flow .

[2] M. Baker, R. Buyya, Cluster Computing: Architectures and Sistems Vol.1, Prentice Hall, 1998.

[3] K.G.Shin, S.W. Daniel, Analysis and Implementation of Hybrid Switching, IEEE Trans. on Computer .

[4] H. Sullivan, T.R. Brashkow, A large scale homogeneous machine, In Proceedings of the 4th Annual Symposium on Computer Architecture .

[5] National Semiconductor, DP83846A DsPHYTER, Single 10/100 Ethernet Transceiver (2000).

[6] S.W.Daniel, K.G.Shin, S.K.Yun, A Router Architecture for Flexible Routing and Switching in Multihop Point-To-Point Networks, IEEE Trans. on Parallel and Distributed Systems .

[7] A.A.Chien, A Cost and Speed Model for k-ary n-cube Wormhole Routers, IEEE Transaction on Parallel and Distributed Systems .

[8] W.J. Dally, C.L. Seitz, The Torus Routing Chip, J. Distributed Computing .

[9] A.A. Chien, J.H. Kim, Plannar-adaptive routing: Low cost adaptive networks for multiprocessors, Proc. Int'l Symp. Computer Architecture .

[10] L. Ni, C. Glass, The Turn Model for Adaptive Routing, Proc. Int'l Symp. Computer Architecture .

[11] P.Berman et al., Adaptive Deadlock and Livelock Free Routing with all Minimal Paths in Torus Networks, in: Proc. Symp. Parallel Algorithms and Architectures, 1992.

[12] J.Rexford, J.Hall, K.G.Shin, A Router Architecture for Real-Time Communication in Multicomputer Networks, IEEE Trans. on Computer .

[13] J.R. Hess et al., Implementation and Evaluation of a Prototype Reconfigurable Router, in: IEEE Symposium on FPGAs for Custom Configurable Computing Machines, Napa, California, 1999.

[14] ALTERA Corporation, Flex10K: Embedded Programmable Logic Family, a-ds-f10k-04.01 Edition (June 1999).

[15] M. J. S. Smith, Application-Specific Integrated Circuits, Addison-Wesley Longman, 1998.

[16] C.L.Seitz, The Cosmic Cube, Comm. ACM .

[17] A.Ralston, De Bruijn sequences. A model example of the interaction of discrete mathematics on computer science, Mathematics Magazine .

[18] F.P. Preparata, J. Vuillemin, The Cube-Connected Cycles: A versatile network for parallel computation, Communication of the ACM .

[19] E.D. Demaine, S. Srinivas, A novel routing algorithm for k-ary n-cube interconnection networks, in: High Performance Computing Symposium, 1995.

[20] S. Latifi, P.K. Srimani, A New Fixed Degree Regular Network for Parallel Processing, 8th IEEE Symposium on Parallel and Distributed Processing .

[21] A.W. Fu, S.C. Chau, Cyclic-Cubes: A New Family of Interconnection Networks of Even Fixed-Degrees, IEEE Transaction on Parallel and Distributed Systems .

[22] Y. Sun, P.Y.S. Cheung, X. Lin, Recursive Cube of Rings: A New Topology for Interconnection Networks, IEEE Transaction on Parallel and Distributed Systems .

List of Tables

5 The main characteristics of our and related works when there are 4 or 5 bi-directional links. The number of transistors of our work has been evaluated taking into account that in [14] it is reported that a LC corresponds to 12 gates and a gate corresponds to 4 transistors [15]. We have also assumed 20 transistors for each flipflop [15]. 19

Table 1
Some known topologies in literature and the relative principal parameters.

Name	N	g	D	L
Hypercube [16]	2^n	n	n	$n2^{n-1}$
De Bruijn [17]	2^n	4	n	2^{n+1}
Cube-Connected Cycle [18]	$n2^n$	3	$2(n-1)+\lfloor n/2 \rfloor$	$3n2^{n-1}$
k-ary n-cube [19]	k^n	$2n$	$n\lfloor k/2 \rfloor$	nk^n
Shuffle Exchange Perm. [20]	$n!$	3	$\frac{9n^2-22n+24}{8}$	$3n!/2$
Cyclic Cube [21]	nk^n	$2k$	$\lfloor 3n/2 \rfloor$	nk^{n+1}
Recursive-Cube of Ring RCR(3,n,0) [22]	$8n$	5	$\lfloor n/2 \rfloor + 3$	$20n$

Table 2
Comparision between RCR and k-ary n-cube

Topology	g	$2g/D$
Recursive-Cube of Ring RCR(3,13,0)	5	10/9
10-ary 2-cube	4	8/10
5-ary 3-cube	6	2

Table 3
Simulation results for RCR networks for different sizes (N), With and WithOut Congestion Control (WCC vs WOCC).

N	WOCC throughput	WCC throughput	WOCC Latency	WCC Latency
32	0.90	0.91	79	37
64	0.73	0.74	130	84
128	0.41	0.46	260	230

Table 4. Complexity and work frequency of different units

Unit	$LC_n \cdot LC_{Opt=Area}$	FF	$f_n \cdot f_{max,Opt=Area}$ (MHz)	f_W (MHz)	Evaluated by
IU	28	29	102	25 (f_i)	coefficients and equations
OU	26	29	118	25 (f_i)	”
MUX	56	0	154	50 (f_B)	”
DEMUX	90	0	81	50 (f_B)	”
FIFO	–	24	166	50 (f_B)	equation FF_{FIFO}
MCU	41	25	77	50 (f_B)	compilation and simulation of RTL code
RU	96	56	51	50 (f_B)	compilation and simulation of RTL code

Table 5
The main characteristics of our and related works when there are 4 or 5 bi-directional links. The number of transistors of our work has been evaluated taking into account that in [14] it is reported that a LC corresponds to 12 gates and a gate corresponds to 4 transistors [15]. We have also assumed 20 transistors for each flipflop [15].

Reference	Transistors (K)	Latency	Links	Notes
[6]	500	Higher	4	Programmable, TDM
[12]	900	115 ns	4	Not Programmable, Scheduler
[7]	34-128	a few ns	5	Not programmable, Crossbar
[13]	34-50	>100 ns	?	Programmable
Our work	34	590 ns	5	Programmable, TDM