Hardware for Multiconnected Networks: The Design Flow

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 an architectural study that may be used to design and develop a general, efficient hardware for multiconnected networks. Starting from the constraints, the proposed design flow allows an accurate evaluation of cost and performance of the final implementation. The generality of the architecture permits the realization of networks with practically any size or topologies.

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

1 Introduction

This paper presents a complex study in which we propose the design flow of a general, efficient hardware solution useful for the realization of a router for multiconnected networks. In particular, we will illustrate the design flow by which it is possible to dimension the different modules of the proposed architecture. We will report some experimental data end empirical equations that allow the complexity and the performance of the router to be estimated before its actual realization. We have chosen Field Programmable Gate Array (FPGA) as the target technology because it is well suited to rapid prototyping but our architecture and design flow are not tied to a particular technology. For this purpose, as far as possible, all the equations reported are quite independent of the target technology. The article is structured as follows: Section 2, illustrates the key elements of a distributed system. Section 3 shows how the different elements of a network interact, with the goal of transfering the information correctly. Section 4 depicts the proposed *router architecture*. Here we identify the constituent modules and we explain their functions. Section 5 illustrates the proposed *design flow* that, beginning from the typical needs of a communication network design (bandwidth, latency, number of nodes etc.), allows the correct planning of the different modules and the evaluation of their

complexity, in terms of Logic Cells (LCs) and Flip-Flops (FFs), with the aim of determining the specific FPGA model necessary for the router implementation. In Section 6 the conclusions and future prospects of such research are reported.

2 Distributed Systems

The efficiency of a distributed system is connected to the performance of the communication network. This is characterized by three elements: the topology, the routing algorithm and the policy of flow control. These elements determine the performance of the network, i.e. its bandwidth and its latency. Such performance is not absolute, in fact, an efficient network for one definite application may be inefficient for another [1]. Some applications necessitate a low latency (i.e. applications for voice traffic), others prefer a high bandwidth (i.e. applications related to file transfers). In other cases an application needs an intermediate solution.

As regards the topology, multiconnected networks are used more and more because the increase in the number of cables for each node is a simple and economic way to increase the bandwidth, and also because such topologies increase the system fault tolerance. The presence of several links on each node means that each interface needs a routing unit able to forward correctly the information that transits on the network. In order to have a low latency, it is necessary for this unit to be a hardware unit. The policy by which a packet is forwarded from one host to another is called the switching technique. It has a notable impact on the performance of the network. Currently, there are three techniques of switching that we can use in networking:

- Store & Forward (SF): a packet is completely received by the host, memorized in a buffer, processed later and, after this, resent [2].
- Virtual Cut-Through (VCT): the header of the packet is processed immediately to determine the output port. If a port is free the packet is immediately sent to the next host otherwise it is memorized in a buffer [3].
- WormHole (WH): The difference between VCT and WH is in the way the packets are treated when the output port is busy [4]. In the WH the packet is logically divided into flits (flow control digits). When a packet is blocked, each host crossed by the packet saves only a flit and not all the packet. This allows a reduction in the memory necessary for the routing unit, but, on the other hand, reduces the network throughput. In fact, in WH, if only one buffer of size equal to a flit is associated to each output port, then a stalled packet blocks many physical links while in the VCT such links remain free for other packets. It is possible to increase the throughput of a WH network associating several buffers to each port or, in other words, introducing vir-

tual channels [5].

Compared to SF, the last two techniques permit an inferior latency. In fact, in SF the packet must be totally acquired and memorized to each node passed through. On the other hand, SF allows the use of slower interfaces and, therefore, it is less expensive.

The *routing algorithm* [6], i.e. the way to select the path followed by the information from its source to its destination, is another element that can substantially influence the performance. Such a choice can be made directly by the source host (*source routing*) or can be made host by host (*distributed routing*). Source routing algorithms require that each host knows the topology and the state of the net. This fact reflects negatively on the size of the packet header. For this motive, many algorithms of routing are distributed.

A second classification can be made between *adaptive* and *non-adaptive algorithms*. The latter, called also *oblivious routing*, determine the output port exclusively on the basis of the addresses (of the destination, source and current host), typically forwarding the packet to the port that minimizes the distance between the current host and the destination host (*minimal algorithms*). Most routers implement a single policy of routing, often minimal and not adaptive, because this reduces hardware complexity and the computation time for the routing [7], but the least distance path does not always coincide with the least latency path. The adaptive algorithms, adapting themselves to the state of the network, usually have the best performaces. They are usually distributed.

The aspects analyzed so far, essentially related to the interconnection network and to its management, do not consider another important factor for global performance. A central role in a distributed system is played by software. It is known from literature that different interconnection networks also have very different performances depending on the distributed algorithm and on its software implementation [1]. Additionally, it is possible that the distributed system, in concomitance with a multitasking operative system, possibly also distributed, has to manage many applications that can be very different in terms of hardware requirements. All of these considerations accentuate the importance of the flexibility in the hardware of a distributed system.

2.1 The Topology

By *topology* we intend the way of connecting the network elements (hosts). An optimal topology must connect the maximum number of hosts with the minimum number of links, minimizing the distance between the hosts. The symmetries in the topology simplify the routing algorithm. For this reason the

hosts usually have the same number of links (g, called degree of the topology) and they are connected with regular geometric structures. In multiprocessor systems and clusters the most common topologies are mesh and hypercubes, both particular cases of the k-ary n-cube topology (this is adopted for example in the CRAY-T3E [8]). There are many works in literature that specify, by simulation, the performance of different topologies and routing algorithms in terms of throughput and latency [9]. Usually, a network simulator, such as PP-MESS-SIM [10], is employed with the aim of evaluating different solutions and taking projectual decisions.
For regular topologies, an important relation in the selection of the best topology, given certain requirements, is the relation that links the principal parameters of the topology, that are the degree g and the diameter D, defined later in this section, to the maximum capacity of the network (C_{max}), that is the maximum amount of information that can travel concurrently in the network.

Let us suppose that the number of links concurrently held by a packet is equal to d. If we indicate the total number of bidirectional links of the network with L, than the maximum capacity of information on the network in a time unit, C_{max}, is $\frac{2L \cdot C_i}{d}$, where C_i is the unidirectional bandwidth of a single link. Since each link connects two hosts and each host is connected to g links, then L is equal to $\frac{Ng}{2}$, where N is the number of hosts. As a consequence of this the maximum capacity can be evaluated as $C_{max} = \frac{NgC_i}{d}$.

In the case of VCT switching or WH switching, if the packet transmission time is greater than the signal propagation time and the time of elaboration, then the mean number of links occupied by the packet is equal to the mean distance between the source and the destination hosts. Generally, for multidimensional networks that are symmetrical to each dimension such as Hypercubes, Mesh, k-ary n-cube and Recursive-Cube of Ring (RCR), the mean distance is half of the maximum distance also called the diameter (D) of the topology.

In such a hypothesis, the maximum capacity per host can be expressed as:

$$\frac{C_{max}}{N} = \frac{2g}{D} \cdot C_i \quad \text{bps} \tag{1}$$

2.2 The routing algorithm

In accordance with Fleury and Fraigniaud [11] any routing algorithm can be defined by means of three functions:

(1) A topological function (T) that, on the basis of the packet header to be routed and of the state of the network, determines, for each host, a

subset of output channels of the host on which it is possible to forward the packet;
(2) A selection function (S) with the task of selecting, a free channel, if one exists, among those defined by T;
(3) A header function (H) that considers the eventuality that the routing algorithm can alter the header of the packet to be routed.

When implementing the routing algorithm in hardware, the flexibility of the algorithm is usually sacrificed in favor of less hardware complexity. Many of the actually implemented algorithms [12] are oblivious (dimension order [4], e-cube [13]). Therefore function T is simply a look-up table and functions S and H are not necessary. Such types of algorithm present performances that are inferior with respect to the adaptive algorithms because they do not exploit alternative paths for the routing [6].

3 Basic operations of a router

In the following, we analyze the steps necessary to send and receive a message, with the aim of illustrating the operations of the proposed router (see Fig. 1) and of explaining how the router interacts with the relative host and the network.

When an application needs to send a message, the host, or the interface, divides the message into small, simple units called *segments*. Some information (*header*) is added to each segment. It permits the routers to forward the segment to its destination, to reconstruct the original message and to exchange information related to the *state* of the network (for example the connection of new hosts, the congestion of the traffic or the tear-down of a connection, the current communications etc.). Other information is added, usually in the tail, i.e. the Frame Check Sequence (FCS). It allows the interface to determine if the information has been corrupted by noise [6], [14]. The set of segment, relative header and protection code is called *packet* and is the informative unit sent from the interfaces on the physical medium. The more operations that are delegated to the interface, the less amount of time will be wasted by the CPU for the management of the network. Negatively, this requires the interfaces to be more complex and more expensive. Typically, the operation of segmentation is effected by the host while the FCS is calculated in hardware by the interface.

After the host has transfered the data to the memory of the interface, the routing unit implements functions T, S and H. In the simplest case (not adaptive or static routing) the decision is based entirely on the state of the channels (free/busy) and on the address of the destination host (and/or source host).

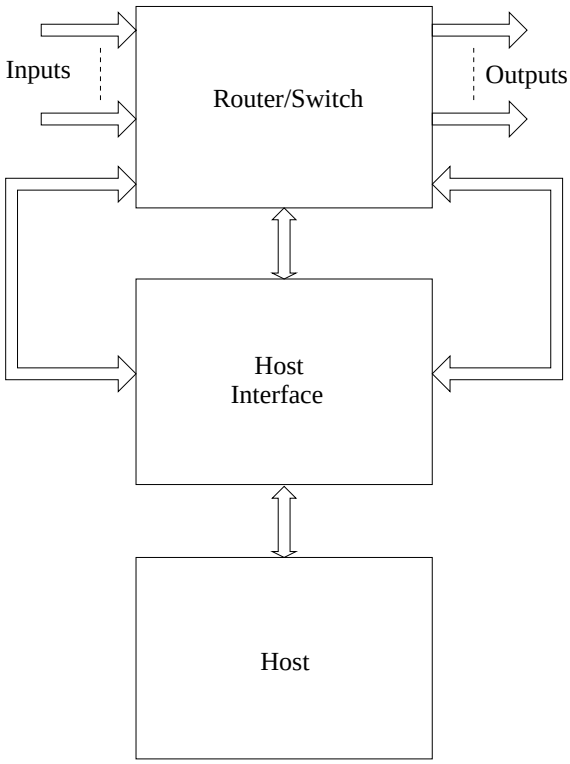

Fig. 1. General blocks of the router. The Router/Switch is physically connected to the network and manages the correct transmission and reception of the information. The host can be a source and a destination of the information and communicates with the Router by means of a Host Interface. The Host Interface has the task of transfering the information from/to the host by means of an opportune supported protocol (e.g. PCI bus) and to memorize temporarily the information exchanged between the Router and the Host.

Otherwise, it is possible to use also the state of the network. Once the destination port has been established, the packet is sent to the next host. In particular, for a correct transmission, bits must be encoded and/or modulated to adapt them to the physical medium. The physical interfaces that deal with this conversion in transmission and also in reception are called transceivers. Usually, the transmission is serial; so the words arriving from the memory must be serialized in the network interface.

The packet arrives at the next router. A suitable receiver unit begins to acquire bits and accumulates them until there is sufficient information to take a decision on the routing. Afterwards, the routing algorithm is applied again. If the packet is destined to another host, the routing unit tries to forward the packet. As was said in Section I on switching policies, if the physical channels are all busy, the router can decide to memorize the whole packet in a *buffer*, in the case of VCT switching, or only part of it, in the case of WH switching.

The introduction of the buffers (known also as virtual channels in WH) involves an increase in the cost; for this reason we need to choose the correct

number of buffers. Also the position of such buffers is important. In the case of WH switching, since the buffer dimension is relatively small, they are put inside the router (usually there are 2 or 3 buffers for each physical channel). As regards VCT and SF switching, only recently has it been possible to integrate, in the router, buffers with sufficient dimension to contain the complete packet. However, for packets of large dimension or in the case that the number of required buffers is high, it is necessary to use memories external to the router. In order to foresee this demand we have decided to consider the buffers external to the router.

To cope with the possibility that many packets with the same destination arrive contemporarily we need an arbitration system that establishes which packet must go on and which packet must be temporarily blocked or discarded.

After it passes through various routers, a packet arrives at destination. The routing unit recognizes that its address is present in the header, removes the packet and sends it to the host interface. The host interface checks whether there are any errors, and if there are not, forwards the packet to the host. The interface of the destination host considers the other information in the header, for example the address of the source host. Finally, when all the packets of the same message have arrived, the original message can be recomposed and forwarded to the operating system or to the application of the destination host.

4 General Description of the Architecture

In accordance with other works in literature [7] [15], we can identify the following base components in a generic architecture of a router (see Fig. 2):

- A switching unit that allows the redirection of a flow of information from the input channels to the output channels;
- Some input units that allow the flows of information to be stopped and the packet to be delayed so as to read the information contained in the packet that is necessary for routing.
- A Control Unit that can be logically divided in:
  - · An arbitration system that permits the correct management of packets with the same destination port that arrives simultaneously at the switch;
  - · A routing unit that elaborates the algorithm of routing;
- Some output units to interface the router to the physical channels and to the buffers.

Some considerations must be made related to the switching unit. A solution consists in a cross-bar system that allows point-to-point connections between

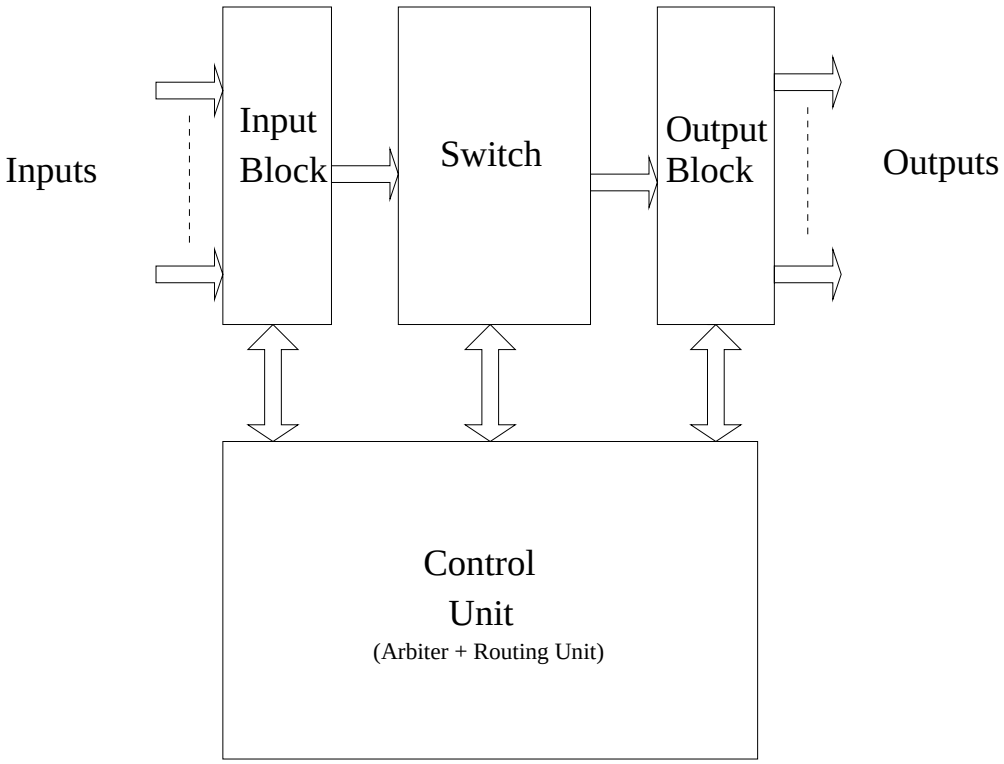

Fig. 2. Router Blocks Diagram. The packets come from the adjacent hosts and from the same host to which the router is connected. They are collected from a block of Input, commuted by a switching unit and sent on to the opportune output ports. All the operations are syncronized by a Control Unit

the input channels and the output channels. Otherwise, it is possible to multiplex the different flows of information by space division or time division multiplexing techniques. The second solution allows the centralization of the routing unit and then the reduction of the cost of the system. For example, the architecture proposed in [7] uses a cross-bar and needs an address decoder for each input while, with time division multiplexing, it is possible to use only one shared decoder for all the inputs, placed in the Control Unit. When the router is implemented with a FPGA, a time division multiplexing is desirable because, usually, there are no internal three-state buffers inside such programmable devices. This makes the realization of like-bus connections (necessary for cross-bar) impossible.

The architecture that we propose is shown in Fig. 3. We distinguish in it the Input Units (IUs), whose task is to adapt the input signals that come from the physical interfaces so that they can be correctly processed; a multiplexer (MUX) that has the task of multiplexing the outputs of the IUs on the internal bus; a demultiplexer (DEMUX) that has the opposite role; the Output Units (OUs) that adapt the internal signals to the format required by the physical transmission interfaces; the Routing Unit (RU) that decides packet routing; the Mux Control Unit (MCU) that, essentially, pilots the MUX in the correct

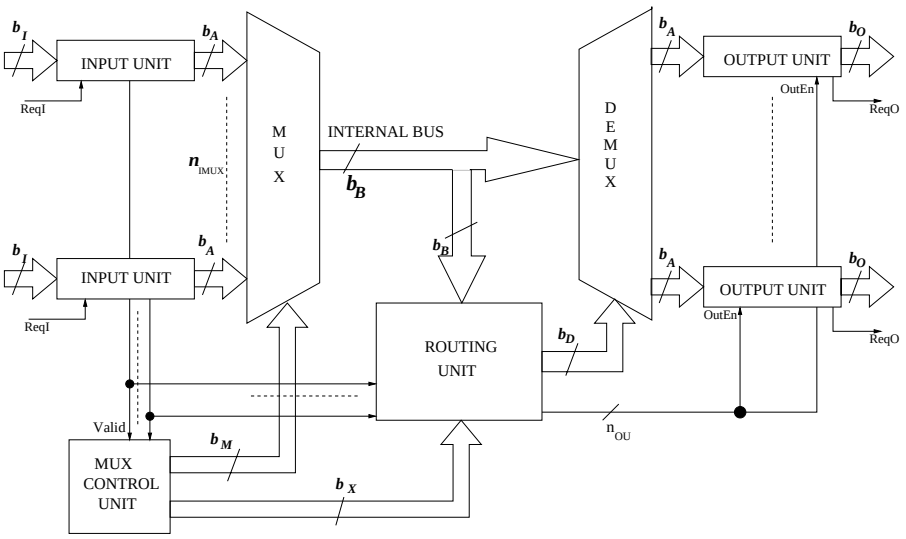

Fig. 3. Switch Block Diagram. It is possible to distinguish: the Input Units (IUs), whose assignment is to adapt the input signals that come from the physical interfaces so that they can be correctly processed; a multiplexer (MUX) that has the task of multiplexing the outputs of the IUs on the internal bus; a demultiplexer (DEMUX) that has the opposite role; the Output Units (OUs) that adapt the internal signals to the format required by the physical transmission interfaces; the Routing Unit (RU) that decides packet routing; the Mux Control Unit (MCU) that, essentially, pilots the MUX in the correct way.

way.

The different modules that constitute the router are now described in greater detail.

4.1 Description of the Input Units

The IUs develop the following functions:

- They change the dimension of the words coming from the physical interfaces (transceiver) to adapt the transmission frequency of the bits (f_i) to the work frequency of the internal bus of the switch (f_B).
- They synchronize the transfer of packets from the transceivers to the MUX.

From an architectural point of view (see Fig. 4), an IU is composed of a shift register (SHR), a register (REG), a comparator (CMP), a counter (CNT) and a finite state machine (FSM) for the control of the signals.

The data coming from the physical interfaces are resized by means of the SHR. In the SHR the data are written in words of b_I bit and they are read in words of b_A bit. It is assumed that b_A is a multiple of b_I. It is supposed, besides, that

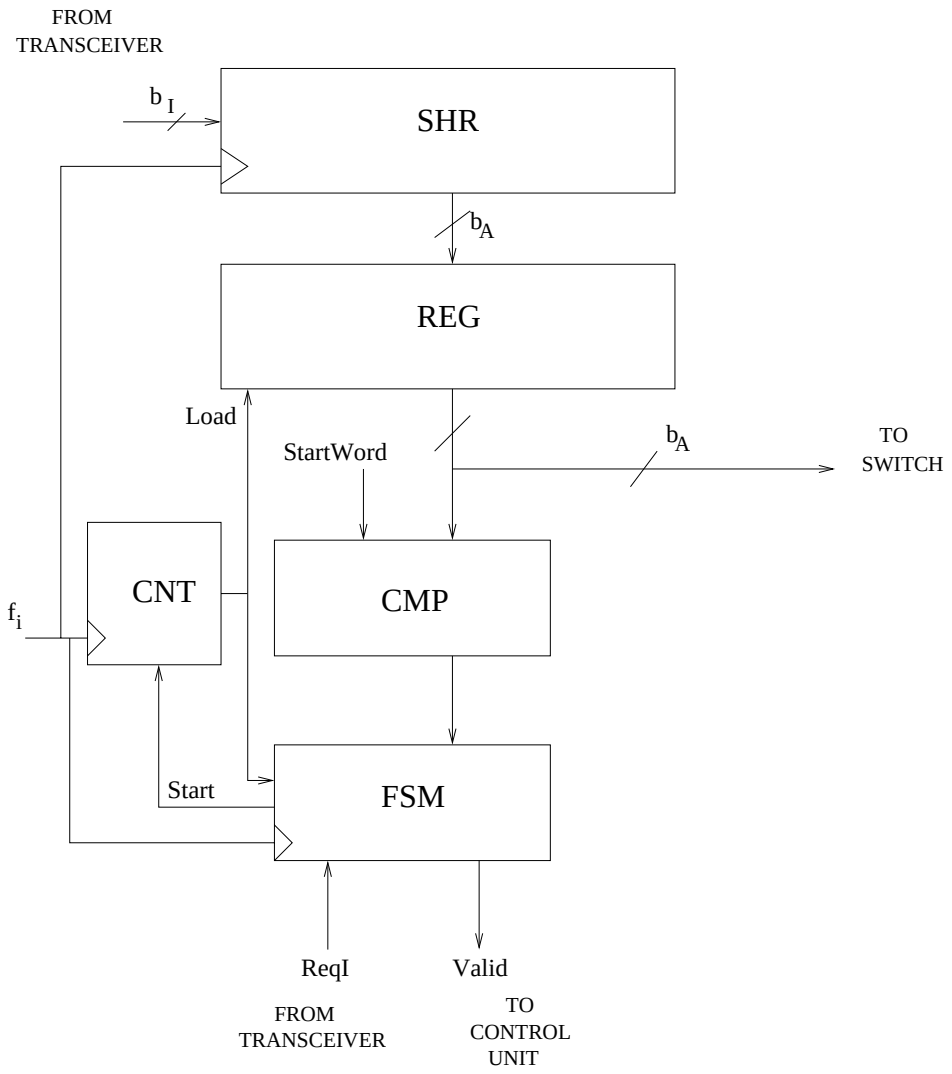

Fig. 4. IU Blocks Diagram

a request signal (ReqI), coming from the transceiver at the entrance of the IU, indicates that the transceiver wants to begin the transfer of bits. During the processing of the words by the units after the IU, the acquisition of the bits must continue. So, the content of the SHR is copied in REG each b_A/b_I clock period beginning from the activation of the ReqI.

For this purpose a counter (CNT) is used. The output word of the REG is compared, by means of the CMP, with a particular configuration of b_A bits, called StartWord, which identifies the beginning of the packet. The presence of the StartWord in the REG is signaled by the FSM to the arbitration units (MCU) with the signal Valid.

The MCU has the task of controlling the MUX and carrying out the necessary arbitration functions in order to put the incoming packets on the internal bus correctly.

The bits coming from the IUs must be transferred to the internal bus. For this purpose it is possible to employ two techniques: space division multiplexing (SDM), in which the bits coming from different IUs are copied onto different lines of the internal bus, and time division multiplexing (TDM), in which the bits of different IUs can be copied onto the same lines of the internal bus but in different times.

It is also possible to use combined techniques, where space division and time division are simultaneously employed (S&TDM). In the next part of the paper we will refer, for simplicity, only to the TDM. It is easy to extend the concepts presented here to the case of S&TDM supposing the subdivision of the internal bus into a certain number of sub-buses which use TDM separately. In this case we can suppose that the MUX and the DEMUX are also composed of a certain number of equal devices each dedicated to the multiplexing or demultiplexing of one of the sub-buses. However, as a consequence of this, the MCU and the RU are more complex.

In the remainder of the article, we will use the term "channel" to identify a set of lines of the internal bus that, in a precise interval of time (time slot), are allocated to a precise word coming from an IU.

The Valid signals, coming from the IUs, permit us to know when a new packet has arrived at the switch. The transition of the Valid signal is interpreted by the MCU as a request of the internal bus from the corresponding IU. Simultaneous requests are opportunely scheduled and managed according to the current availability of the bus, as will be explained successively in subsection 5.1.

The Valid signals allow the dynamic allocation (allocation on demand) of the channels and allow better performances compared to the polling [6].

The channels must be allocated in such a way that all the words of the different packets are obtained. In particular one word of a packet must be transfered onto the internal bus before the following word of the same packet arrives. This means that the internal bus must be of suitable size in terms of number of lines and also of operation frequency. For example, in the hypothesis that all the words that come from the IUs are equally sized (b_A bits each) and that they arrive at the MUX with the same frequency (f_A), if the number of wires in output to the MUX (b_B) is also equal to b_A then, in order not to

lose packets, the frequency of work of the internal bus must be at least equal to $f_B = n_{IMUX} \cdot f_A$ where n_{IMUX} is the number of MUX entries. It is thus possible to realize a TDM of the inputs copying, for each time slot, one word of a different IU onto the internal bus of the switch.

4.3 Description of the Routing Unit

The RU has the principal task of implementing the routing algorithm. It determines the output port of the packet on the basis of the addresses of the destination host, the source host and the host to which the RU belongs. In the case of adaptive and dynamic algorithms, besides the addresses, the selection of the port can also consider the information concerning the state of the hosts and/or of the links of the network and eventually control information contained in the packet. We have substantially identified three possible implementations of the RU:

- hardwired: if the output port can be determined on the basis of a logical-arithmetic calculus, then the RU can be a logical network, more or less complex, that has the addresses and if necessary the state of the network as input and the number of the output port as output.
- by memory: a memory can contain a look-up table between addresses, state and ports. In particular, the pair of source-destination addresses and, if necessary, a suitable codification of the state of the network can be sent to the address bus of the memory; the data contained in the memory cell can be the output port to be selected. This memory can be a ROM in the case of static routing or a RAM in the case of dynamic routing.
- SMART : an intelligent device (microprocessor, neural network, etc.) can elaborate the output port, on the basis of state information [16], [15].

In all three cases, since the packet arrives segmented in words and the addresses are present only in the first words of the packet, it is necessary that, once the output port has been chosen, this is maintained for all the following words of the packet. Additionally, the RU must register the ports already occupied by other packets and use this information on routing because it is not possible to send another packet to a busy port.

4.4 Description of the Output Unit

The OUs have the task of reconverting the size of the words to match the requirements of the physical interfaces. The logical scheme of the OUs is shown in Fig. 5.

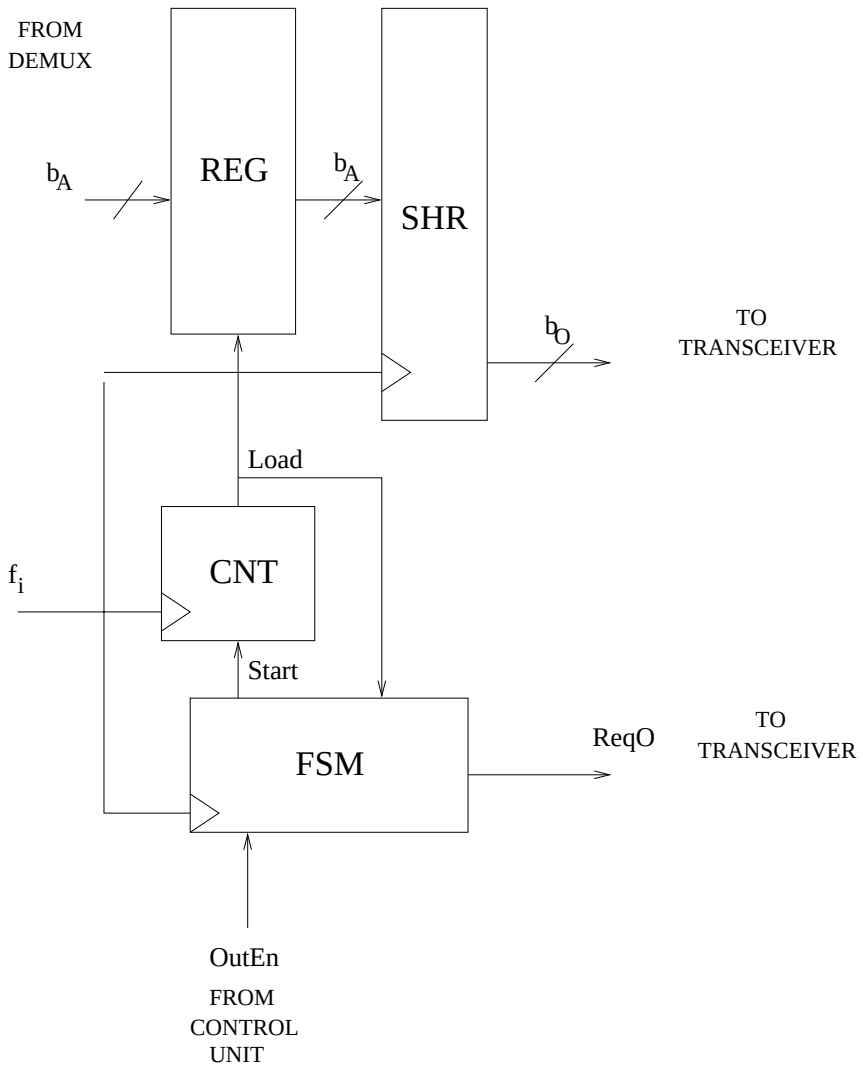

Fig. 5. OU Blocks Diagram

Typically, the transceivers have the same dimension of the bus in transmission and in reception; so, it follows $b_O = b_I$. Additionally, the OU reintroduces the StartWord at the beginning of the packet. The implementation of such a unit requires an architecture similar to that used for the IU. At the beginning, the register (REG) is initialized with the StartWord. When the RU activates the signal OutEn, the content of the REG is copied in the SHR. This transfers the bits to the transceiver in groups of b_O bits. The following words coming from the DEMUX, must be acquired every b_A/b_O clock period, and, for this reason, a counter (CNT) is necessary. The FSM manages the control signals. By means of the signal ReqO, it advises the transceiver that new bits must be transmitted.

5 Design Flow

Subsequently, we depict the design flow we have proposed by means of which it is possible to implement a router for reconfigurable networks.

The applicability field of such a flow is restricted to VCT and WH switching techniques. This restriction is due to the application domain of our research and in consideration of the theoretical results available in literature. The objective of the research is the development of a network for a cluster of hosts where the propagation time of the signals on the links is inferior to the transmission time of the packet and the signal to noise ratio is high. Under these conditions, it is known [2] that the performances in terms of latency and bandwidth are better for VCT and WH than SF.

With reference to Fig. 6, the first step is to characterize the topology/topologies for which we want to use the router. This has the aim of determining the maximum number of links for each host, i.e. the maximum degree of the topologies (g_{max}). In fact, while it will be possible to program the logical part of the router many times, it will not be possible to change the physical part. In particular, the number of pins of the FPGA and the number of physical interfaces (transceivers) present on the board, will necessarily be fixed once the Printed Circuit Board (PCB) has been established. Concerning the choice of the more suitable topologies for a particular application or for particular requirements (transmission bandwidth, latency, etc.) see Section 2.1. Besides, we suppose that the transmission capacity of the links (C_i) is known and that it is the same for both the directions of signal propagation. Another constraint is the maximum clock frequency that can be used for the operation of the router (f_{WM}).

5.1 Router Design

On the basis of the data formerly estimated, it is necessary to size the input and output units (IUs/OUs) that is to say, to establish the dimensions and the frequencies of the different buses. Typically, the physical interfaces of the links (transceivers) can accept the bits (in transmission) and can forward them to the router (in reception), serially or in parallel, as words of a fixed number of bits [17]. As a consequence of this, the number of bits in entry to the input unit, b_I, is determined by the particular transceiver selected. The transceiver receives from the link a flow of C_i bps and transmits it to the IU, as words of b_I bit at the frequency of f_i words per second. Thus we have $C_i = f_i \cdot b_I$. The number of bits that exit from the IUs (b_A) and the frequency at which they must be sent to the multiplexer (f_A), depends on the type of multiplexing that

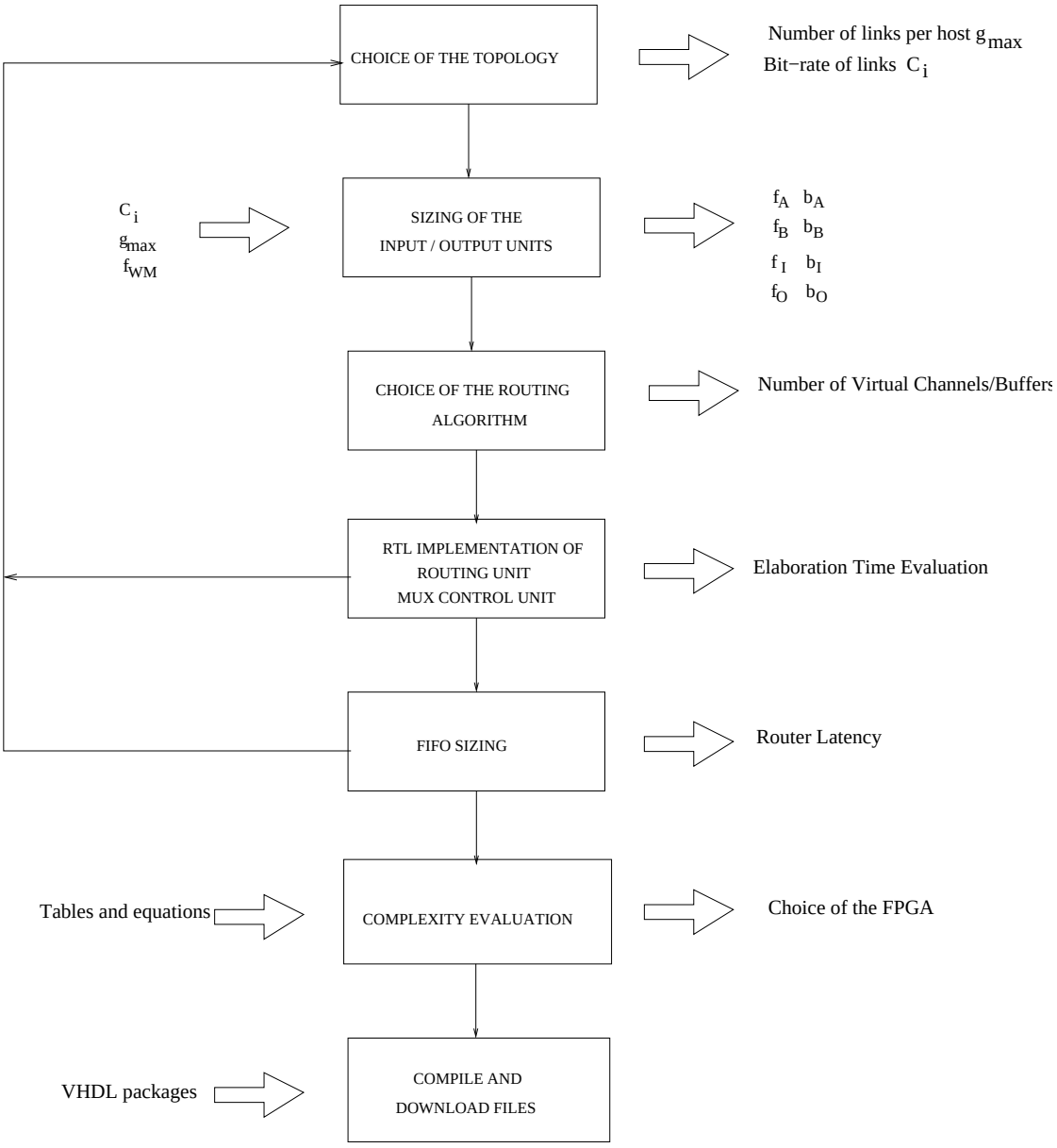

Fig. 6. Design Flow

is used and on the work frequency of the router (f_B), as well as the number of lines (b_B) that constitute the internal bus of the switch. In TDM, for each time slot, the bits coming from an IU are copied onto the internal bus. If all the links have the same capacity C_i and if the same type of transceiver is used for each link, assuming a time slot of $1/f_A$ and a number of time slots equal to the number of entries of the MUX (n_{IMUX}), these relationships follow:

$$\begin{aligned} b_A &= b_B \\ C_i &= f_i \cdot b_I = f_A \cdot b_A \\ f_B &= f_A \cdot n_{IMUX} \leq f_{WM} \end{aligned} \tag{2}$$

From which:

$$b_A \geq \lceil \frac{C_i \cdot n_{IMUX}}{f_{WM}} \rceil$$
$$f_B = \frac{C_i \cdot n_{IMUX}}{b_A} \tag{3}$$

where the symbol $\lceil \rceil$ denotes the operator of superior integer. It is possible and convenient to choose the least integer of b_A that satisfies the inequation. In fact, due to the presence in the IUs of a shift register of b_A bit, each router will delay the arrival of the packet by at least b_A clock periods. In other terms, in conditions of low traffic, the latency of the network is proportional to b_A.

Otherwise, the dimension of b_A could also be selected in order to have all the necessary bits to do the elaboration of the routing on the internal bus of the switch. Later, we will assume this hypothesis. Moreover, as has been said on the internal architecture of the IUs, in subsection V-A, b_A must be a multiple of b_I.
Once determined b_A and f_B, we need to verify that the units, dimensioned as explained above, can support the respective frequencies, otherwise we need to change the target technology.
As already pointed out, in the more general case in which one desires to use a S&TDM, it is possible to analyze the switch as several parallel TDM units. Therefore, the following results allow an estimation of the router complexity also in such a case.

Afterwards, according to the number of virtual channels, in the case of WH switching, or the number of buffers, in the case of VCT switching, we proceed with the determination of the dimensions of the external memory necessary to the routing algorithms that we want to use. For example, Duato's algorithm [18] requires three virtual channels for each physical link. Since each virtual channel must be able to save a flit, a memory of $3 \cdot g_{max}$ flits is necessary.
An implementation with a Register Transfer Language (RTL), i.e. VHDL, allows the evaluation of the complexity of the routing algorithm (i.e. number of Logic Cells and Flip-Flops of MCU and RU) and the estimation of the elaboration times.

From this estimation it is possible to dimension the elements of the datapath suitably. In particular, it is possible to decide whether or not it is necessary to use the pipelining technique [19] inside the RU, usually the slower unit, or in other modules. The pipelining is not necessary if the clock period is long enough when compared to the time required to elaborate the routing algorithm and to transfer the bits from the IU to the appropriate OU. Vice versa, it is necessary to evaluate the number of pipe stages.

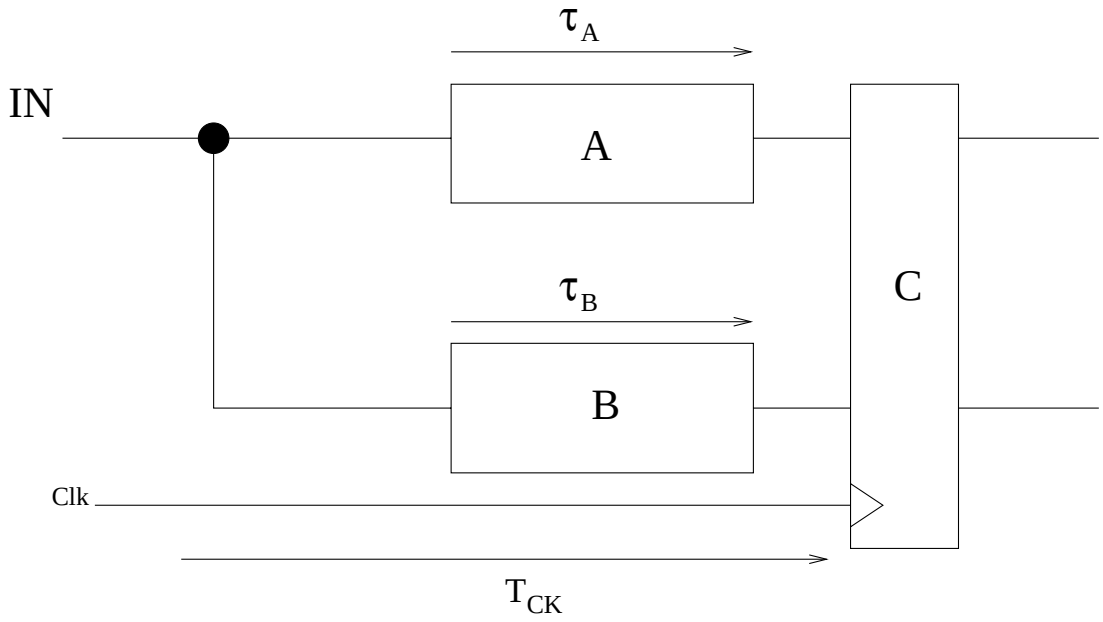

Fig. 7. The figure shows a register (C) and two blocks of combinational logic (A,B) with different propagation times (τ_A,τ_B) in order to show the usage of pipelining.

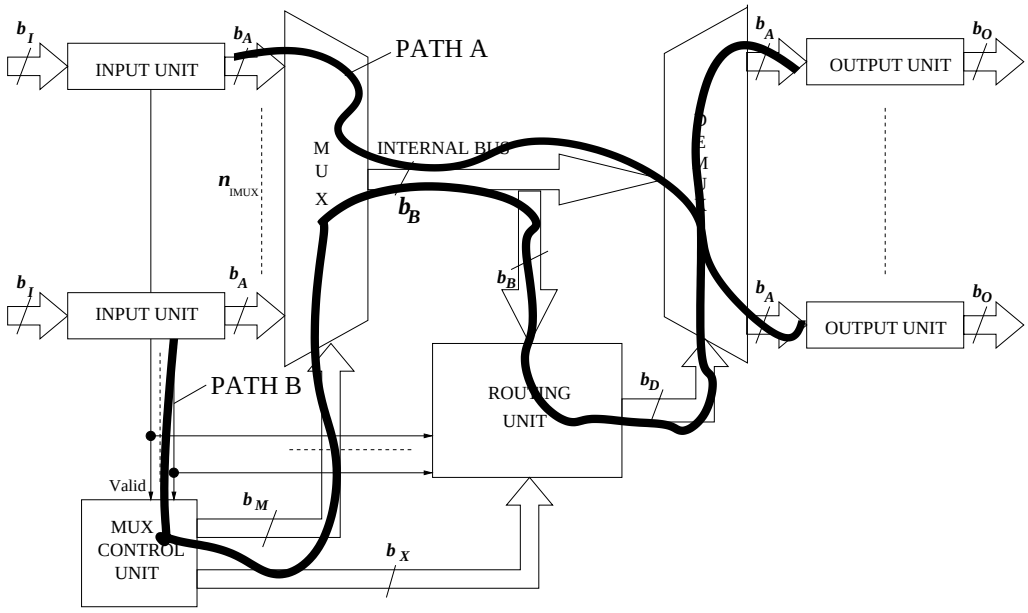

Fig. 8. Critical paths. In the figure the critical paths of the switch are displayed by means of wide-line, in analogy to Fig. 7

The best number of stages can be determined by the following reasoning.

Let us suppose we have the circuit in Fig. 7 and for simplicity we assume that the wires have no delay. Blocks A and B are combinational and introduce respectively the delays τ_A and τ_B, while block C is sequential and ideal (the setup, hold and clock to out times are assumed to be null). If both τ_A and τ_B are inferior to the clock period T_{ck}, then there is no problem because the

bits in entry are propagated across blocks A and B and they arrive at block C before the arrival of the rising edge of the clock. In the case that one or both the delays are greater than T_{ck}, it is necessary to use the pipelining technique. With such a technique it is necessary to divide A and B into the same number of stages, in such a way that the slower stage introduces a delay inferior to T_{ck}. A register is inserted between one stage and the following. Note that, in order to determine the delays of the different stages, it is necessary to consider also the delays introduced by the registers.

Fig. 8, which is a copy of Fig. 3, can be evaluated in the light of what has just been explained. Blocks A and B, in this case, are not disjointed. They are constituted by the units crossed by the two paths displayed by means of wide-line. The second path (path B) is the longer therefore it is necessary to divide this block into a certain number of pipelined stages so that each stage has a delay inferior to the clock period $\frac{1}{f_B}$. If the delay due to the MUX and to the DEMUX is denoted by τ_{io} and the delay necessary to the elaboration of the routing, due to the MCU and to the RU, by τ_e, we have $\tau_{io} = \tau_A \leq \tau_B = \tau_{io} + \tau_e$. Therefore the least number of pipelined stages is given from the smaller integer n_R that satisfies the disequation $\frac{\tau_e + \tau_{io}}{n_R} \leq \frac{1}{f_B}$. Note that the subdivision in pipelined stages of the MCU involves the insertion of a First In First Out (FIFO) queue, between the IUs and the MUX. The number of registers in this FIFO is equal to the number of those in the MCU and each register of the FIFO is composed of as many Flip-Flops as the input wires of the MUX. Likewise, the subdivision in pipelined stages of the RU involves the insertion of a FIFO between the MUX and the DEMUX. This FIFO will have a number of registers equal to the number of those in the RU and each register will be composed of b_B Flip-Flops. The following formulas refer to the more common case in which the MCU does not need a pipeline. Therefore, with the term FIFO, we mean the possible registers between the MUX and the DEMUX.

5.2 Router Latency

Now, the minimum crossover time of the router (*router latency*) can be estimated. Before proceeding with calculations, we wish to underline that the IU and OU units work at clock frequency f_i because they are synchronized with the transceivers, while the MCU and RU units work at frequency f_B. The main components determining the router latency are listed below:

- *Delay introduced by IU.* In Subsection 4.1, we have described all of the operations executed by the IU. In this subsection, we are interested in considering the delay it introduces. Such a delay derives from the necessity to acquire the part of the packet containing the information required for the

routing.

For this reason, it is important to take into account the structure of a packet (see Section 3) that we briefly summarize here. The first bits constitute the StartWord pointing out the beginning of the packet itself. Afterwards, there are the destination and source addresses and any information related to the flow control. Lastly, the data and the FCS complete the packet.

If the routing can be performed on the basis of only the destination address, then it will be sufficient to acquire only the StartWord and destination address. The StartWord can be eliminated from the IU and reintroduced by the OU; so, we have a virtually null delay for its transfer. The destination address is supposed to be equal or inferior to b_A bits (if it is bigger, we can increase b_A to satisfy this condition). Therefore, we have a delay of $\frac{b_A}{C_i}$ for the acquisition of the destination address in the IU. To this delay, we must add a subsequent clock period ($\frac{1}{f_i}$) for the synchronization of the IU and the MUX. Such a clock period allows the content of the shift register SHR to be memorized in the REG so that the IU can acquire the following sequence of bits while the MUX is elaborating the preceding bits. In summary, the IU introduces a total delay of $\frac{b_A}{C_i} + \frac{1}{f_i}$.

- *Delay introduced by RU and MCU.* As we said in the previous subsection it is equal to $\frac{n_R}{f_B}$.
- *Delay introduced by OU.* A time interval equal to $\frac{1}{f_B}$ is required for the acquisition of the word present on the internal bus. A subsequent clock period ($\frac{1}{f_i}$) is necessary to transfer the first word, i.e. the first b_O bits, to the transceiver. In summary, the OU introduces a total delay of $\frac{1}{f_B} + \frac{1}{f_i}$.
- *Statistical delay introduced by TDM.* In the worst case, it is possible that the packet is granted the last time slot. In that case, it is necessary to add a further delay, equal to $\frac{n_{IMUX}}{f_B}$. In the average case, the delay is $\frac{n_{IMUX}}{2 \cdot f_B}$.

By summing the contributions described above and expressing them by the capacity of transmission of the links, we obtain the overall mean delay introduced by the router:

$$\tau_R = \frac{1}{C_i} \cdot [2b_I + b_A(1.5 + \frac{n_R + 1}{n_{IMUX}})] \tag{4}$$

To obtain the total delay of the router, the delay introduced by the transceivers in transmission (τ_{Tx}) and in reception (τ_{Rx}) has to be added to the result above. Finally, the total delay is equal to $\tau'_R = \tau_R + \tau_{Tx} + \tau_{Rx}$.

It is possible that the delay obtained is too high in comparison to the design goals. In fact, in conditions of low load, the mean time between the exit of the first bit of the packet from the transmission host and the arrival of the same bit at the receiving host (*network setup latency*) is linked to the delay

inserted by each router through the relationship $lat_{setup} = \tau'_R \cdot d$. If this time turns out to be too high for the requirements imposed by the system, it will be necessary to go back to the design flow and select a different topology (to reduce g_{max} and likewise b_A) or a different target technology (to increase f_{WM} and to reduce n_R and b_A).

On the contrary, if the results obtained are acceptable, we go to the analysis of the complexity of the units with the aim of determining the number of FlipFlops, LogicCells and pins necessary for the integration of the router in a FPGA. For this purpose, in the next subsection the method of analysis and the necessary formulas to estimate the router complexity will be illustrated.

5.3 Router Complexity

In Table 1 the results of our empirical study are shown; from their analysis it is possible to estimate the complexity and the timings of the different units constituting the router. Such values have been obtained by means of the tool MAX+PLUS II ver. 9.21 [20] of Altera Corporation and refer to FPGA of the series FLEX10KA-1 [21]. All the values shown in the table have been normalized. In particular, the values related to Logic Cells (LCs) are normalized with respect to the number of LCs required to implement a MUX with 8 entries each with 8 bits ($LC_n = 48$), while the frequencies are normalized with respect to the maximum frequency of transfer between two registers ($f_n = 167$ MHz).

As we expected, when the dimensions of the buses (b_A, b_I) and the number of inputs (n_{IMUX}) increase, the number of LCs necessary to the implementation of the units increases too, and the maximum operative frequency (f_{max}) of the units decreases. Additionally, according to the optimization (Opt) selected in the synthesis phase, it is usually possible to increase the maximum frequency ($Opt = Speed$) of a unit using more LCs. Vice versa, if a lower operative frequency is acceptable we can reduce the number of LCs necessary ($Opt = Area$). From Table 1, it is possible to obtain the real values (not normalized) by multiplying them for the relative normalization factor. As an example, let us calculate the number of LCs necessary for a MUX with 5 entries and 16-bit buses in the case of optimization for Area. It is sufficient to read the coefficient in the line related to the MUX with n_{IMUX}= 5 and b_A= 16 and in the column $LC_{Opt=Area}$ and multiply it for the normalization factor LC_n. We obtain $LC_{MUX} = 1.333 \cdot 48 = 64$.

In order to calculate the complexity and the timings also for configurations not present in Table 1, we have interpolated the obtained data with quadratic relations of the kind

$$z = c_1 \cdot x^2 + c_2 \cdot y^2 + c_3 \cdot xy + c_4 \cdot x + c_5 \cdot y + c_6 \tag{5}$$

Depending on the object unit, x and y represent the parameters b_A, n_{IMUX}, b_I, b_O; z is the number of LCs or f_{max} and c_i are coefficients. Table 2 shows the coefficients c_i obtained for the differents units and the maximum relative error, in percent, ($err\%$) that is committed when using the formula. The error refers to the results shown in Table 1.

Once the dimensions of the buses have been established, it is possible to estimate the number of LCs and the maximum frequency of the different units, by means of the coefficients shown in Table 2. For example, the number of LCs needed for a 5-input MUX with 16-bit buses, optimized for Area, will be $LC_{MUX} = 0.67 \cdot 16 \cdot 5 + 0.67 \cdot 16 = 64$. It should be noted that if the maximum frequency does not respect the design goals it is necessary to change the target technology (FPGA family).

If working frequencies match the design goals then, to choose the particular FPGA model necessary for the implementation of the router, the following values have to be considered:

- total number of Logic Cells (LC_{tot}):

$$\begin{aligned} LC_{tot} = n_{IU} \cdot (LC_{IU} + LC_{OU}) + \\ +LC_{MUX} + LC_{DEMUX} + LC_{RU} + LC_{MCU} \end{aligned} \tag{6}$$

- total number of Flip-Flops (FF_{tot}). It is the contribution of several terms and can be calculated from the following equations:

$$FF_{IU} = FF_{OU} = 2 \cdot b_A + \lceil log_2(b_A/b_I) \rceil + 3 \tag{7}$$

$$FF_{FIFO} = b_B \cdot n_R \tag{8}$$

$$FF_{tot} = 2n_{IU} \cdot FF_{IU} + FF_{FIFO} + FF_{MCU} + FF_{RU} \tag{9}$$

- total number of pins (PIN_{tot}). It depends on the number of the control signals (clk, reset, handshake signals with the transceivers and with the memory etc.) and the signals necessary to interface the switch with the transceivers and with eventual memory modules.

As a pratical rule the use of a FPGA device with a quantity of resources about 10-20% greater than the values obtained is suggested. This makes eventual retouches easier and simplifies the "routing" (here intended as the interconnection of the logic cells).

From these data, it is possible to choose a FPGA. Once the FPGA has been selected, we can compile the codes and download the configuration files into the FPGA.

6 Conclusions and Future Works

In this paper, we have presented the design flow and the architecture of a router that can be employed in multidimensional networks. In particular, the design flow presented permits the correct design of the different units of this architecture. We have also given some experimental data and empirical equations that allow the complexity and the performance of the router to be estimated before its actual realization. As future prospectives for our research we are studying the design of a flexible router. Just considering one of the fundamental aspects in the use of the FPGAs, i.e. the possibility of reprogramming them in system, we are thinking of re-programming the hardware on-line. So, the implemented router will be better matched to the distributed software requirements that we wish to run on our cluster. Further, we are planning to realize a network interface with our router architecture and to develop a routing algorithm based on techniques of artificial intelligence.

References

[1] K.J. Liszka et al., "Problems with comparing interconnection networks: Is an Alligator Better Than an Armadillo?," *IEEE Concurrency*, Dec. 1997.

[2] A. S. Tanenbaum, *Computer Networks*, Prentice Hall, 1991.

[3] P. Kermani, L. Kleinrock, "Virtual cut-through: A new Computer communication switching technique," *Computer Networks*, September 1979.

[4] W.J. Dally, C.L. Seitz, "The Torus Routing Chip," *J. Distributed Computing*, Vol.1, no.3, pp. 187-196 1986.

[5] W. Dally, "Virtual-Channel Flow Control," *IEEE Trans. Parallel and Distributed Systems*, March 1992.

[6] W. Stallings, *Data and Computer Communications*, Prentice Hall, 1997.

[7] A.A.Chien, "A Cost and Speed Model for k-ary n-cube Wormhole Routers," *IEEE Transaction on Parallel and Distributed Systems*, February 1998.

[8] Cray Research Inc., *Performance of the Cray T3E Multiprocessor*, on Cray's web page www.cray.com/products/systems/crayt3e/paper1.html.

[9] PCRCW, "Parallel Computer Routing and Communication Workshop," .

[10] J.Rexford, W.Feng, J.Dolter, K.G.Shin, "PP-MESS-SIM: A Flexible and Extensible Simulator for Evaluating Multicomputer Networks," *Trans. on Parallel and Distributed Systems*, 1997.

[11] E. Fleury, P. Fraigniaud, "A General Theory for Deadlock Avoidance in Wormhole-routed Networks," *IEEE Trans. Parallel and Distributed Systems*, July 1998.

[12] S.F. Nugent, "The iPSC/2 Direct-Connect Communication Technology," in *Proc. Conf. Hypercube Concurrent Computers and Applications*, 1988.

[13] H. Sullivan, T.R. Brashkow, "A large scale Homogeneus Machine," in *Proceedings of the 4th Annual Symposium on Computer Architecture*, 1977.

[14] G. Campobello, G. Patané, M. Russo, "Parallel CRC Realization," *in press on IEEE Trans. on Computers*, 2003.

[15] 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*, January 1999.

[16] H. Kurokawa, C.Y. Ho, S. Mori, "The neural network approach to a parallel decentralized network routing," *Neural Networks*, n.11 1998.

[17] National Semiconductor, *DP83846A DsPHYTER, Single 10/100 Ethernet Transceiver*, 2000.

[18] J. Duato, "On the Design of Deadlock-Free Adaptive Routing Algorithm for Multicomputers: Design Methodologies," *PARLE (1)*, 1991.

[19] J. Hennessy, D. Patterson, *Computer Architecture a Quantitative Approach*, Morgan Kaufman, 1995.

[20] ALTERA Corporation, *Max+Plus II Getting Started*, 1997.

[21] ALTERA Corporation, *Flex10K: Embedded Programmable Logic Family*, a-ds-f10k-04.01 edition, June 1999.

List of Tables

Table 1
Logic Cells and f_{max} of the different units: The values in the columns relative to the number of Logic Cells (LCs) are normalized with respect to the number of LCs needed for a MUX 8x8 $LC_n = 48$, the values relative to the frequencies are normalized with respect to the maximum frequency of transfer between two registers $f_n = 167$ MHz

Unit	n_{IMUX}	b_A b_B	b_I b_O	LC *Opt Area*	f_{max} *Opt Area*	LC *Opt Speed*	f_{max} *Opt Speed*
IU	-	8	1	0.479	0.581	0.875	0.623
	-	12	1	0.708	0.455	0.958	0.569
	-	16	1	0.875	0.329	1.229	0.443
	-	20	1	1.104	0.299	1.542	0.479
	-	8	4	0.375	0.748	0.542	0.904
	-	12	4	0.583	0.629	1.104	0.832
	-	16	4	0.771	0.485	1.479	0.754
	-	20	4	0.979	0.299	1.437	0.503
OU	-	8	1	0.470	0.736	0.437	0.796
	-	12	1	0.604	0.701	0.583	0.641
	-	16	1	0.729	0.653	0.708	0.748
	-	20	1	0.896	0.653	0.917	0.689
	-	8	4	0.375	0.748	0.396	0.748
	-	12	4	0.562	0.665	0.542	0.671
	-	16	4	0.646	0.748	0.646	0.653
	-	20	4	0.812	0.683	0.792	0.557
MUX	5	8	-	0.667	0.934	0.667	0.934
	5	12	-	1.000	0.934	1.000	0.934
	5	16	-	1.333	0.934	1.333	0.934
	5	20	-	1.667	0.934	1.667	0.934
	8	8	-	**1.000**	0.713	0.833	0.796
	8	12	-	1.500	0.713	1.250	0.736
	8	16	-	2.000	0.695	1.667	0.808
DEMUX	5	8	-	0.917	0.581	1.187	0.826
	5	12	-	1.333	0.563	1.604	0.778
	5	16	-	1.750	0.575	2.104	0.665
	5	20	-	2.167	0.587	2.604	0.689
	8	8	-	2.062	0.976	2.687	0.838
	8	12	-	2.979	0.527	3.937	0.575

Table 2. Interpolation functions

Unit	x	y	z	c_1	c_2	c_3	c_4	c_5	c_6	$err\%$
IU	b_A	b_I	$LC_n \cdot LC_{Opt=Area}$	0	-0.70	-0.02	2.47	1.88	2.52	2.6
			$f_n \cdot f_{max,Opt=Area}$	0.04	-9.02	-0.73	-4.42	62.15	79.94	10.5
			$LC_n \cdot LC_{Opt=Speed}$	-0.14	-0.26	0.32	6.35	-3.31	-4.08	17.12
			$f_n \cdot f_{max,Opt=Speed}$	-0.12	-3.89	-1.01	1.96	45.82	58.25	15.5
OU	b_A	b_O	$LC_n \cdot LC_{Opt=Area}$	0.01	-1.32	0.01	1.42	5.23	6.87	6.4
			$f_n \cdot f_{max,Opt=Area}$	0.07	-14.22	0.26	-3.48	68.91	89.70	6.8
			$LC_n \cdot LC_{Opt=Speed}$	0.02	-0.82	-0.11	1.33	4.55	5.90	3.8
			$f_n \cdot f_{max,Opt=Speed}$	0.10	-13.49	-0.53	-3.22	71.37	92.59	12.3
MUX	$b_A = b_B$	n_{IMUX}	$LC_n \cdot LC_{Opt=Area}$	0	0	0.67	0.67	0	0	0
			$f_n \cdot f_{max,Opt=Area}$	-0.01	-4.79	-0.14	1.09	51.33	16.80	0.72
			$LC_n \cdot LC_{Opt=Speed}$	0	0	0.33	2.33	0	0	0
			$f_n \cdot f_{max,Opt=Speed}$	0.10	-5.23	0.21	-3.82	57.09	18.69	5.11
DEMUX	$b_A = b_B$	n_{IMUX}	$LC_n \cdot LC_{Opt=Area}$	0	0.19	2.00	-5.00	-0.14	-0.05	0
			$f_n \cdot f_{max,Opt=Area}$	0.08	6.07	-6.08	28.35	-8.13	-2.79	0.8
			$LC_n \cdot LC_{Opt=Speed}$	0.06	-0.73	3.27	-12.38	7.39	2.42	0.8
			$f_n \cdot f_{max,Opt=Speed}$	0.19	-1.88	-2.43	4.72	43.98	14.34	4.59