A Reprogrammable CBR and VBR Traffic Generator/Monitor on FPGA *

J. Claver
Dept. de Informàtica
Universitat de València
Campus de Burjassot
46100 Burjassot
jclaver@uv.es

P. Agustí,M. Canseco,G. León
Dept. Ingeniería y Ciencia de los Computadores
Universitat Jaume I
Campus de Riu Sec
12071 Castellón
leon@uji.es

Abstract

Nowadays, high performance System/Local Area Networks (SAN/LAN) are filled by heterogeneous traffic, consisting of information flows with different bandwidth and latency requirements. The bottleneck produced by the throughput of network processing elements (servers, routers,...) and the increasing bandwidth of links, makes it necessary to propose new designs for these network components.

The MMR (and its simplified version, the SMMR), a router that supports QoS, is a very well-known proposal in this area. In this article we propose the architecture and implementation of a hardware reprogrammable traffic multimedia Generator/Monitor to study these sorts of routers under different traffic conditions and server models. The use of this generator makes it possible to reduce the complexity of routers and to optimize their design The Generator/Monitor is modular and has been implemented inside an FPGA. Using a high level hardware programming language such as the Handle-C makes the design and updating process easier, as it was made before with the SMMR. The result is a highly parameterizable and scalable platform, which allows prototyping the communication system of a high speed LAN/SAN environment on a FPGA device.

*This work has been jointly supported by the Spanish MEC and European Comission FEDER funds under grants "Consolider Ingenio-2010 CSD2006-00046" and "TIN2006-15516-C04-02"

1 Introduction

Current high performance System and Local Area Networks (SAN/LAN) are filled by heterogeneous traffic, consisting of information flows with different bandwidth and latency requirements. So, beside the *best effort* (BE) traffic generated by applications like FTP or e-mail, we found traffic with quality of service requirements (QoS) generated by multimedia applications: web applications, interactive simulations, virtual meetings, video, etc. This traffic is generated and consumed by servers within a large list of applications, while the central routers distribute them properly. Both servers and routers have to cooperate to guarantee the requirements of QoS streams mixed with the BE traffic.

The relative low performance of routers with respect to the incredible speed and the bandwidth increase on links, intensified by the use of optical links, make routers the bottleneck of throughput in current high performance networks. For this reason, new technologies to improve the response time and the way in which the information is stored and transmitted are needed for these network components.

The Multimedia Router (MMR) [1] architecture is a proposal of an input queued router that supports QoS requirements, via hardware, into an interconnection compact component. It is conceived to be used in cluster environments and high speed LANs. The MMR is connection oriented and requires a high num-

ber of virtual channels (VC) per input port (128). Hence, the router needs a complex link scheduler for every VC, which manages, updates and selects packets to be sent each time. In order to reduce the MMR complexity, mantaining the QoS guarantees, a simple version, the SMMR (Simple MMR) [2], is proposed. This router is implemented on a FPGA and its study has helped to optimize some MMR components [3].

According with these proposals, we have suggested the design and implementation of a reprogrammable Generator/Monitor compatible with the characteristics of the SMMR to evaluate the behavior of this router and to propose improvements in its design, taking care of the injected traffic characteristics . For that, a device that provides different information streams for every SMMR port is necessary. The Generator/Monitor has to be parameterizable for different router implementation sizes, charge situations and distribution of traffic flows. It is also important that the Generator/Monitor has different scheduling models in order to evaluate the router when the network servers have different behaviors. It is no less important it is to monitor this device, which must be configurable and not intrusive, not affecting the particularities of the generated traffic. Previous FPGA-based traffic generators (like in [4] for an ATM network) generate traffic for a unique link, and has no monitor module. In our design we configure the number of links used by the Generator/Monitor and can test the behavior of all data streams.

One of the most important conditions to be accomplished by the Generator/Monitor is the scalability. If we increase the number of links and data streams per link (in this case it is equal to the number of virtual channels) the device must be able to grow in a reasonable form and to keep the properties of the generated traffic. To make it possible, we decided to use a hardware design with parallelism characteristics and innate independency in this sort of implementations. We have used reconfigurable architectures, due to the demonstrated versatility in recent research about networks on chips (NoC) [5].

The wide field opened by the FPGAs and the high level hardware specification/programming languages today, allow us to build this prototype in an easy and efficient way. Our proposal is the first step in the design of a NoC emulation platform on FPGA, like in [6], but in which both servers and routers have QoS requirements. A first version of the Generator/Monitor has been developed [7], but only supplies CBR (Constant Bit Rate) traffic. This new release generates both CBR and VBR (Variable Bit Rate) traffic with the presence of BE traffic.

The rest of this paper is organized as follows. The following section addresses the characteristics of the considered network architecture and the traffic generated. In section 3, the architecture and *hardware* of the Generator/Monitor is presented, where the organization, configuration and the most important modules are described. The results obtained after the Generator/Monitor evaluation are reported in section 4. In the last section we highlight the most important results obtained on this work and overview further lines of research.

2 Network architecture and generated traffic

Our Generator/Monitor has been designed to generate different traffic distributions on a high performance Local and System Area Networks (LAN/SAN), and server clusters. In these sorts of networks heterogeneous traffic, best effort and with QoS requirements, CBR and VBR, coexist.

In the network architecture used for the MMR router, the minimum information unit with flow control, *flit*, is composed of a fixed size of 64 *phits*, where a *phit* is a word width in the links (16 bits). The network establishes the QoS connections using an adaptative algorithm connection oriented of minimum path search called *Exhaustive Profitable Backtracking* and the flow control is carried out by PCS (Pipeline Circuit Switching). BE traffic is not connection oriented and the control traffic is carried out by VCT (Virtual Cut Through).

The communication through the links is car-

ried out in a serial and synchronized way, in blocks of 1040 bits (64+1 *phits*), named *flit cycles*. After every *flit cycle*, 64 *phits* of information, a data *flit*, followed by a flow control *phit*, is transmitted. This last *phit* corresponds to the sending of a flow control credit, that coincides with the SMMR reconfiguration stage. The use of credits establishes a back-pressure mechanism on the data stream to prevent channel saturation and lost data, in a similar way as in InfiniBand.

As explained before, traffic in these networks is heterogenous and can be classified, according to their QoS characteristics, as follows:

CBR traffic.

This kind of traffic has high QoS requirements of bandwidth and latency. It is the traffic generated by applications using non-compressed video and audio data streams.

The CBR traffic is transmited in a sequential way using a pattern called CBR rate. This CBR rate (C) means that for each C flit cycles, an information flit of this stream is sent. The bandwidth BW_i reserved for a CBR traffic can be expressed as

$$BW_i = \frac{1}{C} BW_T,$$

where BW_T is the total link bandwidth.

VBR traffic.

The VBR traffic is usually generated by applications sensible to delay and jitter, which injects traffic in a variable cadence. This is the typical traffic generated by multimedia applications using compressed video and audio data streams.

In order to transmit the VBR traffic, the bandwidth is partitioned into two different kinds of bandwidth reserves: one called CBR_a (or average CBR bandwidth) and a another called PBR (Peak Bit Rate). The CBR_a reserve is transmitted like a CBR traffic, and the PBR is transmitted, as soon as possible, using a BB(Back-to-back) algorithm [1].

A VC with a reserve for VBR traffic is always candidate to transmit if it remains PBR bandwidth reserve. If the VC is elected and the CBR_a rate shows that is ready to transmit, a CBR flit is sent in order to mantain its CBR behavior. A VC with VBR configuration shows a CBR behavior when the PBR reserve is empty.

BE traffic.

The BE traffic uses the available bandwidth, and has no QoS requirements on bandwidth, delay and jitter. This kind of traffic is asigned to a VC and is sent when any other VC has not data able to transmit.

3 The Generator/Monitor

The architecture and hardware of the Generator/Monitor is shown in Fig. 1. It is possible to observe that both Generator/Monitor and SMMR router are implemented on the same FPGA. All the network system is implemented in a unique FPGA to simplify the design and evaluation of the Generator/Monitor. The traffic generated by the Generator/Monitor is configured by a table stored in an SRAM external to the FPGA. The network could be implemented on different FPGAs, using one for the router and another for the Generator/Monitor. This solution will only be necessary when the size of components makes their inclusion in one FPGA impossible.

The Generator/Monitor has as many traffic generator modules (GENERATOR in the Fig. 1) as has the SMMR router has links (ports). Each GENERATOR module manages as many traffic streams as VCs were defined, and they must be equal to the number of VCs in the SMMR. A traffic generator has two functions; it is a source and a drain (ideal at the moment) of traffic. For each VC the following parameters in the external SRAM configuration table are defined : the kind of connection (STATE), the VBR reserved bandwidth (CBR_a and PBR) reserved for the data stream (if the PBR reserved bandwidth is 0 means that the data stream has CBR traffic), the SMMR input and output VC and ports (PORT_OUT, CV_OUT, PORT_IN, CV-_IN) used, the sending rate of flits (T_DELAY), and the initial delay in which the flits are sent (I_DELAY) (this parameter is only used to continue an unfinished experiment previously started).

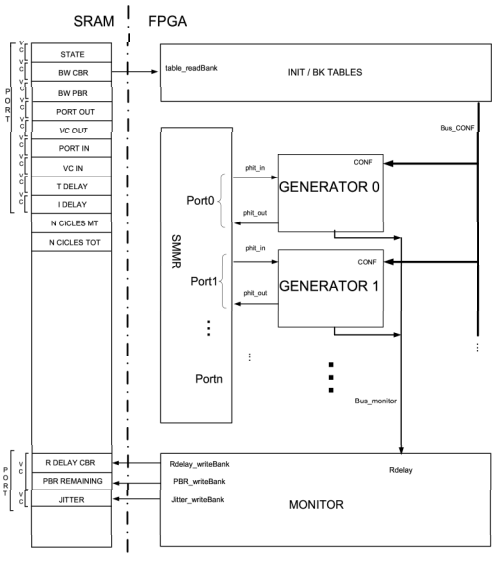

Figure 1: Architecture of the Generator/Monitor system and FPGA interface.

3.1 Monitor

The monitor module works independently of the traffic generators, but synchronized with them. A traffic generator begins the communication, reserving a virtual channel for each connection. The process starts when the generator sends a "connection flit" to the router . This connection flit goes through the router and reaches another generator module, which will send a confirmation flit in the reverse direction. When this confirmation arrives to the source generator, the connection is established and the source generator starts the traffic generation.

While the communication is being established, the monitor stores the delay values (R_DELAY CBR), the remaining PBR bandwidth (PBR REMAINING), and the jitter (JITTER) related to data streams in an external SRAM. This process is done following a defined sample rate (N_CYCLES_MT) until a total number of samples has been stored (N_CYCLE_TOT). These parameters are previously defined in the system configuration table (see Fig. 1). Monitored data are written into a report file to be studied later.

3.2 Generator

Each Traffic Generator is made of four modules and each one carries out a part of the communication process (Fig. 2).

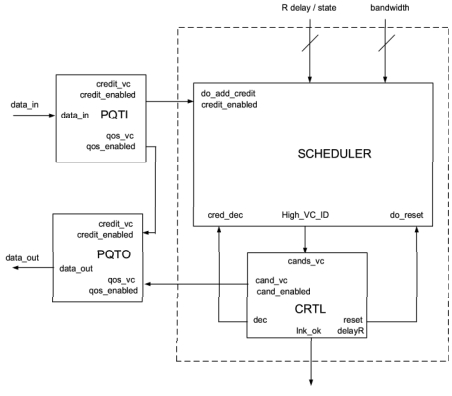

Figure 2: Architecture of the traffic Generator module.

The input module (PQTI) process the input data streams, identifying the flow control*phits* (credits), the synchronization *flits* and the data traffic (BE or QoS). If the input is a credit, the associated VC is identified and the scheduler credit manager is notified. If the input is QoS or BE traffic, the generator becomes a perfect data drain, the output module is notified (PQTO), and a credit is transmitted to the VC of the sender port in the SMMR reconfiguration cycle. When a synchronization flit is received, the generator accepts this flit without doing anything.

The output module (PQTO) sends two kinds of traffic: data *flits* and credits. This module receives the VC's identifier (VC_ID) that must inject traffic into the next *flit cycle*. If a data packet notification is received from the PQTI module, the PQTO module sends a credit during the reconfiguration cycle between *flits*. Otherwise, if the ID of a VC is received from the CTRL module, a QoS packet is generated and sent by the output channel.

The control module (CTRL) is the last step in the scheduling process. This module is devoted to the following tasks: it receives the VC's ID that should be sent in the next cycle and supplies it to the PQTO module to send the corresponding flit, it then indicates to the monitor that the scheduling process has finished and that it is time to have a sample. Finally it synchronizes all the other modules.

The scheduling module (SCHEDULER) is the most important functional component in our proposal. It must share the bandwidth link among all the VCs ready to transmit. In order to share the bandwidth required by every kind of traffic, time has been partitioned in rounds (where a round has a fixed and determinate number K of *flit cycles*). The number K of flit cycles per round determines the minimum bandwidth assigned to a data stream (1/K). In our experiments we set K=2048. On every *flit cycle* the scheduling of the VC that must be transmited in the next *flit cycle*, and the sending of the current data *flit*, are performed simultaneously.

The scheduler has a SUBSCHEDULER for each VC and a maximum bitonic network (MAX). The scheduling algorithm is divided in two steps. First, all the subschedulers work in parallel to generate a global *priority vector*, where every sub-scheduler contributes with its VC_ID and current priority. Second, the high priority VC is selected from the priority vector by using a MAX bitonic network module (see Fig. 3).

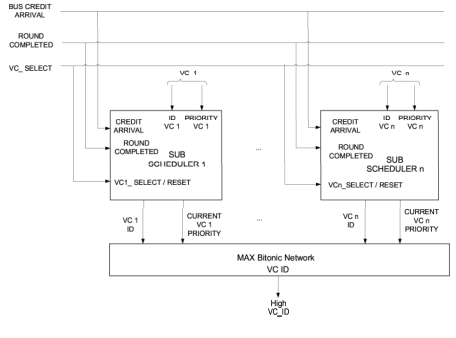

Figure 3: Architecture of the scheduler module with QoS guarantee.

Algorithm 1 Control mask transmitting conditions

if Credit_empty(CV_i)$\neq 0$ **then**
  if (CV_i.BW$_{CBR} > 0$ & CV_i.DL ≤ 0) **then**
    CV_i.CBR_EtT= 1
    CV_i.PBR_EtT= 0
  else
    CV_i.CBR_EtT= 0
    CV_i.PBR_EtT= ($CV_i.BW_{PBR} > 0$)
  endif
else
  CV_i.PBR_EtT= 0
  CV_i.CBR_EtT= 0
endif

Each subscheduler is composed of two submodules: a modified *SIABP* [3, 8] priority module and the CONTROL MASK module. The CONTROL MASK (Fig. 4) module verifies if the connection associated with a VC is able to transmit, following the conditions of **Algorithm** 1.

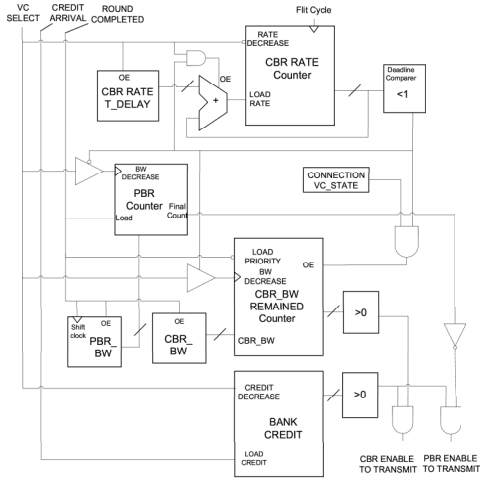

Figure 4: Architecture of the CONTROL MASK module

Thus, if a VC is selected to transmit a CBR_a *flit*, the *CBR_RATE* counter will be increased in a value equal to T_DELAY, the *CBR_BW REMAINED* counter will be de-

creased and the bank credit will be also decreased in 1. If this VC is selected to transmit a PBR *flit*, only the *PBR Counter* is decreased. In any other case, the VC has to wait for a later scheduling process and the *CBR_RATE* counter will be decreased.

The SIABP module (Fig. 5) stores the priority of the VC, which is directly related to its bandwidth reservation as the number of flits that can be sent per round, and guarantees the QoS by upgrading it (CURRENT VC_i PRIORITY). This priority is obtained as the ratio between the delay in the input queue and the interarrival time of flits (for more information see [1]).

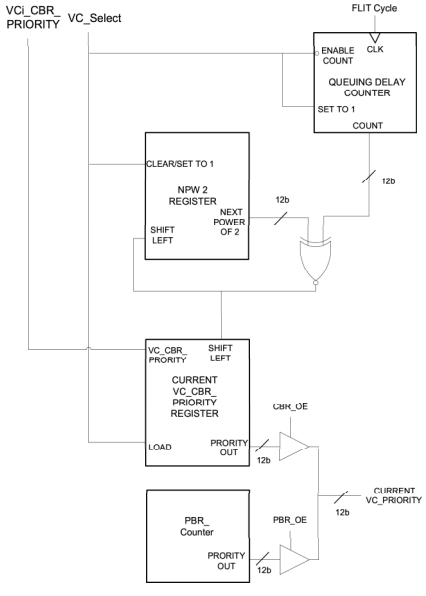

Figure 5: Architecture of the modified SIABP module.

The CONTROL MASK verifies that the VC_i is ready to transmit a CBR *flit* the next *flit cycle*. Then, the current priority of VC_i is where in the priority vector. If it is ready to transmit, a PBR *flit* is placed the PBR bandwidth remaining. In case where two kinds of traffic are able to transmit, the CBR traffic has priority over the PBR traffic. If it is not ready to transmit any kind of traffic, the priority associated with this connection will be zero. The priority vector is the input for the MAX bitonic network, giving as a result the ID of the VC with the highest priority (High VC_ID).

Each scheduling of the QoS scheduler has a temporal cost of $10 + log_2(N_{CV})$ clock cycles. Five cycles are from the local scheduling, and $log_2(N_{CV})$ are associated with the candidates' selection (MAX Bitonic Network). The remaining five cycles come from the actualization of the VC states and the synchronization with the output module.

4 Evaluation

In this section, the Generator/Monitor performance is reported. The proposed implementation has been developed on a Celoxica RC1000 card, which has a Xilinx Virtex 2000E (with 38,400 LUT and 640 Kb of Block-RAM), 8 MB SRAM external memory divided in four independent banks, and two PCI Mezzanine (PMC) cards for input/output data. The FPGA has been programmed with the hardware programming language Handel-C [9], using the Celoxica DK4 Design Suite development tool.

4.1 Area and delay results

In order to evaluate the implementation of the Generator/Monitor, we have scaled the design from 4 to 32 virtual channels per port. For the experiments, we have implemented a Generator/Monitor with two ports and a very simple SMMR router. The results presented in the Table 1 are the area size occupied by only one traffic generator, and the size of the SRAM external memory is not included.

Different configurations of the Generator/Monitor (4, 8, 16 and 32 virtual channels per link) show a lineal growing in the area size (in LUTs) occupied by the implementation when the number of VC per port is increased. Meanwhile, a logarithmic reduction in the maximum system operation clock rate, which tends to be stable around 20 Mhz, is

Table 1: Area and clock rate for different configurations of the QoS Generator

N. Streams	LUTs	Clk Rate
4	1.889	41 MHz
8	3.681	36 MHz
16	6.582	30 MHz
32	13.968	21 MHz

observed.

If these results are compared with the previous work [7], in which there is no generation of VBR traffic, the area size has increased by 80 LUTs per VC. This is due to the higher compexity of the new traffic generator. On the other hand, the maximum operation clock rate tends to be stable, around the same value, but has less clock rate for the configurations with a small number of VCs.

4.2 Traffic analysis

In this subsection, the behavior of the Generator/Monitor is evaluated. We center our study on the bandwidth link use of the Generator/Monitor and its distribution among the generated data streams. For the test we have configured the FPGA as one Generator/Monitor with two ports or links, and a very simple (ideal) SMMR router. One generator performs as a traffic source, meanwhile the another preforms as a data drain. The ports have been configured with 4 virtual channels, and we have made two different experiments.

In the first experiment, we want to show our proposal for a VBR traffic generator. For this purpose, we use only a data stream transmitted by a single VC. As explained in section 2, we distributed the VBR bandwidth reserved in CBR_a and PBR traffic. In this experiment, the CBR_a traffic generated has a 25% of link BW reserved and the PBR traffic has a 6.25%. The Fig. 6 shows that the CBR_a is transmitted as a CBR rate, while the PBR traffic is sent using a BB (*Back-to-back*) algorithm [1], as soon as possible, until its complete transmision.

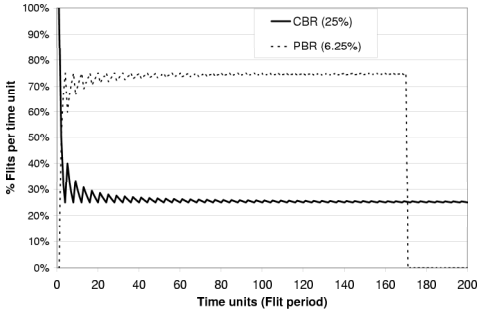

Figure 6: Bandwidth distribution for only one VC supplying VBR traffic.

In the second experiment, we have evaluated the Generator/Monitor when different kinds of traffic are circulating in the network. We have configured the traffic generator to transmit CBR traffic by two virtual channels (VCs 0 and 3) and VBR traffic by the other two VCs (1 and 2). We have not reserved the 100% of the bandwidth and have removed the BE traffic to observe better the evolution of VBR traffic.

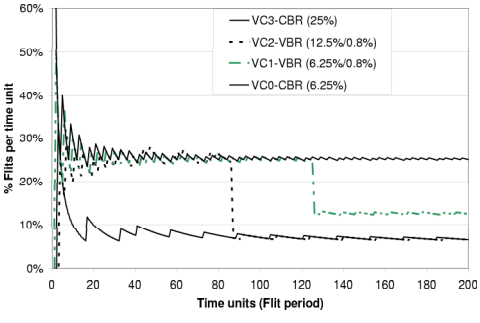

Figure 7: Bandwidth distribution for 4 virtual channels with mixed traffic, two CBR streams and two VBR streams.

The Fig. 7 shows the bandwidth distribution. The VCs 1 and 2 use more bandwidth at the beginning because first they have CBR_a and PBR traffic, and when the PBR traffic is ended they have only the reserve for CBR_a. The throughput at the begining is almost a

100%, but when the PBR traffic is finished, the bandwitdh tends to be stable in the percentage equivalent to the equivalent CBR reserved bandwidth. In this case, it is the 50 % of the total bandwidth link.

5 Conclusion and further works

The most important thing in our proposal is the design and implementation of an integrated and configurable tool that can help in the future development and design of high performance QoS traffic servers and routers through the use of more realistic evaluations.

As both traffic generators and routers are in the same platform, we can improve its development in a global way, reaching higher optimization levels. This proposal is developed in a reconfigurable architecture (FPGA) and using a high level programming language (Handel-C). These allow us to prototype the system, to make it very parameterizable and scalable.

The new Generator/Monitor for the SMMR router generates combined QoS (CBR and VBR) and BE traffic for an evaluation of routers closer to reality. Currently we are working to optimize the design in order to obtain less clock rate decrease when the VCs per link are increased.

References

[1] M. Caminero. *Diseño de un encaminador orientado a trafico multimedia en entornos LAN (in Spanish).* Ph.D. Tesis. Universidad de Castilla-La Mancha, 2002.

[2] M. Canseco, J. Claver, G. Leon, and I. Vilata. Prototipado de un MMR simple en una FPGA (in Spanish). In *XV Jornadas de paralelismo - Computación de altas prestaciones*, 2004.

[3] J. Claver, M. Carrion, M. Canseco, M. Caminero, and F. Quiles. A new hardware efficient link scheduling algorithm to guarantee QoS on clusters. In *Lecture Notes in Computer Science, Euro-Par, Parallel Processing: 11th International.* Ed. Springer-Verlag, 2005.

[4] P. P. Chul and B. Frantz. A reprogrammable FPGA-based ATM traffic generator. In *Proceedings of the 6th Great Lakes Symposium on VLSI*, pages 35–38, 1996.

[5] L. Benini and G. De Micheli. Networks on chip: A new SOC paradigm. *IEEE Computer*, 2002.

[6] N. Genko, D. Atienza, and G. De Micheli. NoC emulation on FPGA: HW/SW synergy NoC features exploration. *Proceedings of Parallel Computing, Parco*, 2005.

[7] J. M. Claver, P. Agusti, G. Leon, M. Canseco and G. Fabregat. A reprogrammable and scalable QoS traffic Generator/Monitor on FPGA. *IEEE International Conference on Communication Technology, Guillin (China)* 2006.

[8] M. Caminero, C. Carrion, F. Quiles, J. Duato, and S. Yamalanchili. A cost-effective hardware link scheduling algorithm for the mul-timedia router (MMR). In *Lecture Notes in Computer Science, Networking (ICN'01)*, 2001.

[9] S. Chapell and C. Sullivan. Handel-C for co-processing an co-design of field programmable systems on chip. In *Proceedings of the JCRA'02*, 2002.