A Scalable Multiprocessor for Real-time Signal Processing

Daniel Scherrer, Hans Eberle
Institute for Computer Systems, Swiss Federal Institute of Technology
CH-8092 Zurich, Switzerland
{scherrer, eberle}@inf.ethz.ch

Introduction

Multimedia applications that operate on data such as audio, video or graphical data are showing a growing need for processor power. To meet the required performance, often specialized processors and custom hardware architectures are used. Our goal was to design a scalable multiprocessor that is versatile enough to replace many of these custom solutions for applications that perform signal processing on multimedia data in real time.

To achieve high overall performance, a scalable multiprocessor system requires an interconnection structure that offers high communication bandwidth. Further, to process data in real time, a suitable interconnection structure has to provide real-time or quality of service (QoS) guarantees when transmitting data. These requirements led us to the development of the Switcherland system [1, 2]. Switcherland uses a scalable interconnection structure based on switches and forms the basis of the work described here.

The Switcherland Interconnection Structure

Traditional computer architectures use internal and external interconnection structures based on busses. Examples are the Ethernet network and PCI bus. However, busses have become a communication bottleneck. For this reason more modern approaches such as the ATM networking technology replace busses by switches. This offers the advantage that communication bandwidth can be increased as switches are added and the size of the structure is enlarged. Switcherland is a switch-based interconnection structure for workstation clusters that is even more ambitious than modern networking technologies in that it replaces both the internal and external connections of computers by a single fast network. The network consists of switches and nodes, that is, I/O and processor nodes connected by Fibre Channel links.

Figure 1 shows an example of a Switcherland system for audio processing. It consists of a control unit with a processor node equipped with disk drives and a frame buffer that hosts the user interface, and an audio unit with an audio I/O node and two signal processor nodes.

Switcherland is able to provide QoS guarantees in respect to bandwidth, latency and latency jitter in that it supports two types of traffic classes: variable bit rate (VBR) traffic and constant bit rate (CBR) traffic. The two traffic classes differ in the guarantees provided by the switches and nodes. For VBR traffic, buffer space is reserved along the transmission path to guarantee that data never has to be dropped due to overflowing buffers. For CBR traffic, bandwidth is guaranteed and latency is bound. These properties make Switcherland an ideal platform for embedded applications that have to process multiple continuous data streams.

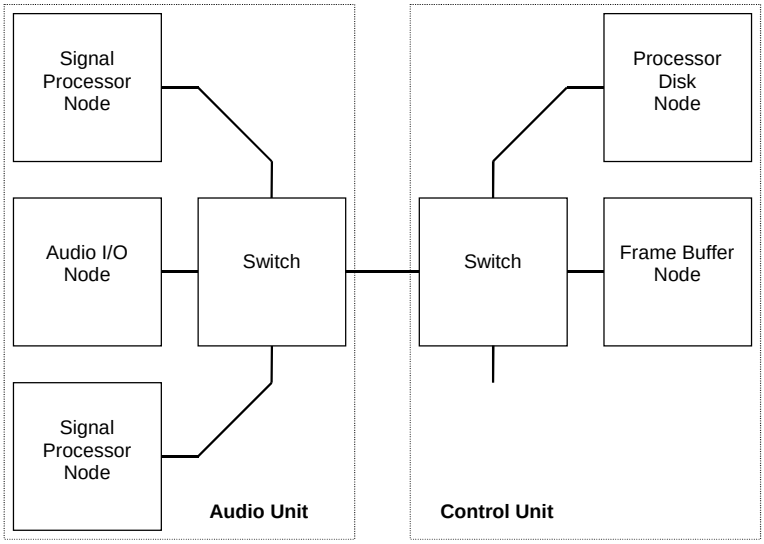

Figure 1: An example of a Switcherland system.

The Need for a Real-time Signal Processor

Though general-purpose processor nodes for Switcherland are available, they do not provide an adequate solution for most signal processing applications. More specifically, a node optimized for signal processing can be characterized as follows:

- Signal processing requires special optimized operations, for example, a fast multiply/accumulate operation and, therefore, a specialized instruction set. Also multiplication has to be performed in hardware.
- Most signal processing applications need only a small amount of memory. Therefore, support for virtual memory and a sophisticated memory management unit are not needed.
- A data cache can be omitted on the CPU chip in favor of an internal memory, because the working set is usually small and static.
- Signal processing tasks usually consist of several sequential subtasks. Ideally this property is reflected by a hardware architecture optimized for data flow processing.

Based on these requirements we built a dedicated signal processor that is capable of processing data in real time. In terms of throughput the node is fast enough to consume and produce data at the full Switcherland link bandwidth and to pass it on adding only a small and easily predictable amount of latency and latency jitter.

The Board Architecture

Processor Evaluation

When evaluating a suitable processor, we found that there are two groups of processors well suited for signal processing in real time. One group consists of digital

signal processors (DSP), which are characterized by having large internal memories, usually no caches and a very specialized instruction set. To illustrate the latter, most DSPs are able to compute one filter tap of a FIR filter (including addition, multiplication, counter and address incrementing and a loop branch) by executing a single instruction. Though several projects have used DSPs for building multiprocessors [3], the major drawback of DSPs is that they are difficult to program since the instruction set can only be used efficiently by programming them in assembler.

In addition to high performance, a simple general purpose programming model was another evaluation criteria since we intend to provide a flexible object-based environment for developing programs. Therefore, we have chosen a processor of the second group: a RISC microcontroller with an instruction set extended by typical DSP instructions. We have chosen the Hyperstone E1-32 [4] because of its simple structure and its glueless bus interface, which makes it easy to connect different types of memory (including FIFO memories that we use for communication).

Another interesting aspect of these RISC/DSP microcontrollers is their low cost. We found that for applications that can be parallelized easily it is much more cost-effective to use several low-cost RISC/DSP microcontrollers than a single sophisticated DSP.

Board Design

Figure 2 shows a block diagram of the board. The main components of the board are eight processor elements and an FPGA, which acts as the communication controller. Each processor element has its own 4 Mbyte of local memory (DRAM) and two FIFO memories used for buffering data when communicating with other processors and other Switcherland nodes. No additional device or glue logic is needed because of the simple bus interface of the E1-32 processor chip. We decided to put eight processors onto the board since we estimated that with the given bandwidth of a Switcherland link up to eight E1-32 processors can be kept busy with simple computations.

Local Interconnect

As Figure 2 shows, the eight processors are divided into two banks with each having separate busses to connect the FIFO memories to the FPGA. Together with the FIFO memories, the FPGA implements an input-buffered crossbar switch. It has three ports connected to the two processor banks and the Switcherland link. The data rate on all ports corresponds to the link bandwidth of 26.6 Mbyte/s. The communication controller also has the capability to split an incoming data stream and distribute it to several processor elements. This feature is useful when a single processor is not fast enough to process data at the full link data rate. For example, this mechanism to split data can be used to split video images into smaller parts, each of which is handled by a single processor. As the FPGA is reprogrammable, more functionality can be added as required by the application making it possible to implement further preprocessing steps.

Care has to be taken when splitting a task and distributing the resulting subtasks to several processors. Although any kind of distribution is possible, the interconnect can be used most advantageously if the resulting subtasks of a vertical split are assigned to the processors within a bank and if the resulting subtasks of a horizontal split are executed by different banks. This way, it is possible to make use of the full bandwidth

of each bank. We have chosen this kind of communication architecture in contrast to a more sophisticated output-buffered switch that needs to provide significantly more bandwidth, because we wanted to keep the cost low not only of the processor elements but also of the communication controller. With this organization, one FPGA with a relatively low pin count is enough to implement the data paths and control logic needed.

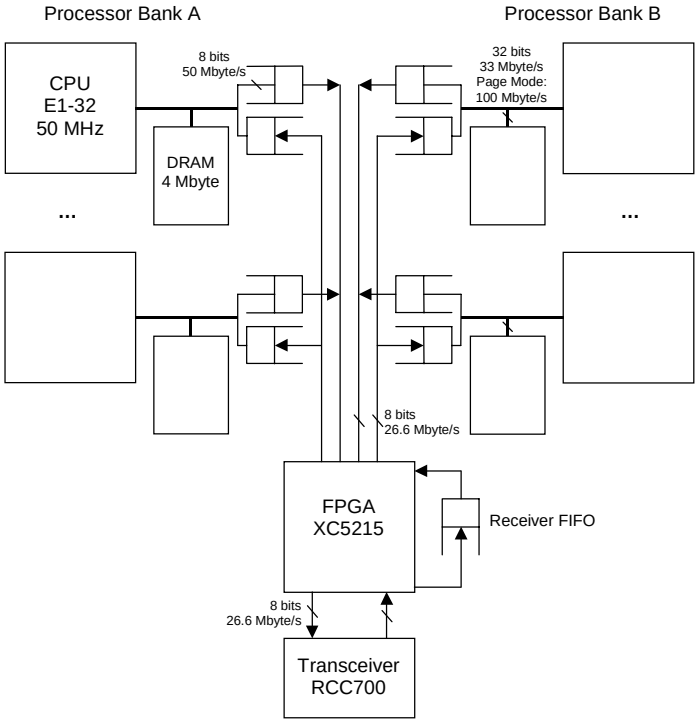

Figure 2: Board block diagram.

Table 1 summarizes the characteristics of the board. We currently use E1-32T processor chips that are available in two speed grades allowing clock rates of 50 MHz and 66 MHz, respectively. Each processor contains a 4-kbyte internal RAM and a 128-byte instruction cache. Aggregate peak performance of all processors is 400 MIPS and 530 MIPS, respectively. The table also mentions the recently introduced E1-32XT. With this processor version the board has a peak performance of 800 MIPS.

Processors:	8 x Hyperstone E1-32(X)T, 50/66/(100) MHz, 4/(8) kbyte internal RAM, 128 byte instruction cache
Memory:	8 x 4 Mbyte DRAM
Peak performance:	400/530/(800) MIPS
On-board switch bandwidth:	3 x 26.6 Mbyte/s

Table 1: Board characteristics.

The Programming Model

As the hardware structure implies, we envision a message-passing programming model. We assume that data is passed from one processor to another one without the need to wait for an answer or a result of a computation. This assumption is justified by the observation that communication in data flow applications happens in only one direction.

We suggest an object-oriented data flow programming-model, in which data is passed on between objects as parameters of method calls. The model allows objects to be located on several processors whereby it makes no difference for the programmer if an object calls the method of another object located on the same or a different processor element. We are currently building a kernel, which implements this programming model. The system is event-driven and there is only one thread of control per processor. This seems to be the appropriate solution for processing continuous data, since data that is produced at a constant rate and that tolerates little latency should be processed in steps that are executed either in a constant amount of time or within certain time bounds. It seems that such stringent requirements are best met without allowing for task preemption.

Applications

We foresee a wide range of applications to be implemented with the hardware and software described here. As a proof of concept we will first implement a sound studio. Mixer elements and effects can be assigned freely to the processor elements of one board or even to the processor elements of several boards interconnected by the Switcherland network. If more sound channels and/or processing steps are added and, eventually, computing power needs to be increased, one can simply add further nodes to the network.

Other possible applications are the compression and decompression of audio and video data, image and video processing, as well as animated graphics and virtual reality. Yet, we have to verify our programming model with the help of these applications.

Status and Conclusions

The hardware of the multiprocessor board has been implemented and tested. Fully functional prototype boards are available and are currently being used to develop the mentioned kernel. Simple programs have been implemented to verify that the throughputs of the processors as well as of the local communication infrastructure match the one of the Switcherland link.

By tailoring the architecture of the board to the data flow model of the targeted signal processing applications the design of the hardware was simplified drastically. In addition, the simple and flexible bus interface of the Hyperstone E1-32 processor also helped to reduce the complexity of the board design. The result is a multiprocessor that offers high performance for signal processing applications at low cost.

References

[1] E. Oertli, H. Eberle, *Switcherland – An Interconnect for Workstations*, Proceedings of the 21st Euromicro 95 Conference, Como, September 4-7, 1995, pp. 369-375.

[2] H. Eberle, *Switcherland – A Scalable Interconnection Structure for distributed Computing*, 3rd International Conference of the Austrian Center for Parallel Computing, Klagenfurt, September 22-25, 1996. In L. Böszörményi, Editor, Parallel Computation, Lecture Notes in Computer Science, vol. 1127, Springer-Verlag, 1996, pp. 36-49.

[3] A. Gunzinger, U. A. Müller, W. Scott, B. Bäumle, P. Kohler, W. Guggenbühl, *Architecture and Realization of a Multi Signalprocessor System*, International Conference on Application Specific Array Processors. IEEE Computer Society Press, 1992.

[4] *Hyperstone E1-32/E1-16, 32-Bit-RISC/DSP Microprocessor User's Manual*, hyperstone electronics GmbH, Am Seerhein 8, D-78467 Konstanz, Germany.