Hardware Reconfigurable Neural Networks

Jean-Luc Beuchat, Jacques-Olivier Haenni and Eduardo Sanchez

Swiss Federal Institute of Technology, Logic Systems Laboratory,
EPFL – LSL, IN – Ecublens, CH – 1015 Lausanne

Abstract. This paper presents the concept of reconfigurable systems using the example of a digital hardware implementation of neural networks, as well as RENCO, a platform very well-suited for the prototyping of such systems. RENCO is a network computer containing a reconfigurable part composed of four Flex10K FPGAs from Altera (10K130 or 10K250).

1 Introduction

General-purpose processors perform various tasks, such as scientific computation or image processing. However, the performance of dedicated circuits (ASICs) are much better. Unfortunately, these circuits only perform a very specific task and their development cost is important. The availability of reconfigurable circuits (like FPGAs) opens the way to new interesting architectures.

This paper presents the concept of reconfigurable systems based on FPGAs. It then briefly describes RENCO, a reconfigurable network computer, which is a platform for the prototyping of logic designs or reconfigurable systems, before explaining our current research in hardware reconfigurable neural networks.

2 Reconfigurability using FPGAs

An integrated circuit is said to be reconfigurable when the user can change its functionality by programing. There are many programing mechanisms (fuses, memory cells, etc.), but none requires the circuit to be sent to the manufacturer unlike ASICs.

An FPGA [1, 2] is composed of a matrix of logic cells placed over a network of interconnection lines. The user can program the function of each cell as well as the interconnections between cells and the I/O cells, which are also programmable.

With a structure similar to the MPLDs (Mask Programmable Logic Device), the FPGAs have the advantage to be on-site reconfigurable, allowing the use of a circuit only a few minutes after the design, versus many months for the MPLDs. The most recent FPGAs can be reconfigured in a millisecond. In ten years' time we will certainly dispose of circuits which will be programmable in less than one hundred microseconds [3]. Some FPGA families (such as the 6200 from Xilinx)

allow a partial reconfiguration of the circuit. This mechanism notably reduces the programing time.

The integration of FPGAs in traditional computers allows more optimal implementations of applications. It is possible to design a platform with a certain amount of programmable logic available to the user. This characteristic is the basis for reconfigurable systems. The user can, for example, design a specific coprocessor for each of his/her applications. It is also possible to split an algorithm into different steps and to design a different FPGA configuration for each step. The FPGAs are then reprogrammed during the application execution.

The reconfigurable systems are particularly efficient for the implementation of algorithms working at the bit-level, such as pattern matching or integer arithmetic. However, FPGAs are not the optimal solution for all hardware realizations; they are not suited for floating point computation for example [3].

3 RENCO, a REconfigurable Network COmputer

In order to implement our reconfigurable systems, we have designed a computer, RENCO (REconfigurable Network COmputer), combining both the power and ease of use of the current FPGAs and the notion of application decentralization characteristic of network computers [4].

RENCO does not only allow to download an application from the network, but also the hardware architecture which optimally executes it. RENCO is composed of a standard commercial general-purpose processor (a Motorola MC68EN360 [5]) and a cluster of FPGAs (Altera Flex 10K130 [6]) connected to their own memories and to the processor bus. The 68360 can use these FPGAs and their memories as a specialised coprocessor whose functionality (configuration) can be dynamically downloaded through the network as and when required.

The 68360 runs the real-time operating system RTEMS [7] (Real Time Executive for Multiprocessor Systems) and a Java virtual machine, Kaffe. Therefore, RENCO can be seen as a Java network computer.

The current version of RENCO contains about 520,000 logic gates in its reconfigurable part, but the use of the latest chips from Altera will allow us to have one million logic gates, which is enough for the implementation of high-complexity dedicated processors.

3.1 General Architecture

Figure 1 shows the general architecture of RENCO which is composed of a processor part (built using the Motorola MC68EN360) and a reconfigurable part (four Flex 10K130 FPGAs).

The processor bus is connected to the four FPGAs. They can therefore be accessed as peripherals by the processor. Each FPGA has its own memories (DRAM and SRAM).

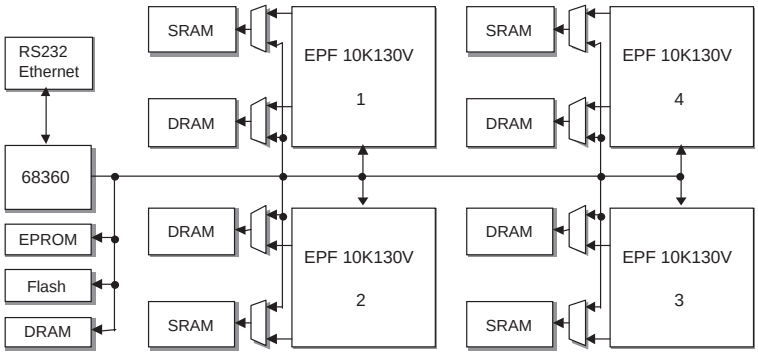

Fig. 1. RENCO block diagram

3.2 The Processor Part

The MC68EN360 has been chosen for its communication capabilities (Ethernet, RS232), for its integrated memory controller and for the availability of many free software tools (development systems, operating systems, etc.).

The 68360 is directly connected to an EPROM storing a minimal operating system, as well as a DRAM and a Flash memory. The processor can also access the FPGA memories. A programmable circuit transparently arbitrates the simultaneous accesses from the processor and an FPGA.

RENCO is connected to the network through an Ethernet 10Base-T interface. This communication interface is used at boot time for downloading the operating system and then the applications and hardware configurations.

3.3 The Reconfigurable Part

The reconfigurable part is composed of four Flex 10K130 (or 10K250) from Altera. They can be used independently or together; this allows the implementation of very large designs distributed over many FPGAs.

Each FPGA is connected to a SRAM and a DRAM, as well as to the other FPGAs, as indicated on Fig. 2.

Each FPGA receives two clock signals: the 25-MHz system clock and a programmable clock signal whose frequency is between 320 kHz and 100 MHz. Each FPGA can therefore run at a different frequency.

Three pairs of 120-pin connectors allow the board capabilities to be expanded. They contain FPGA I/O signals and the processor bus.

3.4 Software

On grounds of versatility, power of use and user convenience, we have decided to implement a whole operating system, and have chosen RTEMS. It is a pre-

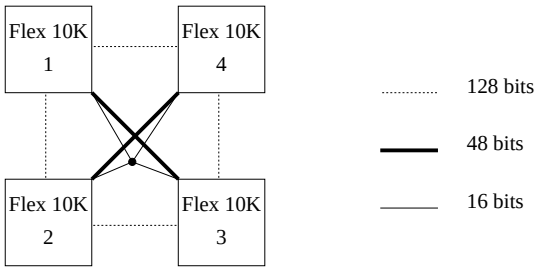

Fig. 2. FPGA interconnections

emptive multitasking operating system with quite modest memory requirements, available for the 68360 processor. RTEMS also includes a TCP/IP stack.

We have then ported Kaffe [8], a Java virtual machine, because, among others, Java is simple to use, has a standardised API and understands threads at the language level. A board-specific API has been added. It allows, for example, to configure the FPGAs or to set the programmable clock frequencies.

The Ethernet interface and the TCP/IP stack are not only used to transfer FPGA configurations, it is also possible to access RENCO through the http protocol. It is therefore possible to get the board state or to control it from a Web browser such as Netscape.

4 Reconfigurable Neural Networks

Among the reconfigurable systems currently designed in our laboratory, we present here a hardware implementation of multi-layer perceptrons. These networks are well-suited for classification problems like hand-written character recognition. Let us briefly consider this problem. The network is trained with a set of characters written by different people. After pre-processing steps (normalisation, smoothing,...) we obtain a grey-level image (M x N pixels) of each character. The M x N grey-level values are stored in an input vector to which is assigned a class attribute.

Figure 3 depicts the learning system. An input vector is presented to the neural network which determines an output. The comparison between the computed and desired output (class attribute) provides an output error. This signal is used by a learning algorithm to adapt the network parameters.

We now briefly describe the architecture of a multi-layer perceptron. It is composed of several layers of interconnected processing elements (PEs) or neurons:

- an input layer which only holds input values,
- one or more hidden layers of PEs and
- an output layer of PEs, from where the response comes.

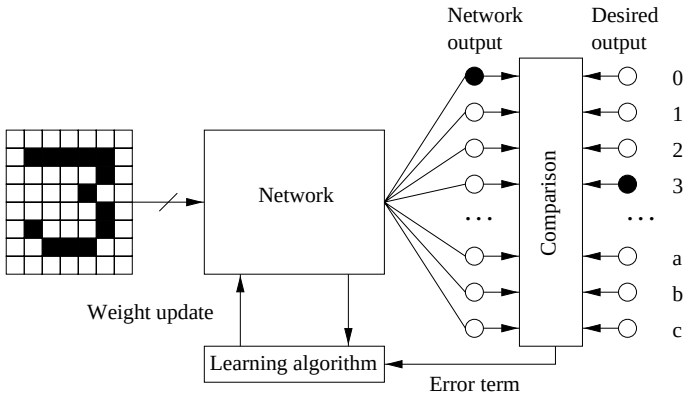

Fig. 3. Learning principles

Notice that connections are only possible between two adjacent layers. A network is fully connected if all possible connections exist.

Each PE computes a weighted sum of its inputs. The activation of a neuron is obtained by applying a non-linear function to this sum. The weights at each connection are modified by the learning algorithm during the training.

4.1 The Backpropagation Algorithm

The Backpropagation algorithm is widely used for training multi-layer perceptrons [9]. It iteratively computes the values of weights using a gradient descent algorithm. The Backpropagation algorithm consists of the following stages:

1. Network initialization.
   All weights are initialized to small random numbers.
2. Forward propagation.
   An input vector is presented and propagated layerwise through the network.
3. Output error computation.
4. Backward propagation.
   The output error signal is back-propagated through the network. This process allows to assign errors to hidden neurons.
5. Weight update.
   Previously computed errors (stages 3 and 4) and neuron activations determine the weight changes.

Steps 2 to 5 are carried out for all vectors in the database. This training process is repeated until the output error signal falls below a predetermined threshold.

When we train a system by example, it is usually impossible to provide every possible input signal. Therefore, an important issue of training is the capability of the network to generalize to previously unseen patterns. However, the

generalization capability depends on the network topology. A rule of thumb for obtaining a good generalization is to use the smallest system that can learn the training vectors.

Unfortunately, the Backpropagation algorithm does not give any information about the topology of the network (number of hidden layers, interconnections, number of neurons). Pruning (or growing) algorithms allow to find an optimal topology during training by removing (or adding) neurons and connections.

4.2 Hardware Implementation

As the training of a large neural network requires an important amount of computation time, the design of specialised circuits is interesting. We are currently developing an FPGA-based neuroprocessor. Among the difficulties encountered during this work, let us mention:

1. The choice of the topology.
   If we implement a fully interconnected network on an FPGA, we obtain a really fast system. However, this method has some drawbacks:
   - Waste of hardware resources.
     The Backpropagation algorithm requires neuron activations during the weight update stage. Therefore, pipelining cannot occur during training and hardware dedicated to each layer is partially used.
   - Scalability.
     Such an architecture requires a multiplier for each connection. Furthermore, the number of interconnections between neurons increases as the network grows. A network designed for character recognition involves thousands of connections. So it would be impossible to implement it with RENCO.

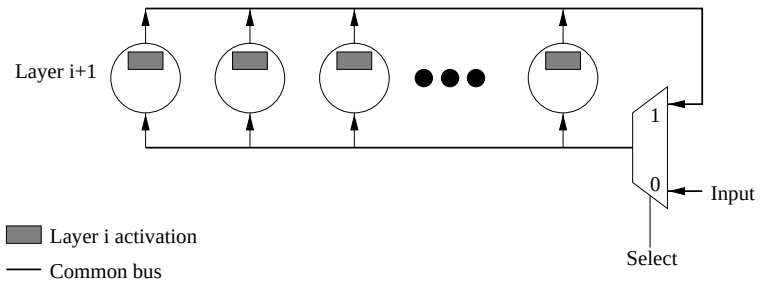

Fig. 4. General architecture

A time-multiplexed interconnection scheme provides a good trade-off between speed and scalability. Figure 4 shows the general architecture of the

system. The main idea is to connect all outputs of layer i and inputs of layer i+1 to a common bus. The network is implemented by a single layer of PEs. Weights are stored in the memory associated with the FPGAs. This architecture offers a good scalability. The number of hidden neurons (for a given layer) is bound by the number of programmable logic cells of RENCO. The amount of available memory limits the number of connections.

2. Number representation [10].
   As 16- or 32-bit floating-point multipliers require an important amount of space on FPGAs, other number representations have been developed. Among them, let us mention
   - Restriction to negative power-of-two (or sums of negative power-of-two).
   - Pulse stream encoding.
   - Fixed-point representation which is used in our work.

4.3 Dividing the Backpropagation Algorithm

Each step of the Backpropagation requires a different hardware configuration. For example, the random number generator is implemented using a Linear Feedback Shift Register (LFSR). Once initialization has been performed, the LFSR remains unused. The main idea of our system is to divide the backpropagation in several sequentially executed stages (such an approach has been successfully used by J. Eldredge [11]):

1. Network initialization.
2. Forward propagation.
3. Error computation.
4. Backward propagation.
5. Weight update.
6. Pruning.

A peculiar FPGA configuration is associated with each stage and RENCO is reconfigured during the network training. Notice that the reconfiguration time is a crucial point. If this process needs more time than computation, our approach is not appropriate. The choice of the FPGA family is really important in order to realize an efficient system.

5 Conclusion

The reconfigurable computing allows the on-site implementation of efficient complex systems. However, the current FPGAs are not suited for all kinds of problems. It is therefore interesting to split a problem into a software part (executed on a commercial processor) and a hardware part (executed on a dedicated reconfigurable coprocessor). Currently, the decision on where to put the boundary between software and hardware depends on the designer's experience. A global design theory, called codesign, is still to be developed.

Moreover, considering the constant technological progress, it would not be surprising to see in a few years' time the application of this approach inside commercial processors. It would then be possible to have at hand a programmable space inside the processor, in the same way that cache memories have moved from outside to inside the processors [3].

While waiting for the availability of such circuits on the market, we use RENCO which is very well suited for the prototyping of reconfigurable systems and codesign because of the amount of memory and programmable logic it contains. Furthermore, the concept of network computer brings in many advantages. The main resources of the computer (mass memories, applications, software libraries, etc.) are exclusively accessible through the network, which reduces the maintenance cost while adding flexibility to the system.

References

1. Z. Salcic, A. Smailagic. *Digital System Design and Prototyping Using Field Programmable Logic*. Kluwer Academic Publishers, Boston, 1997.
2. E. Sanchez. Field Programmable Gate Array (FPGA) Circuits. In E. Sanchez, M. Tomassini (eds.). *Towards Evolvable Hardware*. Springer-Verlag, Berlin, 1997, pp. 1-18.
3. J. Villasenor, W. H. Mangione-Smith. Configurable Computing. *Scientific American*, June 1997, pp. 54-59.
4. M. Slater. The Many Faces of Network Computers. *Microprocessor Report*, Vol. 10, num. 16, 1996, p. 3.
5. Motorola. *MC68360, Quad Integrated Communications Controller*, User's Manual. 1993.
6. Altera Corporation. *EPF10K130, Embedded Programmable Logic Device*. San Jose, April 1997.
7. RTEMS Home Page, http://lancelot.gcs.redstone.army.mil/rtems.html.
8. Kaffe, A free virtual machine to run Java(tm) code, http://www.kaffe.org/.
9. M. A. Lehr, B. Widrow. 30 Years of Adaptative Neural Networks : Perceptron, Madaline and Backpropagation. *Proc. IEEE*, 78(9): 1425-1442, September 1990.
10. S. Sakaue, T. Kohda, H. Yamamoto, S. Maruno, Y. Shimeki. Reduction of Required Bits for Backpropagation Applied to Pattern Recognition. *IEEE Transactions on Neural Networks*, Vol. 4, no. 2, March 1993, pp. 270-275.
11. J.G. Eldredge and B.L. Hutchings. Density Enhancement of a Neural Network Using FPGAs and Run-Time Reconfiguration. In Duncan A. Buell and Kenneth L. Pocek, editors, *Proceedings of the IEEE Workshop on FPGAs for Custom Computing Machines*, pages 180-188, Los Alamitos, California, April 1994. IEEE Computer Society, IEEE Computer Society Press.