Software Radio Reconfigurable Hardware System (SHaRe)

Xavier Revés, Antoni Gelonch, Ferran Casadevall, and José L. García

Universitat Politècnica de Catalunya, Signal Theory and Communication Department
Jordi Girona 1-3, 08034 Barcelona (Spain)
{xreves, antoni, ferran}@xaloc.upc.es
garciam@teleline.es

Abstract. Recent requirements and evolution of personal communications systems will tend to increase the number of applications that will run over the same hardware/software. While an option is providing this platform with all the algorithms needed, a more suitable one is providing such a platform with the capacity to evolve, along time, from one function to another. Here we present a hardware platform with self reconfiguration abilities depending on system demand. The reconfiguration can be partial or complete within a short time to cope with the current application. This capability has an important effect on the software radio techniques applied to terminals and base stations as it will add extra value through a quick support to new standards and the incorporation of new software-designed applications.

1 Introduction

When working in mobile systems, the present third generation (3G) deploying has produced a substantial growth of services offered by network providers. Multiplicity of bands, data formats and rates, and processing demands in general will be common attributes. These diversity requires a quite complex structure with separate silicon for each possible standard. Another solution is using an application-multiplexed structure able to bear all this dispersion. It is clear that an aspect that can reduce implementation costs is reusing as many times as possible the same system in several applications. In this case, the second approach is quite reasonable when designing terminals from the point of view of the Software Radio techniques [1], [2], [3]. Those terminals must be capable of managing new and different standards and/or applications but with only part of them being executed concurrently. It's reasonable considering that different applications share part of the structure so it seems interesting to change only part of the processing structure. Also, software radios can reduce manufacturing costs and allow upgrading of the system as soon as different processing techniques and applications appear.

One important aspect in reconfigurable systems is the cost in terms of time, that is, the time spent on reconfiguring the system. In mobile communications area, reconfiguration times are not specially stressing when a user changes from one data stream format to another since a certain delay will be allowed. Perhaps a harder requirement should take this delay into account, like in the case where a mobile user

R.W. Hartenstein and H. Grünbacher (Eds.): FPL 2000, LNCS 1896, pp. 332–341, 2000.

changes from one standard protocol to another trying to maintain the current call. Moreover, we must not forget that an important design issue of this kind of systems is the scalability, that is, the capability to provide increasing processing power without increasing complexity.

By other hand, it is well known that Field Programmable Gate Arrays (FPGA) offer the possibility to designer of reshaping the application as many times as wished. Taking advantage of that, it is possible building hardware systems that change their functionality at any time. Also several options appear when focusing on what parts of the system can be modified or how you are actually doing it. Totally or partially reconfigurable FPGAs are suitable if the overall system flexibility is kept to allow, at least, working within the selected area. This can be accomplished using many different architectures where inherent FPGA flexibility can polish errors in the previous design stages. But choosing an adequate initial architecture can help reaching final objectives. One important aspect when designing the system architecture is the minimum reconfigurable block size. Each of these blocks must be small enough to avoid reconfiguration overhead when only a small part of the system is to be modified. Conversely, large blocks have the ability to simplify the architecture.

With this in mind, and considering the structures usually implemented in a typical radio link, a Reconfigurable Hardware System (SHaRe) was designed and built. This partially auto-reconfigurable platform is being presented in more detail further on.

The final objective of this research work was to develop a reconfigurable hardware platform, mixing FPGAs and DSP technology, able to test, in real time, the proposed third generation mobile system radio access technologies and evaluate its performance. This platform should be as flexible as possible to allow the characterisation by software of the different radio access techniques proposed. From the viewpoint of the Software Radio implementation requirements, the aggregate processing demand of a generic mobile cellular base station managing 30 users, without considering the IF processing demand, can be estimated around 150 MOPS (Million Operations per Second). This value include only the baseband, bitstream, source and the signaling processing demand. About the IF processing demand it is estimated around 2500 MOPS which is assumed, until now, carried out by special-purpose digital receiver chips.

2 SHaRe Platform Architecture

Considering all the previously commented ideas, our aim was building a system based on a set of blocks made of non-partially reconfigurable FPGA [4] of an undetermined size. Each of these blocks will include one or more clusters which will be the smallest reconfigurable portions of the system. Defining the blocks dimensions can be done on the basis of the knowledge of the set of possible applications (a priori knowledge) and associating every block with one ore more clusters. If those applications are not known the definition can be done on the basis of certain abstract criteria. The second approach requires to define a flexible platform that allowing the union of several clusters to accommodate a single block. Again the size of these clusters must be a trade-off between the available number of them and the range of valid sizes for each one. It cannot go unnoticed that a highly clustered system

increases designer effort as the underlying hardware must be known and the desing must be partitioned. Conversely, a not much clustered system, although simplifies architecture and designer effort (must deal with fewer and larger FPGAs) can drift towards high reconfiguration cycles or towards scarcely used clusters what represents a lost of resources.

The final clustered architecture can be supported by an automation software at a high level language with a compiler having perfect knowledge of the underlying structure. This approach is out of the scope of this document, but the actual structure of the system being presented has grossly been thought as a tridimensional, simple but flexible, data, control and configuration flow so that the "only" task of the compiler would be to identify those data flows present into the design.

2.1 Defined Blocks

The proposed flat architecture for the FPGA network board (SHaRe) is shown in the Fig. 1. Three different modules are shown: the Peripheral and Processing Management (PPM) module, the VME [5] Interface and Programming (VIP) module and the Intensive Processing Unit (IPU) module. All them are constituted of several FPGAs as "intelligent" part. The modules are housed over a printed circuit board with a VME physical interface in 6U format which allow the scalability of the system. Notice the possibility of SHaRe to be connected to a typical VME backplane and to be concatenated with another VME compliant board.

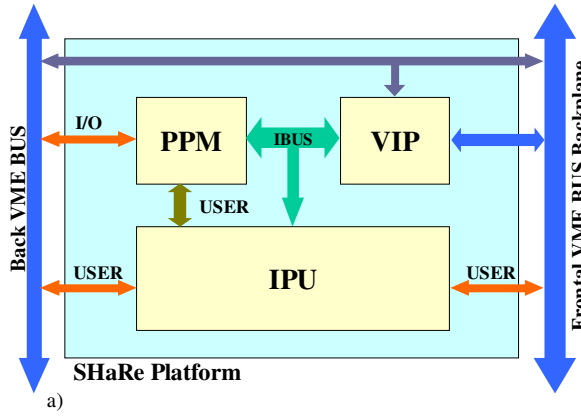

Fig. 1. General SHaRe block diagram

The most important block from the point of view of managing physical resources and system scalability is the VIP one. It provides support to the different buses defined in VME standard. The accessibility to all these buses together with the potential capabilities of the rest of elements gives the system the ability to build a complete VME structure (needless to say that without data sources or sinks that structure is useless) with transfer rates beyond 60Mbytes/sec.

The VIP module provides the necessary link between the VME bus and the inner part of the SHARE board through the internal bus (IBUS: Synchronous Burst User

Interface). It also manages the reprogramming facilities of the FPGAs included in the rest of modules. The VIP FPGA is the only one that starts-up automatically, and only automatically, (in normal operation) to make the system functional at power-up.

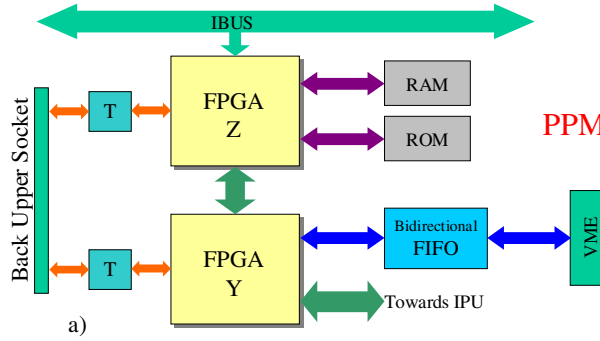

Fig. 2. PPM module internal structure

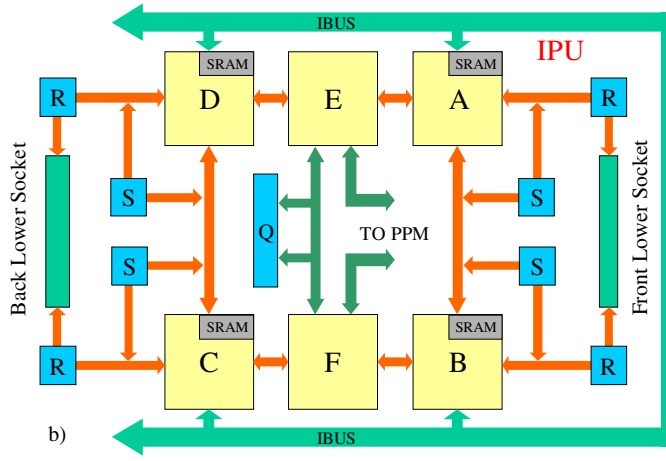

Fig. 3. IPU module internal structure

A simplified diagram of PPM and IPU appear in Fig. 2 and Fig. 3 respectively. PPM module is built around two FPGAs, and IPU one is basically made up of six FPGAs, each of them constituting a cluster. PPM module is designed to be able to perform CPU-like tasks and to communicate through serial/parallel ports to external environments. As the serial/parallel ports can be redefined, no limitations are introduced about used interface or speed (some FPGAs can easily reach 100Mbit/sec serial interfaces), thus allowing an easy migration to different communications protocols (e.g. Ethernet, X.25, RS232, IEEE-488 etc.) only adding the appropriate physical level link. This is because no drivers at physical level have been introduced to have available a wide range of possibilities. Then, the corresponding drivers, depending on the interface required, must be appended at the socket.

With regard to what has been called CPU-like tasks, the idea in mind is having the possibility to introduce a small processor core into one or both FPGAs to execute a

rather simple managing program. The aim is not getting the performance of a commercial processor (otherwise placing one instead of the FPGA would have been a better approach) but being able to check how an FPGA working this way can help the system. For this purpose, up to 1Mbyte of SRAM and 512kbytes of ROM have been supplied. Not only managing or CPU tasks can be assigned to PPM but also signal processing tasks interacting with IPU.

The last block (IPU) consists of an FPGA network connected in a software-defined basis. The main processing will be assigned to this matrix. As a processing unit, will include most of digital processing system capabilities, which don't need to be the same along application's life since in this case the structure can be modified dynamically thus allowing reusing the hardware platform. Moreover, several SRAM blocks (up to a maximum of 256kbytes) have been distributed inside this network in order to provide additional support to the processing task.

2.2 Programming Methods

As mentioned above, VIP module programs itself at power-up as the corresponding FPGA acts as glue logic between buses and general configuration registers and utilities. This option can also be used for PPM block which is useful when building a stand-alone system, including a CPU, or in similar cases. For IPU module, no self-programming possibility exists.

All PPM or IPU FPGAs have the possibility to be programmed individually by means of a host processor over VME bus through functions implemented into VIP. This can be done, at any time and as many times as required, in a similar way as code for general purpose processors is loaded into memory. This feature allows to an external process to determine when any functionality must by modified and then load the correct code into any of the FPGAs building PPM or IPU clusters. Notice that any FPGA into PPM can only be dynamically reprogrammed if autoboot is disabled for it.

Much more important than the external reconfiguration is the possibility of implementing it internally. From the figures shown, VIP connection to the internal bus gives the opportunity to PPM module to access the resources that allow to program/reprogram the FPGAs over the board. The strong interaction of PPM with IPU allows it to monitor application-dependent variables for watching system evolution. Then, when a modification in any of the clusters into IPU is required, a programming cycle can be done. This ability is which we have called auto-reprogram. Of course PPM can participate in the application tasks, but the designer can take advantage of that capacity to build an auto-managed application.

Each IPU cluster may require several kilobytes of data. Storing mechanisms have been provided to PPM (the same resources that allow it to act as a CPU) to retain different configurations of more than one cluster. When the diversity of configurations required for the present application exceeds storage capabilities, more data can be found into VME domains through VIP IBUS-VME bridge.

In any case, reconfiguration is parallel to any processing since a dedicated programming bus has been installed on the board. At any point of the application some devices can be working while others are being reconfigured. A certain mechanism puts reconfiguracion on record so that working application may wait until a concrete part is ready.

2.3 Soft Radio Architecture

Until this point a description of SHaRe's architecture has been presented to allow a general understanding of the system. The blocks depicted have been considered flat representations of the ensemble. But the blocks can be reordered and represented in another way. This representation will give a hierarchical representation per layers of SHaRe and is shown in Fig. 4. It can be seen the presence of three layers. The lower one will perform basic system functions (modulation/demodulation, bitstream processing, etc.). This corresponds completely to IPU. The next level above will deal with higher system tasks (channel parameter extraction, synchronism functions, etc.) and is shared between IPU and PPM. Finally, the top level will manage the part of application running over the board. This task will be exclusively performed by PPM.

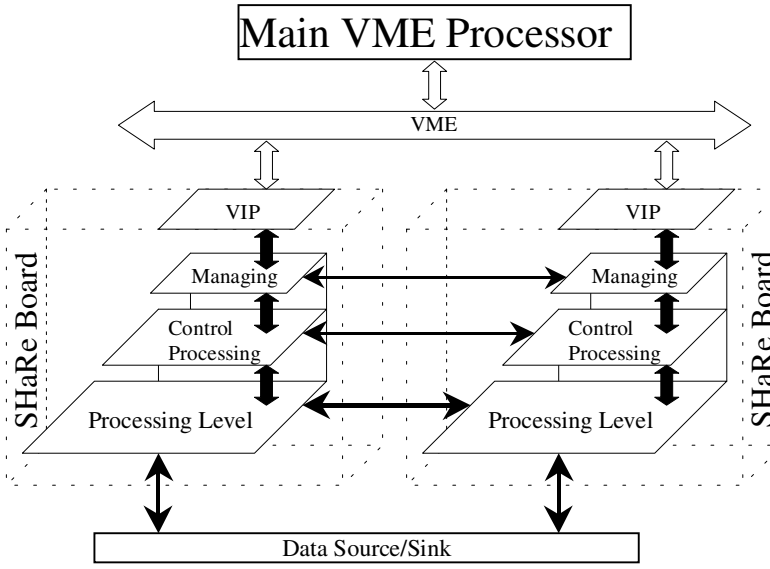

Fig. 4. SHaRe Hierarchy Stack

Viewing the architecture this way it can be understood that the system can modify parameters corresponding to any of the levels through data being received from either the channel or the system administrator. This approach is useful in a canonical Software Radio transceiver because evolution from environment analysis is a key point. Evolution is not only done online by the same system but is also done through system operation enhancement.

Another important aspect to highlight is the system scalability that provides SHaRe. Although it is given the complete set of resources to cope with a complex application, those can be insufficient for the current one. SHaRe gives a simple solution to this problem. A set of boards can be joined as showed in Fig. 4 getting a more complex system but also a more powerful processing machine. It is important to see that the hierarchical structure is kept.

2.4 Data Communication

As main paths of data between clusters we define dedicated paths allowing word transfer widths of up to 32 bits (if data is transferred synchronously several hundreds of megabytes per second can be achieved although generally not required), large enough for most of signal processing applications even if complex data are used. These datapaths are specially useful when signal sampling rates are high and give the structure the capability of sequentially concatenate different blocks, each one of them performing a different task, or implementing parallel paths with cross-passing information.

These dedicated paths would be the main data flow, but to transfer data at lower mean rates, as it can happen when sending control parameters or tables to be updated from time to time, a shared bus structure is acceptable. In our case, as observed in the figures, several clusters have access to a common IBUS. This bus is mainly for bidireccional interaction between PPM and the external world, represented as a VME bus, but it can optionally be used to send and receive data from IPU clusters.

IBUS allows the transaction of data synchronously at high speed rates (128Mbytes per second with 32MHz clock) with a simple mechanism that uses few resources into the FPGAs. Using an address and data multiplexed scheme, only some few handshaking signals are used between bus master and slave, having a very simple protocol. The method allows new and more complex versions to interact with older ones and gives a flexible way to deal with a large amount of addressable resources. By other hand, it is important not spending a lot a resources in the transfer of data (configuration and/or processing) to leave as much room as possible to the application. Even when using the smaller of possible FPGAs, IBUS master and slave occupies less than 10% of resources (additional resources may be necessary depending on the options implemented).

2.5 Applications, Software Tools and Current Works

An example application developed is the uplink in a indoor mobile multimedia DS-CDMA system which represents a system of complexity similar to that of WCDMA for UMTS. In the transmitter a maximum of 256kbps were allowed per user in eight different channels of 32kbps separated each other by Gold sequences. Data were spread at 4Mchips/sec and, after a QPSK modulation, translated to the digital to analog converter at a sampling rate of 32Msps. The receiver, after analog to digital conversion, performed the corresponding tasks of synchronization (lock and follow) and CDMA demodulation. The complete system gross processing demand is about 2Giga operations (some more complex than others) per second in the transmitter and about twice as much in the receiver. Most of processing capacity was used in channel and I/Q demodulator filtering in reception (32Msps and 9-11 bits). The whole transmitter could be inserted into a XC4013 FPGA. Also the synchronism algorithm and CDMA demodulation could be implemented into a XC4013. Notice that this is the smallest FPGA that can be placed over SHaRe. Reception I/Q demodulator and filtering was using almost three times more of space depending on the algorithm used. Using larger FPGAs and taking into account that the complexity (in terms of resources) of the system implemented does not increment as fast as the number of

users does, it is deduced that over a single SHaRe board some tenths of users can be implemented.

By other hand, software tools are important when managing a complex hardware structure. The knowledge of every corner of the structure is hard for novel users but is the only way of doing it without an automated tool. At present developing and programming applications can be done through definition of the different cluster tasks and loading the code generated into FPGAs. The code is sent through a driver running on a Sparc processor card over VME bus with a Solaris operating system. The definition of FPGA code can be done by means of a hardware description language (VHDL).

At present, a tool based on IEEE 1149.1 (Boundary-Scan and Test Access Port) is being developed which will allow the user to test, debug and monitor applications remotely or locally with the help of analysis systems. This can be done from the top of the hierarchy presented before.

3 SHaRe Features

The main Share board features and performances are described in the Table 1. The most important features indicates us the capacity that the system has to communicate with the environment and the maximum processing capacity that can be obtained.

Table 1. SHaRe's set of features

Description	Feature
Host bus	VME64 (ANSI/VITA 1-1994) Up to 66Mbytes/sec.
Test Port	JTAG TAP (IEEE 1149.1)
Board's Devices	Up to 9 Xilinx® FPGA
Devices Family (5V or 3.3V)	Xilinx® XC4000E/EX/XL/XLA (4013-4085) Xilinx® Spartan/XL (30-40)
SRAM Memory per board	2 Mbytes maximum
ROM Memory per board	8 Mbits maximum
FIFOs input/output size	From 2k x 32 bits up to 64k x 32 bits
Number of MACs[1]	Up to 40 Giga MACs with larger devices
IBUS mean transfer speed over sustained transfer	Up to 110 Mbytes/sec with 32MHz clock.

4 Conclusions

Partially reconfigurable platforms with high processing capabilities can help supporting many standards and applications by reusing the same hardware. Software Radio techniques can take advantage of the efficiency of reconfigurable or self-

[1] Estimated peak capacity per full board based on a Xilinx® 16 bits FIR filter benchmark. (MAC: *multiply-accumulate*). One DSPTMS320C6x can perform about 0.5 Giga MAC/sec.

configurable platforms to improve system behaviour without lost of flexibility. The diversification of services, protocols and access topologies will require the introduction of clever multiband and multi-standard terminals able to implement new applications without hardware modification, so partially reconfigurable systems will likely be next generation terminals. In this paper a partially self-reconfigurable system (SHaRe, Fig. 5) has been presented based exclusively on FPGAs. As a testbed, SHaRe will allow checking multiple digital environments and, as a tool, will allow to add quickly and easily new improvements and functions into commercial mobile terminals by using hardware description languages (e.g. VHDL, Verilog) in a first step and even high level languages (like C/C++ or JAVA) for system-level description. SHaRe's properties make it suitable for checking auto-reconfiguration algorithms on the basis of self-evolution control.

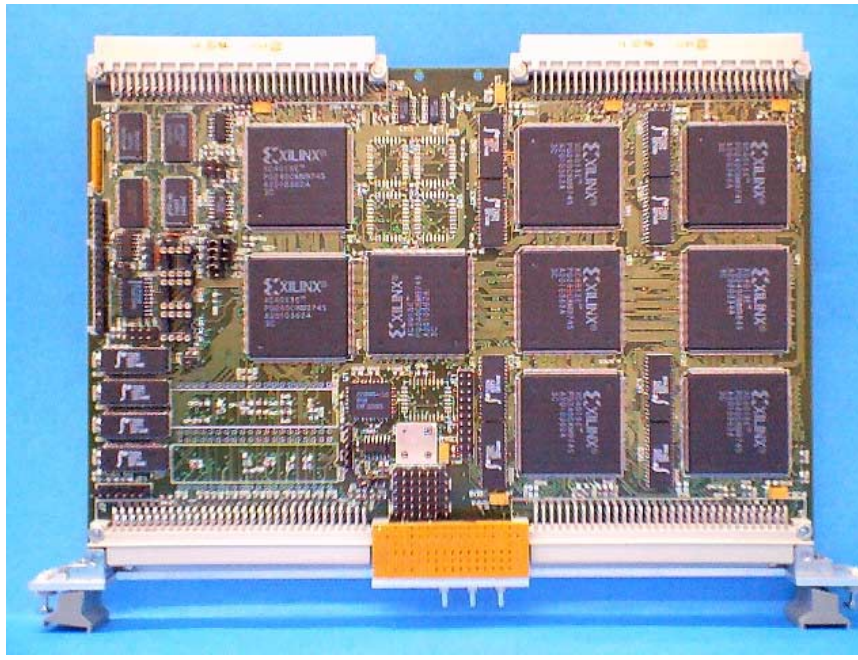

Fig. 5. Frontal photograph of first SHaRe prototype

Acknowledgement

This work has been supported by CYCIT (Spanish National Science Council) under grant TIC98-0684

References

1. Joe Mitola. "The Software Radio Architecture". IEEE Communications Magazine. May 1995.
2. Srikathyayani Srikanteswara, Jeffrey H. Reed, Peter Athanas and Robert Boyle. "A Soft Radio Architecture for Reconfigurable Platforms". IEEE Communications Magazine. February 2000.
3. Mark Cummings, Shinichiro Haruyama. "FPGA in the Software Radio". IEEE Communications Magazine. February 1999.
4. XILINX XC4000E XC4000X Series Field Programmable Gate Arrays. Xilinx. May 1999.
5. VME bus Specification, ANSI/IEEE STD1014-1987. VITA 1987.