Reconfigurable Hardware Platform for Software Radio Applications (SHaRe) in Mobile Communications Environments

Xavier Revés, Antoni Gelonch, Ferran Casadevall

Universitat Politécnica de Catalunya

Signal Theory and Communication Department

Jordi Girona 1-3, 08034 Barcelona (Spain)

Tel.- +3493 4017197   Fax.- + 34 93 401 7424

e-mails.- xreves@xaloc.upc.es ; antoni@xaloc.upc.es; ferranc@tsc.upc.es

Abstract

The third generation of mobile systems will bear a wide range of services, including multimedia applications. To support the envisaged user requirements different service quality, radio environments and network topologies are needed. As a result, to confront the technological challenge that represents designing radio systems with a wide range of possibilities and features the Software Radio techniques have appeared , which require a high computational power and flexibility. In this context, to test in real time the radio access technologies proposed and to evaluate its performance, a reconfigurable hardware platform has been developed. Based on DSPs and FPGAs, it is intended to cope with flexibility and computing needs of a UMTS Software Radio application.

Introduction

The third generation of mobile systems will cope with a wide range of services, including multimedia applications, each of them requiring a specific service quality. Moreover, the services will be oriented to packets and circuits. They must operate in all the radio environments whatever it is, macro, micro and picocellular, and provide service in any moment guaranteeing the mobility of users.

To confront the technological challenge that represents designing radio systems with this wide range of possibilities and features, a new technique, so called Software Radio, have appeared as a possibility every time more attractive. It allows freeing the functional qualities and the offered services from the physical characteristics of the terminal. Its implementation is based on a generic hardware platform, able to adapt itself to the particular characteristics of the radioelectrical environment wherein the radio will operate, as well as to the net structure and services of different providers. Then, implementing any processing function of the transmitted/received signals by means of software programming techniques, the radio terminal will be provided with the required functions for every one of the applications considered.

On the other hand, and in parallel with the technological requirements of the third generation of mobile communications systems, the term Reconfigurable Computing (associated to reconfigurable logic) has increased in popularity in high technology environments. This is due to the development of so-called FPGA (Field Programmable Gate Array) devices that allow the integration of a large amount of logic gates and registers into them. Nowadays FPGAs able to store about 1 million gates and several tenths of thousands of flip-flops can be found [1]-[2]REF. The availability of some supporting software that eases its programming makes relatively simple its use for any digital application.. Then, if you wish to rise the computational capabilities of the system, you must organise the set of FPGAs setting up a structure flexible enough to bear different applications.

REFThe previous characteristics we have seen, that is, re-programmability and capacity, give to these integrated circuits a special attractiveness in software Radio applications because we can assign them intensive and repetitive processing tasks (filtering, coding, modulation and/or demodulation). They are able to carry out those typical tasks of digital processing nearly at specific hardware speed and with software flexibility. Their use in communications applications (or processing in general) allows reducing costs without giving up working in applications, that require an intensive computing capability, as it will likely happen in the third generation of mobile communications systems. Indeed, comparing typical features of a digital system build up of Digital Signal Processors (DSP) and hard-coded logic into Application-Specific Integrated Circuits (ASIC) (as it is being generally done until now) we can deduce the important advantage that represents using reconfigurable logic [1]. This offers a reduced power consumption, cost and size and, by the other hand, it is ease to upgrade as silicon evolves. This is the point of view of a fully flexible implementation using FPGAs or even DSPs. But when only a software radio with limited functions is desired the ASIC option costs less [3].

Although FPGAs are opening a new paradigm in digital computing, DSPs must not be forgotten as they can increase system performance if particular features of both devices are properly combined. For instance, DSP programming tools are generally more advanced as they have spent about two decades in our laboratories.

Objectives

The final objective of this research work is to develop a reconfigurable hardware platform 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. The basic demonstrator structure is shown in Figure **Error! Unknown switch argument.**. It includes DSPs (Digital Signal Processors) and FPGAs (Field Programmable Gate Array) networks mounted over an standard VME bus, which are devoted to emulate, in real time, the lowest layers of the radio interface of a generic Third Generation Mobile Communication system. The Link and Network Layers will be emulated in each one of the three PCs that carry out the functionalities of two Base Station (BSi) and one Mobile Terminal (MT), which constitute the minimum mobile system to be emulated. Moreover, in order to implement and evaluate the system mobility management, they will be connected through an ATM interface to the ATM switch.

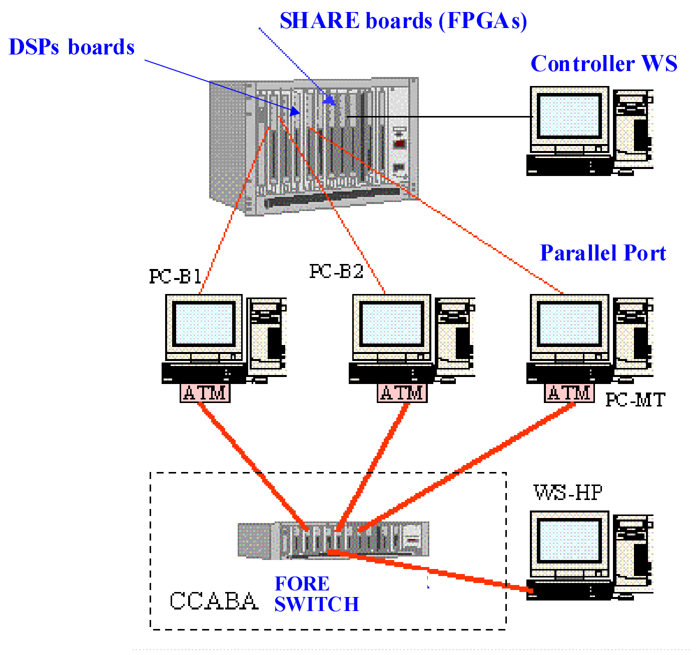

Figure Error! Unknown switch argument.**: Demonstrator structure.**

From the viewpoint of the Software Radio implementation requirements, the agregate processing demand (all users, including overhead) 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) REF[4]. This value include 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. This figures give us an idea of the UMTS system needs.

The work reported in this paper is focused on developing the FPGA network.

SHaRe board

The structure of the FPGA network board is shown in the Figure **Error! Unknown switch argument.**-a. We can observe three different modules: the Peripheral and Processing Management (PPM) module, the VME Interface and Programming (VIP) module and the Intensive Processing Unit (IPU) module, all them are constituted of several FPGAs as the *intelligent* part. The different modules are placed over a printed circuit board with a VME physical interface and 6U shaped (160mm x 233mm) which has been chosen due to its well known flexibility, and robustness. One point to be highlighted with respect to Figure **Error! Unknown switch argument.**-a is the presence of two VME BUS interfaces, the backplane, corresponding to the actual VME bus as defined, and one called optional. This one is only a *virtual* bus that can be seen from some connectors installed over SHaRe board allowing the concatenation of another VME compliant board following SHaRe board. In principle, this feature is introduced to concatenate two or more SHaRe boards but concatenation is not limited to them. Nevertheless, care must be taken with respect to VME characteristics of the concatenated board as it will be commented ahead.

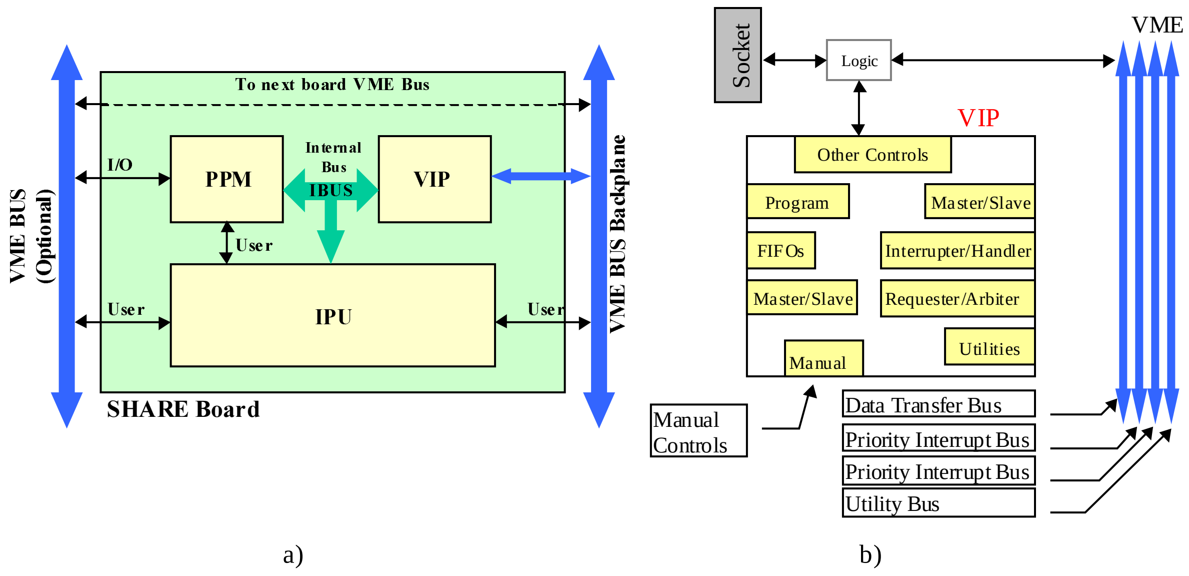

Figure Error! Unknown switch argument.**: SHaRe board structure (a) and VME interface detail (b).**

The VIP module provides the necessary link between the VME bus and the inner part of the SHARE board. This block not only carries out most of functions described in the VME specification but also manages the reprogramming utilities of the FPGAs included in the rest of the modules. To allow this to happen, as VIP has an FPGA as main element, capabilities to wake up automatically must be supplied to have the system functional immediately after power-up. This is because an FPGA stores its configuration into SRAM [5]. As a consequence, every time the system is re-powered the configuration must be downloaded into FPGA's internal configuration SRAM. This is the *drawback* of reprogamability since the possibility of modifying the configuration memory of the FPGA, that allows the behaviour modification in real time[i] (or simply dynamically reconfiguration), also makes mandatory an initial system configuration much in the manner of a booting process. In the same way, the rest of FPGAs must be programmed every time the system is restarted or every time the function has to be modified. This can be performed in several different ways: semi-manually (when debugging), by means of the VME bus using standard writing cycles (in an individual manner) or, in certain cases, automatically to give a full range of possibilities. This allows to a remote element (e.g. a processor over a VME bus connected to a file server) to program the board to perform the desired task, previous design and loading of the FPGA contents.

Figure **Error! Unknown switch argument.**-b shows a more detailed scheme of VIP module contents. The different buses defined in VME standard REF[6] can be observed. The accessibility to all this buses together with the potential capabilities of the rest of elements gives the system the ability to build a complete VME

[i] Real-time modification depends on the application. Those having less than about 10ms-50ms (depending on the FPGA used) from cycle to cycle won't be modified in real time. At present, FPGAs with lower reconfiguration times are being issued.

structure without external elements, excluding file servers, data processing sources and data processing sinks. This feature has been included to avoid the mandatory presence of another board over the VME bus to perform the most elementary tasks defined in the specification, such as reset, arbitration, interrupt driving, etc.

Each one of the functions defined into VIP block (see Figure **Error! Unknown switch argument.**-b) are implemented into a single FPGA. The relationship among them is not depicted but it can be understood that those concerning bus operation are controlling those that perform activities over the system. Among them, it must be highlighted the Program utility, which is accessible from VME bus, from internal bus (IBUS) and from Manual Controls, and supports the programming modes introduced previously.

Another interesting aspect represented in the figure is the VME bypassing because it allows system expansion in larger (depth expansion) racks to increase the system performance as commented previously. The Socket block of the figure represents the connectors over SHaRe making possible VME bypassing. This Socket is compound of an Upper Socket and a Lower Socket corresponding to both 6U board's connectors. It must be said that VME capabilities of the added board are limited from Socket point of view as some functions are not available (master operation not allowed) and the performance is slightly reduced as electrical bus integrity has to be kept (increased delay due to intermediate logic). Expansion can only be performed by concatenation of a VME compliant board at SHaRe's back being able to work under these limitations, as another SHaRe board can do. Power increase is not intended through VME connection but through dedicated lines as it will be explained further in this point. Therefore, limitations in this interface must not limit the processing capacity of the concatenated system.

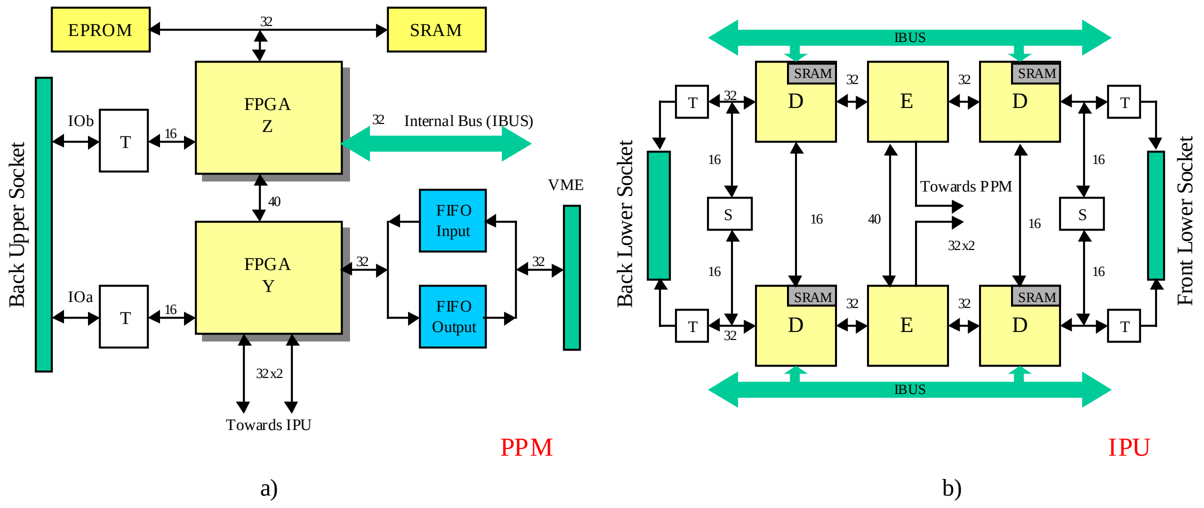

Figure Error! Unknown switch argument.**: Main module (a) and Processing unit (b)**

Figure **Error! Unknown switch argument.**-a shows the PPM module, built around two FPGAs. This module REFis designed to be able to perform typical CPU tasks and to communicate through serial/parallel ports (requiring up to a maximum of 16 or 32 control and data lines depending on whether using two ports or only one, respectively) to external environments. As the serial/parallel ports can be redefined, no limitations are introduced about used interface, 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 be available a wide range of possibilities. Then, the corresponding drivers, depending on the interface required, must be appended at the socket. It must be noticed that is not possible to concatenate a VME board following SHaRe and use the serial/parallel ports at the same time as both use the same sockets, but this is not a limitation because peripheral communications will generally be carried out by the top board in the concatenated chain (the farthest from VME backplane). On the other hand, we used a FPGA instead of a CPU because we pursuit two objectives: to work as a CPU when it is required for managing tasks, introducing the corresponding CPU core inside the FPGA (as definable silicon, an FPGA can be configured to behave as a CPU), and to use it as an intensive task processor in others situations. To perform CPU functions in a right way, some SRAM and EPROM memories has been added to the prototype, as it can be observed in the figure. Analogously to perform intensive tasks a strong connection to the Processing unit has also been provided. Finally, the module includes one bidireccional FIFO to provide a more flexible interface with VME bus (VME side controlled by VME Interface in Figure **Error! Unknown switch argument.**-b).

The third block, IPU, showed in Figure **Error! Unknown switch argument.**-b, is compound of six FPGAs interconnected among them in a variable way that can be easily modified by software. The main processing will be assigned to them. This processing unit will include all digital processing systems capabilities, which generally are the same along the application live. But in this case, the parameters of the application or the same application can be modified dynamically thus allowing reusing the hardware platform. Although physical paths are already determined, the way they are used makes possible choosing among several processing topologies, so the application designer can select the most suitable one. As data-paths of 32 bits are available, external communication bandwidth is highly increased. Moreover, several SRAM blocks have been distributed inside this network in order to provide additional support to the processing task. Certainly, the distributed memory placed all round the processor is useful to avoid bottlenecks, derived from using a central memory pool, and increase memory bandwidth.

Finally, the possibility to concatenate more than one IPU has been considered. This feature is an immediate consequence of the possibility of concatenation presented above. Observing **Error! Unknown switch argument.**-b you can see the symmetrical structure of the IPU module. As well as in the front of the structure (connected to VME backplane through the lower front socket when standing alone) , the rear part has its own socket to concatenate two (or more) IPUs (board1 back joined to board2 front and so on). Data are passed through the user lines described in the VME bus standard. This feature gives the possibility to scale the system to tolerate macro applications. But expansion can be achieved not only using more SHaRe boards but also using any other board with data connection to VME user lines either being VME compliant (as mentioned before) or not[ii].

Loading and updating DSP&FPGA network

The SHaRe system, designed as a tool for testing new techniques and applications on mobile technology, have in the requirements of loading code one of the most important points. It must be easy to load, upgrade and change certain parts without modifying any other. For example, it could be interesting in an Third Generation Mobile Communications system to change, on real time, the air interface. Then while a part of the software/hardware is just receiving the code through the present air interface, other parts of the terminal must be modified in order to support the second air interface. Up to this moment is not possible to change the full configuration of the system from cycle to cycle due to reconfiguration speed limitations. New devices and techniques will make it possible depending on the cycle rate. With the requirements of loading code in mind, we have supported several loading modes, such as by means of VME bus, by means of JTAG interface, from the DSPs commports, from a remote element in a network, etc. About the code generation we are currently investigating how to distribute easily the tasks between the DSPs and the FPGAs automatically by means of any standard language like VHDL, C or JAVA. Firsts stages of the work will be devoted to lower level languages. As soon as they become solid platforms for higher level languages these will be introduced to give more power and reliability to the ensemble. It is clear that it is easy to distribute tasks manually but the different kind of devices involved and possible final performance makes of this automation task and interesting goal.

DSPs and FPGAs links

It could be helpful to connect the DSPs network and the FPGAs network in some way. It is clear that each kind of device takes advantage over the other in some applications. Then, in the case of using DSP devices TMS320C40 from T.I., an easy and flexible way to interconnect them is through the DSPs comports, which are parallel ports 8 bits wide, that can be managed by DMA and/or interrupts. So, the final network is defined by the wired links between each one of the devices, DSPs or FPGAs and can be modified according to the system requirements.

[ii] If not, VME lines must not be disrupted.

SHARE features and performance

The main Share board features and performances can be described by the Table **Error! Unknown switch argument.**. These features can be modified in some cases by silicon evolution as stated in the introductory paragraph.

Description	Feature
Communication Ports, Buses	VME (IEEE 1014), Definable Serial Port
Test Port	JTAG TAP (IEEE 1149.1)
Board's Devices	Up to 9 Xilinx® FPGA
Devices Family	Xilinx XC4000E/EX (4013- 4036) Xilinx XC5200 (5210 - 5215)
SRAM Memory per board	2 Mbytes maximum
FIFOs input/output size	From 2k x 32 bits up to 64k x 32 bits
Number of MACs[iii]	Up to 8-12 Giga MACs with XC4036EX devices
VME Maximum transfer speed	Up to 66 Mbytes/sec

Table Error! Unknown switch argument.**. Share Board features.**

Conclusions

The Software Radio techniques involve a set of processing performance requirements that can be hardly achieved for testing by traditional digital systems based on DSPs. FPGAs are opening a new door in the digital processing environments and offer a new basis for high power processing applications. They are every time more attractive as FPGA-related tools performance and availability increases at a dramatic speed. The combination of both digital signal processing devices (DSPs and FPGAs) can take advantage of their respective features. In this paper we have presented a reconfigurable system (SHaRe) based exclusively on FPGAs which can bear the hardness of third-generation strategies for global communications. 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 terminal by using hardware description languages (e.g. VHDL, Verilog) as a first layer and even C o JAVA languages in higher and mature levels.

References

[1] Mark Cummings, Shinichiro Haruyama. "FPGA in the Software Radio". IEEE Communications Magazine. February 1999.

[2] XILINX Xcell. Issue 31, First Quarter 1999.

[3] Joseph Mitola III. "Technical Challenges in the Globalization of Software Radio". IEEE Communications Magazine. February 1999.

[4] Joe Mitola. "The Software Radio Architecture". IEEE Communications Magazine. May 1995.

[5] XILINX XC4000E XC4000X Series Field Programmable Gate Arrays. Xilinx. June 1996.

[6] VME bus Specification, ANSI/IEEE STD1014-1987. VITA 1987.

ACKNOLEDGEMENT

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

[iii] 16 bits FIR filter Estimation (MAC: *multiply-accumulate*). One DSPTMS320C6x can perform 0.5 Giga MAC/sec.