Soft-Core FPGA Processor Based Platform for Embedded Systems Rapid Prototyping

L. Guedria, D. Hubaux
Centre d'Excellence en Technologies de l'Information et de la Communication (CETIC)
8 Rue Clément Ader
Charleroi 6041 Belgium
lg@cetic.be
dh@cetic.be

J.-D. Legat
Université Catholique de Louvain / Unité des Dispositifs et Circuits Electroniques (DICE)
3 Place du Levant (Maxwell buil.)
Louvain-la-Neuve 1348 Belgium
Legat@dice.ucl.ac.be

Abstract

Rapid prototyping is a highly needed capability in today embedded systems design in order to validate requirements, succeed the design and fulfill time-to-market constraints. Efficient and flexible hardware/software platforms are the ultimate support to achieve this goal. In this scope Soft-Core FPGA processor based Platforms constitute an interesting approach thanks to customization capability offered for both the hardware and software parts of the system.

This paper describes a soft-core FPGA processor based platform targeted for embedded systems prototyping. It shows how it is possible to achieve very short development cycle times with such platform. It details the key characteristics that are most relevant with regard to fast and flexible prototyping approach.

Keywords*: embedded system; soft-core; FPGA; flexibility; rapid prototyping.*

1. Introduction

With the advances of today technology, embedded systems design is getting more and more complex: Systems are implementing a continuously increasing number of advanced functionalities and integrating more and more interfaces at both hardware and software levels.

Hence embedded systems design has to master this complexity while being able to efficiently address time-to-market constraints. Also it should be able to quickly adapt to changing needs and new or additional demands.

In this scope, rapid prototyping is necessary to achieve the first challenge. However, for the second one, high flexibility is highly needed.

Some prototyping approaches of embedded systems design, mainly guided by performance issues, tend to rely on platforms whose hardware is very close to the final product one: This way, the design process could focus on tuning and optimizing software with regard to the almost fixed hardware system architecture.

These approaches are not well adapted for embedded systems which may need to support new or evolving interfaces or connectivity while keeping their main functionality: For instance, an embedded data logger which needs to support a new wireless communication interface.

Platforms based on soft-core FPGA processor offer an alternative which combines the benefits of a mastered prototyping environment and a flexible support allowing easy extension of hardware features without heavy penalty on software adaptations.

The next chapter identifies relevant issues about rapid prototyping of embedded systems.

Chapter III describes a flexible platform developed by CETIC and dedicated for rapid prototyping of embedded systems. It also enumerates the key benefits of this soft-core FPGA processor platform within the scope of rapid and flexible prototyping.

Chapter IV presents a relevant use case of such platform for rapid prototyping of an In-Vehicle telematics system (IVTS).

The last Chapter discusses the advantages and limitations of the platform and identifies the most suited application frameworks for it.

2. Rapid prototyping of embedded systems

Several definitions could be given for *prototyping* [1] [2] depending on the application domain. A slightly global and general one could be: The construction of an intermediate system to demonstrate, evaluate and validate some aspects of the intended system behavior in order to gain user acceptance or to establish technical feasibility.

Rapid prototyping may be defined as a set of techniques aimed to accelerate the prototyping process in the objective of shortening the design cycle of a system while maintaining efficiency. Main advantages taken from rapid prototyping are risk management and early end-user involvement.

For embedded systems, rapid prototyping issues apply to both hardware and software parts [3]. The integration aspect is also highly important and should be addressed carefully.

Techniques used in embedded systems rapid prototyping give high importance to efficient methodologies elaboration. In [4] a concurrent design methodology is presented and applied to the rapid prototyping of wearable computers. It allowed achieving fast design cycles thanks to an optimized management of time and resources. This results also in a higher level of concurrency in the design process.

In [5] a comparison is elaborated between custom and off-the-shelf design methodologies on basis of six key characterizing attributes for embedded systems: features overhead, cost, person effort, power consumption, storage requirements and software portability.

Besides the design methodology, the approach adopted in rapid prototyping for embedded systems may differ. One approach consists of developing a draft implementation dedicated to the early specification phases in order to learn more about the requirements. The prototype is then "thrown-away" and the production final system is developed on basis of experiences from the prototyping effort. The other approach targets the development of a high quality prototype that evolves over time.

Both approaches present some problems [6]. The most common one with "throw-away" prototyping is managerial. In many projects, the development of a throw-away prototype, primarily intended for experimentation purposes, evolves under timing constraints pressure, to a risky and difficult tentative to deliver it as a production system. This prototype misuse inevitably leads to poorly structured designs, hard to validate, maintain or evolve.

Evolutionary prototyping requires, till the start point, a good and accurate evaluation of the potential evolution scope of the system. Usually this is difficult to identify at first stages but decision has to be made. Often, this approach leads to either an over complicated prototype with lot of unnecessary evolution capabilities or a too limited prototype unable to handle, without lot of difficulties, additional or evolving features.

In both cases this leads to inadequate prototype with regard to the first intent.

In [7] an Integrated Design Environment (IDE) approach is presented for the rapid prototyping of In-Vehicle telematics systems (IVTS). It is aimed to tackle difficulties related to system complexity and resources management. Rapid prototyping is seen here as a technique for validating system requirements and improving specification. The approach considers the "throw-away" prototype as a basis for the phases of system design and implementation. Off-the-shelf components based design is presented as an inherent characteristic of IVTS and is adopted in order to master the prototyping process duration.

Prototyping of IVTS specifically and other embedded systems in general usually requires system to be extensible and flexible by offering facilities to add, remove or upgrade components preferably in a "plug and play"-like way.

We believe that succeeded prototyping process of many embedded systems would rely on both:

- Tools to master complexity and accelerate design cycle mainly for the software part (like in [7])
- Availability of a flexible platform that offers powerful customization capabilities while remaining well tailored to the target application.

3. Soft-core FPGA processor based platform

3.1. Platform hardware

Our rapid prototyping platform, named SAND, is a multi board system consisting of a main board for the system kernel and stackable extension boards for peripheral and external connectivity support. Figure 1 illustrates the stackable structure of the SAND platform.

Main board is based on a low cost FPGA device (Altera Cyclone II family). It integrates SDRAM and flash memories, watchdog and power supply stage. The board size is compact (8x4 cm) and presents at its

borders the I/O connections for extension boards. Table 1 summarizes the main characteristics of the main board.

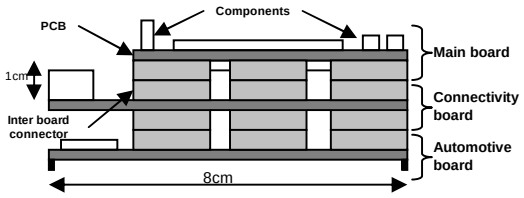

Figure 1. SAND Stackable Boards Structure

Extension boards are customizable depending on the application connectivity needs.
Two boards were developed:

- A connectivity board with integrated wireless (WiFi, Bluetooth and GPS) and wireline (Serial, USB and OneWire) connectivity.
- An automotive dedicated board supporting connection to two standardized open automotive interfaces: *Fleet Management System (FMS)* built on top of CAN bus and implemented in trucks and buses, and *On Board Diagnostic (OBD)* mainly implemented in cars. This board integrates also some extra features such as accelerometer and Analog/Digital I/Os. The use of this board with regard to the whole platform is detailed in chapter IV.

Table 1. Main board characteristics

feature	family	range/size
FPGA	Cyclone C6/C12	6k/12k LE
Flash mem.	NOR	32 Mbytes
Config Mem.	EPCS	8MBits
RAM	SDRAM	16MBytes
Power stage		Input: 4-35V/2A
I/Os	GPIO	64
Soft-Core	NiosII	
PCB	4 layers	4 x8 cm

3.2. Soft-core FPGA processor

A soft-core processor is a customizable software description of processor hardware which could be synthesized and implemented on programmable logic ICs like FPGAs. For instance: the Nios II by Altera, the MicroBlaze by Xilinx, or other cores under LGPL licence (Leon II, OpenRisc, etc.).

Soft-core processors ensure a high flexibility and reconfigurability since they are implemented as customizable blocks inside the FPGA. For instance, it is easy to add a serial port or a USB MAC layer in the FPGA without changing anything to the board, but only changing the configuration of the FPGA.

The configuration of the FPGA (including the soft-core processor) is kept in a special flash memory used at boot time. This means that the processor can modify its own configuration in flash memory so that a new processor can be started at next startup. Another advantage of this technology is the possibility to instantiate several processors within the same FPGA. Current boards accept up to six processors being created inside the FPGA.

3.3. Platform rapid prototyping key features

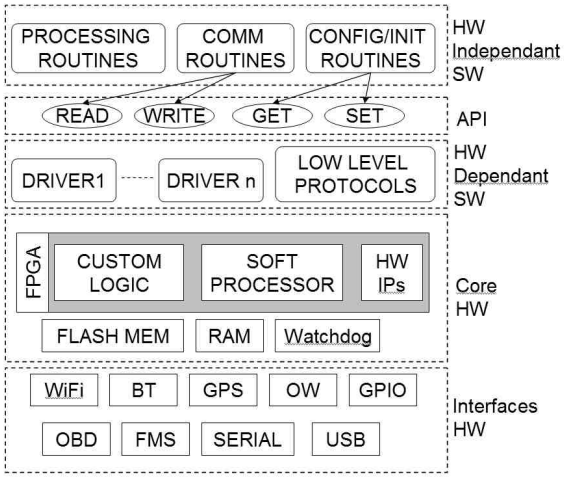

Figure 2. Platform HW/SW architecture

The platform architecture, illustrated in figure 2, is based on a comprehensive separation between core and communication functionalities on the hardware side, and between hardware dependant and hardware independent parts on the software side. This enables interesting features like:

Flexibility: As stated above, flexibility is a key feature for succeeded rapid prototyping of many embedded systems.
As discussed in chapter II, prototype flexibility level should be carefully conceived. It should not be uselessly higher than needed or uncomfortably too low. Flexibility is aimed to allow quick and easy design adaptations.

The SAND platform flexibility is manifested through the following points:

- Inherent flexibility of the soft-core processor which allows easy and fast support of new interfaces at both the software and hardware levels.
- Inherent flexibility of FPGA technology allowing fast integration of hardware blocks (standard or custom IPs).
- Availability of several configurable I/Os from FPGA to the extension boards.
- Easiness of building software application even when platform hardware changes (i.e. new processor configuration or new peripheral added) thanks to the modular software architecture.

Extensibility: The SAND platform was designed on the principle of stackable boards. This allows customizing the prototype to the target application by adjunction, to the main board, of one or more boards tailored to the features and connectivity requirements of the final system.
Hence, this modular architecture of the platform enables focusing on design of specific features and avoiding reinventing the wheel for general features. This obviously enhances the rapid prototyping process.
Furthermore, potential bad design choices would only impact a limited part of the system and are less fatal to the overall design process.
On the software side, the platform structure allows for comprehensive support of new hardware at the hardware dependant software level. At the application level, writing communication and configuration routines is eased thanks to the API availability.

Reusability: The main board which constitutes the system kernel is a general purpose board that could be reused, as it is, for large number of prototypes. This is due to the fact that the board includes a common sub-structure for large number of systems that is the processing element(s) and the memory.
The "soft" customization capability of this board at processor and FPGA levels improves its reusability.

Upgradability: This is the ability:
a. To replace (a software program) with a more recently released, enhanced version.
b. To replace (a hardware device) with one that provides better performance and/or additional features.

In the SAND platform this is enabled at different levels:

- The hardware configuration of the FPGA (including the soft-core description, standard and custom IP blocks, and I/O configuration) is easily upgradable by programming the new configuration files into the board.
- The software program for the processor is, in the same way, easily upgradable.
- Some main components (FPGA, flash memory, SDRAM) could be replaced directly with other components having the same footprint. This allows for fast assembly of customized main board avoiding by the way the design of a new PCB. For example, to support a USB MAC IP, we need a bigger FPGA (a C12 chip rather than C6). Boards can be assembled seamlessly with either chips. The same substitution could apply to SDRAM and flash memories, in order to tailor the prototype to the application requirements.

4. In-Vehicle Telematics System rapid prototyping

4.1. Context description

A company specialized in the coaching of truck drivers, has decided to use telematics in order to gather the required objective data from drivers and vehicle, directly acquired while driving. Concretely, the company was searching for a system able to collect information on the CAN bus of the vehicle (speed, position of the accelerator pedal, braking, etc.) and give an advanced report on the driver's driving behavior. This system would be used for training and also for monitoring (before, while and after the training) so that the results are clearly demonstrated. These results have to be transferred to the final user (the trainer for example) and be displayed comprehensibly on a computer, preferably directly after the training [8]. Data acquisition and all computations are made in real time and stored locally in an optimal way. Data can be accessed in real time inside the vehicle through wireless connection. Data acquisition can either be automatically triggered on a specific trip or turned on all the time.

Since the acquisition of such data needs connection to the CAN bus, an embedded and communicating smart system becomes mandatory.

4.2. System analysis

From hardware viewpoint, the system needs to be flexible: The primary objective was to achieve a rapid

prototype for data acquisition on the CAN bus through the FMS gateway. However, the system have to be enough adaptable to support easy integration of new, but not yet well specified, features like GPS integration, wireless data transmission interface (WiFi or Bluetooth), Temperature sensing, driver identification mechanism, etc.

From embedded software viewpoint, we distinguish two parts:

- Hardware Independent Software: This comprises the computations done on raw data: Statistics, histograms, etc.
- Hardware Dependant Software: This comprises all the low level protocol encapsulations and peripheral drivers.

The two parts should be enough flexible to allow fast and easy support of new hardware interfaces and implementation of additional statistics.

In this scope, the soft-core FPGA processor based platform brings flexibility at several levels: The soft-core processor is easily adaptable to integrate built-in peripherals (Serial interfaces, memory interfaces, custom I/Os, etc.) to connect either to hardware blocks inside FPGA or directly to components outside. Besides, FPGA technology allows implementation of wide variety of custom or standard hardware interfaces and logic (USB MAC, Ethernet MAC, OneWire Logic, etc).

4.3. Prototyping process

The adopted prototyping process was inspired from agile software development methods [9] [10]. These are mainly customer-centric approaches targeting very short development cycles and early and frequent delivery with particular focus on ability to harness changing requirements.

The start points were:

- For hardware part: The already available general purpose main board with the connectivity board.
- For software part: A well structured template project with a clear identification of hardware dependent and hardware independent pieces of software.

An external off-the-shelf interfacing module to the FMS system was rapidly integrated to the connectivity board over its serial link.

Hardware dependant software, consisting mainly of FMS module protocol implementation, was quickly developed.

Hardware independent software, consisting of the an implementation of a first draft of features specification was developed almost concurrently, with the hardware dependant software thanks to the early definition of a flexible data structure serving as data sharing mechanism between the two software parts.

This first prototyping phase was achieved very quickly (few days) and served as a basis for capturing end-user feedback.

New requirements emerged then. They mainly consist of:

- Need to integrate an internal FMS chip to the platform in order to reduce size.
- Implementing a broadcasting mechanism through wireless connection in order to allow some real time parameters monitoring.
- Implementing a control mechanism over wireless link: pinging the embedded system, asking for status, downloading stored data, changing mode, etc.
- Implementing a GPS geofencing mode: i.e. System should be able to start automatically computing statistics based on preconfigured GPS positions settings.

A second prototype was developed in the following way:

Thanks to platform modularity, a new custom extension board was being designed while work on wireless features implementation was being pursued on the first prototype. This separation of concerns allowed a precious time gain with no negative impact on design quality.

Implementation of wireless new features consisted of:

- Implementing a two serial channel links over wireless connection: One for control purpose and the other for real-time monitoring.
- Adaptation of the soft-core processor by adding the needed serial interfaces.
- Design of encapsulation protocol and associated routines for the execution of the different commands.
- Design of a Graphical User Interface on a PC (or Laptop) allowing wireless communication with the platform, mainly for sending commands and displaying the real-time data.

The GPS geofencing mode support was rapidly implemented thanks to availability of hardware and software driver.

The custom board was developed conforming to the form factor of the other boards. Since there was enough space left on it, we decided to add support for OBD too, despite that there weren't an immediate need or demand for it. The motivation was that OBD data could be exploited in a similar way to FMS one. Furthermore

some less important features, but relevant for automotive domain, were added for investigation purposes like accelerometer and analog/digital I/Os.

This approach allowed the rapid setup of the second prototype which was put to tests.

The availability of OBD on the automotive board induced a light overhead (hardware and software interface implementations) however it allowed us to gain lot of time to carry extensive tests on the hardware independent software part of the application: We weren't constrained by the availability of an FMS equipped truck to do it since we were able to experiment directly on OBD equipped cars (more easily available).

The prototype examples are now installed and running on several trucks. They proved to fit well as final systems despite that this was not the first intent when we started the development. The installed prototype examples were assembled with integration of only-needed components. For instance, Bluetooth and OBD chips weren't assembled on them in order to master the overall cost.

Table 2. Short development cycles

Date (weeks)	Customer request	work (days)	Prototypes delivered
0	First customer contact	N/A	0
2	Field tests for feasibility issues	2	0
12	First prototype	10	1
16	Prototype install	1	1
17	Feature update	1	1
20	New features	1	1
25	New features	10	1
28	New features & prototype install	5	4
31	Prototype install & Field tests	3	5
40	New features	10	5
54	Update software & prototype install	2	7
60	prototype install	2	10

Table 2 summarizes the history of the project and shows the fast development cycles we achieved with our 'agile like' approach in our rapid prototyping process based on the SAND platform. As we can see, the platform flexibility allowed very fast response to new requirements or features implementation demands (generally one to three days, and less than ten days for more complex demands). Note That the field tests done on week 2 were essential for our customer to convince some of his potential clients to opt for an embedded solution. He then came back with the first concrete demand for a prototype at week 12.

5. Discussion

Rapid prototyping using our soft-core FPGA processor based platform, proves to be a promising approach. It could bring significant improvements thanks to its inherent offered capabilities of flexibility, extensibility reusability and upgradability.

However, taking full advantage of these capabilities requires a careful approach:

- Attention must be taken to design modularity till the start.
- A reuse thought and mindset must be omnipresent at all design steps.

Based on our experience, achieving rapid prototyping objective relying upon soft-core FPGA processor based platform implies some conditions or prerequisites such as:

- Clear separation between platform processing kernel and connectivity extension.
- Rigorous structuring in software development mainly through clear separation between hardware dependent and hardware independent parts.
- Availability of a comprehensive development environment and tools (preferably integrated within an IDE) that allows a fast and easy support of new processor hardware and software features integration.

With the soft-core FPGA processor based platform, we are in an evolutionary prototype approach rather than a "throw-away" prototype one. However, thanks to inherent flexibility of the platform the side-effects of evolutionary prototyping are better mastered.

On another hand, the prototyping here is a balanced mix between a custom design and off-the-shelf components based one.

If we analyze the approach presented in this article with regard to the six attributes mentioned in chapter II (overhead, cost, person effort, power consumption, storage requirements and software portability) we may conclude the following:

- Overhead and cost could stay mastered giving a structured design methodology.
- Power consumption is a strong constraint in some embedded systems. In that case, the approach is poorly adapted.
- Storage requirements depend on the project type but generally embedded systems design is aware of limited resources. Hence, in most cases our approach remains well adapted.
- Software portability is enhanced by this approach giving a well structured software

design with clear separation between hardware dependent and hardware independent parts. In our application example, code supports both a native execution on the soft-core processor and OS-based execution (µClinux). Selection is easily done through a define tag and adequate compilation scripts. Source code files were sorted in three categories: OS specific, native execution specific and common.

Finally we can conclude that soft-core FPGA processor based platform prove to fit well as an approach for rapid prototyping for several embedded systems providing a structured design process. This is particularly true for designs requiring flexibility and extensibility capabilities.

The SAND is a result of a research project (RETICOM), conducted by CETIC, supported by the European Union (ERDF) and the Walloon Region (DGTRE) under the terms defined in Convention n° n°EP1A1203000076F – 130004.

10. References

[1] Stephen Haag, Maeve Cummings, Donald J. McCubbrey, Alain Pinsonneault, Richard Donovan *Management Information Systems For The Information Age*, Chapter 6: system development, McGraw-Hill Ryerson, ISBN 0-07-095569-7, 2006.

[2] Web definitions from:
http://www.cbu.edu/~lschmitt/I351/glossary.htm
http://en.wikipedia.org/wiki/Prototyping
http://uis.georgetown.edu/departments/eets/dw/GLOSSARY0816.html

[3] Sommerville, I., *Software Engineering*, 8th Edition, Addison-Wesley, ISBN 0321313798, 2006.

[4] Siewiorek, D.P, Smailagic, A., Lee, J.C.Y., Tabatabai,Computers, A.R.A., "Interdisciplinary Concurrent Design Methodology as Applied to the Navigator Wearable Computer System", *Journal of Computer and Software Engineering*, Vol. 2, No. 3, pp. 259-292, Mellon University, July 1994.

[5] Asim Smailagic, Daniel P. Siewiorek, Richard Martin, John Stivoric, "Very Rapid Prototyping of Wearable Computers: A Case Study of Custom versus Off-the-Shelf Design Methodologies", *Journal on Design Automation for Embedded Systems*, Kluwer Academic Publishers, Vol 3., No. 1, 1998.

[6] Jeffrey M. Thompson , Mats P. E. Heimdahl , Steven P. Miller, "Specification-based prototyping for embedded systems", *Proceedings of the 7th European software engineering conference held jointly with the 7th ACM SIGSOFT international symposium on Foundations of software engineering*, p.163-179, September 1999.

[7] D. Reilly, A. Taleb Bendiab, "A Rapid Prototyping Approach For Design Of Extensible In-Vehicle Telematics Systems", *Journal of Integrated Design & Process Science*, Volume 6, Issue 2 (April 2002), p.91-106, 2002.

[8] M. Delehaye, D. Hubaux, L. Guedria, J.-D. Legat, T. Delvaulx, B. Goffard, "Smart adaptable network device for fleet management and driver coaching", *Proceeding of the 7th ITS and Telecommunication conference, ITST*, 2007.

[9] Abrahamsson, P., Warsta, J., Siponen, M.T., & Ronkainen, J. "New Directions on Agile Methods: A Comparative Analysis", *ICSE*, 2003.

[10] Pikkarainen, M. & Salo, O. "A Practical Approach for Deploying Agile Methods", *The 7th International Conference on eXtreme Programming and Agile Processes in Software Engineering*, 2006.