**Communication-Centric Design for FMC based I/O System**

Mokhtar Bouain  
University of Valenciennes, LAMIH, France  
mokhtar.bouain@inria.fr

Venkatasubramanian Viswanathan  
Nolam Embedded Systems, France  
vviswanathan@nolam.com

Rabie Ben Atitallah  
University of Valenciennes, LAMIH, France  
rabie.benatitallah@univ-valenciennes.fr

Jean-Luc Dekeyser  
University of Lille1, LIFL, France  
jean-luc.dekeyser@lifl.fr

---

**Abstract**—This paper presents a communication-centric reconfigurable design for FPGA Mezzanine Card (FMC) based I/O system dedicated to several industrial application domains. The introduction of FMC I/O standard has given a new purpose for FPGAs to be used as a communication platform. In fact, FPGAs can be used for more than just computational purpose in order to improve the system performance. Taking into account the features offered by FPGAs and FMCs, such as runtime reconfiguration and modularity, we have redefined the role of these devices to be used as a generic communication-centric platform. A new modular, runtime reconfigurable, Intellectual Property (IP)-based communication system for industrial applications has been designed. The efficiency and the performances of our platform are illustrated through a real industrial use-cases designed using a computationally intensive application and several I/O bus standards.

I. INTRODUCTION

Embedded reconfigurable systems are being used today in avionic, telecommunication and automotive industries. They are responsible for control, collision avoidance, driver assistance, target tracking, and navigation, amongst other functions. These systems demand high computation rates while carrying-out intensive signal processing applications. Furthermore, these embedded systems often operate in uncertain environments. They should adapt their functionality to provide reliability, fault tolerance, deterministic timing guarantees, and power efficiency. FPGAs have demonstrated superior performance, flexibility, and adaptability to the application constraints. Above all, FPGAs allow the integration of a complete system (processor, peripherals, accelerators, etc.) on a single chip. Such features make FPGAs a privileged target platform to implement intensive signal processing applications. However, industrial embedded systems also have a lot of complex communication sub-systems that rely on different I/O bus standards to exchange data between the on-board equipments (the navigation system, cruise control, etc.). Traditionally, computation and communication are performed using different individual Printed Circuit Boards (PCBs). This not only makes the system more complex and tedious with more Non-Recurring Engineering (NRE) cost, but also consumes more power. It is quite efficient to carry out both computation and communication on the same platform to simplify the system to a great extent. Our objective is to redefine the role of FPGAs for carrying out both communication and computation on the same chip, thereby reducing the cost, simplifying the system and hence making it a centric component in several application domains.

In current industrial practices, one of the biggest challenges of relying on different PCBs for different requirements, is the hardware obsolescence issue. Ever-changing application requirements demand the customization of the I/O bus interfaces. Changing the hardware meant redesigning the entire board, with a lot of NRE cost and significant time-to-market. VMEbus International Trade Association group (VITA) FMC standard [9] solves the I/O obsolescence issue partially, with a single 400-pin connector with a potential overall bandwidth of 40 Gb/s. This essentially means that the I/O bus interface of a PCB is designed separately as a module and interfaced with the board using the FMC connector. Thus, every time an I/O bus interface needs a change, just the module changes, thereby avoiding a complete redesign.

Used in conjunction with FMCs, FPGAs can serve as a very attractive communication platform, since they can be reprogrammed with the necessary protocol. However, the problem doesn’t end there. The communication sub-systems make up a significantly complex and resource consuming part and are still managed using different I/O boards each implementing different protocols. Therefore, these systems have to be managed more efficiently to minimize the hardware consumption and to maximize the functionality.

In this work we present a new modular, runtime reconfigurable, Intellectual Property (IP)-based approach for industrial communication systems. This is mainly based on two principles namely, modular I/O bus interface and runtime reconfigurable hardware. In addition to the computation part, we manage all the required communication protocols as dynamic IPs. This means that, when the communication requirement of the system changes, the necessary IP core is installed and executed on demand. Hence our approach presents an obsolescent-proof, modular solution for a computation and communication-centric reconfigurable system.

The rest of the paper is organized as follows. Section II introduces the state-of-the-art and industrial practices used in different domains. Section III describes our generic runtime reconfigurable communication system and its key features. In Section IV, we describe several industrial use-cases that
make profit from the FMC based I/O system with different communication cores. This is finally followed by evaluating the efficiency and performance metrics of our system in Section V. Then Section VI finally draws up some final words on the advantages of such approach and possible future works.

II. RELATED WORKS

In this section, we provide an overview of the already existing approaches in several industries, specifically with respect to communication systems. We also provide a historical perspective on the evolution of such systems and its current state that serves as the motivation for our work in this context. Several avionic communication architectures were based on proprietary all-in-one communication management units. It lacked flexibility and were not able to adapt easily to new technologies and new communication protocols. The ever-changing application requirements often rendered these systems obsolete. Furthermore, existing solutions typically employ static sub-systems that ensure reliability through hardware redundancy. Such methods are too expensive in terms of hardware in the current generation of industrial applications.

In recent years, the feasibility of using reconfigurable hardware is being explored in the field of aerospace and defense applications [10], [7], [1], [8]. However, using FPGAs in such applications has its own challenges since time, space, power consumption, reliability and data integrity are highly crucial factors. Some of these challenges are being addressed at the technology level, and some of them at the architectural level. One of the main challenges of using reconfigurable hardware specifically in space missions is that, it has to be radiation and fault tolerant. Single Event Upsets (SEUs) are induced by radiation. The environment where the avionic systems operate has unfavorable effects in these devices. Therefore it is important to provide a fault-tolerant computing platform for such applications which are prone to radiation effects. The works done by [8] and [5] address and mitigate the effects of SEUs on FPGAs and provide a reliable computing platform. Extensive work has been done in developing hardware/software co-design for an avionic communication system based on ARINC communication protocol [2]. Another related work also proposes the configuration and deployment of infrastructure and related procedures of a distributed avionic communication system in a FPGA [5]. Such works serves as the foundation for the usage of FPGAs in aerospace applications.

FPGAs have been also proposed as communication support for wireless networking [6], not just for data transmission but also for control and status information. This work, provides a unified data interface that can support multiple networking technologies including Giga-bit Ethernet through the host CPU and also wireless networking through FPGAs with 2.5 GHz frequency band. It aims to provide a framework allows collecting and analyzing information of performance, temperature and power consumption of the system used. In automotive domain, FPGAs are also proposed as execution support for highly intensive signal processing application such as multiple target tracking and adaptive cruise control [3] [4]. Almost of the proposed solutions in this domain are customized design for particular application without overcoming the challenges of flexibility and NRE costs.

In this work, we rely on the FMC standard coupled with runtime reconfiguration feature of recent FPGAs in order to design a modular IP-based communication system. We then establish that such an approach eliminates the need of redundant sub-systems, minimizes resource utilization, eliminates NRE costs and maximizes performance.

III. A GENERIC RECONFIGURABLE COMMUNICATION SYSTEM USING A FMC BASED I/O

Traditional approaches in the industry involve the usage of different PCBs and Commercial-Off-The-Shelf (COTS) components for implementing different communication systems. However, such an approach requires a lot of NRE, time-to-market and system space. In order to address this challenge, we have developed an architecture that is modular, cost effective (monetary and resource wise), and obsolescent proof due to two main reasons. First, any hardware instance (IP) of a communication protocol can be installed and executed on demand, if need arises. Second, the I/O bus interface can be swapped when required (for instance, USB, UART, CAN Bus, MIL-STD-1553, AFDX etc.) using FMC modules. Thus a single architecture/chip can support any number of I/O bus standards.

The architecture of our modular runtime reconfigurable communication system is shown in Fig. 1. The setup consists of an embedded processor attached to a certain peripheral devices and the avionic, automotive or telecommunication I/O protocols via the Processor Local Bus (PLB). These I/O protocols communicate with the external sub-systems via the interface provided using a FMC module. In order to illustrate a working prototype of a runtime reconfigurable communication system, several I/O IP cores such as CAN Bus, ARINC 429, and MIL-STD-1553 were developed. Each IP core can be used in different scenarios depending on the application domain requirement. The industrial communication IP cores are instantiated in a dynamically reconfigurable region of the architecture. Therefore, depending on the scenario, the user can choose a communication IP core to be configured at runtime. This removes the need to have multiple/redundant systems, each for a different protocol. Moreover, when the IP core is reconfigured, the communication channels with the FMC are also reconfigured dynamically along with the protocol. In case the FMC module does not provide a corresponding interface for the communication core being reconfigured (which can be detected using the I²C EEPROM in the FMC module), it can be swapped with another appropriate FMC module. Thus eliminating the need to redesign the entire board based on a new I/O interface requirement.

In order to compare and contrast the efficiency of our runtime reconfigurable communication system, with the traditional redundant systems, we have two different types of architectures. First, only one communication IP core is present at a time and can be swapped on-demand. Second, all the IP cores are present at the same time. An example of image encoding (JPEG) application was chosen to evaluate the architecture and our IP cores. The software interface is provided by a Xilinx Microblaze processor with some peripherals attached using a Processor Local Bus (PLB), as shown in Fig. 1. A Xilinx ICAP controller is used to perform partial reconfiguration. We have chosen PLB interface instead of Fast Simplex Link.
(FSL) because, the IP cores are serial communication protocols and require the configuration of only few registers for their operation. However, on the other hand, intensive signal processing applications (such as JPEG Encoder in Fig. 1) require a high-bandwidth communication link with the processor, for processing and data transfers. Therefore, they are connected to the processor using Fast Simplex Link (FSL).

IV. INDUSTRIAL USE-CASES

In this section, we will detail the usefulness of the proposed generic communication system in different industrial domains and thus emphasising the usage of FPGA as communication-centric platforms.

A. Automotive Domain

Today, transport policies are putting more and more emphasis on the development of sustainable transportation systems in order to solve the problems of urban pollution and seamless and safe mobility of people. A new trend for transportation sector proposes autonomous vehicles as an innovative solution for transporting small groups of people in common. So that upon the demand, we can move in environments such as pedestrian zones, large private industrial sites, airports, theme parks, university campuses or even hospitals. However, this trend is considered as a scientific and technological challenge in order to be safe and effective in terms of technology, cost and autonomy. Due to the necessity of autonomy driving, electric vehicles use many sensors to provide the best control which allows the safety driving.

The technology within the commercialized vehicles uses exteroceptive sensors (lasers, cameras, etc.) for localization and obstacle detection. Currently, the using of Light Detection and Ranging (LiDAR) induced a high cost, as well as the used cameras have too low resolution so that the detection of obstacles is very limited, and characterizing and monitoring of dynamic obstacles are almost non-existent. On the algorithmic level, the existing solution for detecting obstacles by stereoscopic vision works on grayscale images of low resolution implying a low accuracy in the longitudinal and lateral localization of obstacles. The current solution is justified by the real-time constraints. Today, the data processing takes place on a dedicated industrial PCs ensuring the acquisition, the synchronization and the processing of the received data. However, industrial PCs consume hugely the electrical energy, which is an important constraint for electric vehicles that require the minimization of power consumption by every element of its system. To overcome this problem, we can implement the algorithms used by the cameras in an embedded system to reduce the consumption of electrical energy.

Therefore, our application is included in this context. The platform used is coupling the Xilinx Virtex6-based ML605 board and a stereoscopic camera module provided by Advansee corporation\(^1\). This camera allows to ensure the acquisition of synchronized image pairs. Embedded processing of acquired images by each camera is carried out in the FPGA and the transmission of images is performed via the Ethernet network. Because this kind of camera uses the stereoscopic vision which provides two camera, so there are two video outputs as shown in Fig. 2. But the ML605 board provides only one input Ethernet port so we will use the 1000 Base-Ethernet FMC interface allowing to convert the data received by two Ethernet cables to FMC communication protocol.

On the other side, to control the car, the system uses the Controller Area Network (CAN) which is a serial bus system widespread in many field industries, including automotive. So according to the video input from cameras, the FPGA processes the data and makes the decision to calculate the deviation of the trajectory. After this, it sends the control commands to the actuators through the FMC interface respecting the CAN protocol. To do so, we have implemented the corresponding hardware I/O IP core.

1) The CAN Bus: The CAN controller designed, implements the Data Link Layer as defined in the document “BOSCH CAN Specification 2.0”\(^2\). It implements a serial communication which efficiently supports distributed real-time control with a very high level of security. The design has two communication channels. The CAN bus protocol supports up to 16 channels, and has a maximum bandwidth of up to 1 Mbps. The transmission can be programmed to a random frequency using the configuration registers. The architecture

\(^1\)http://www.advantec.com/
\(^2\)http://esd.cs.ucr.edu/webres/can20.pdf
of CAN controller is shown in Fig. 3.

CAN works in the principle of automatic arbitration-free transmission. A CAN message that is transmitted with highest priority will succeed, and the node transmitting with the lower priority message will sense this wait for transmission. This is achieved by using the notion of dominant bits and recessive bits where dominant is a logical 0 and recessive is a logical 1. Therefore, in a physical implementation of the bus, if one node transmits a dominant bit and another node transmits a recessive bit then the dominant bit takes priority over the recessive bit. This is implemented as a logical AND between the bits.

B. Avionic Domain

The second use-case scenario for our architecture is a part of an Unmanned Aerial Vehicle (UAV) avionic system. The task of the system is to travel between different terrains; to take pictures, to encode and either store them internally, or to transmit them to a remote system. Depending on the scenario, an appropriate communication protocol (ARINC429/MIL-STD-1553) has to be selected and configured during runtime. For instance, when a secure transmission is needed, the MIL-STD-1553 is chosen.

The architecture was implemented on a Xilinx ML605 board using a Xilinx Microblaze embedded processor. The board provides two FMC slots. One with a High Pin Count (HPC) and the other with a Low Pin Count (LPC). It can be used to host two different FMC cards simultaneously as shown in Fig. 4. The images were captured using a FMC LPC camera module. The captured images were stored in the Compact Flash (CF) which were later used for encoding. For testing purposes, we have chosen to monitor individual images instead of an entire video.

The Xilinx EDK and ISE tools were used to generate the bitstream. Initially the partial bitstreams are stored in the Compact Flash memory and are read when requested by the application. The operational frequency of the processor, buses and the peripherals is 100 MHz. A C program is used to initialize and to interact with Microblaze and thus the underlying hardware in order to configure the registers and initiate data transfers. A communication protocol (ARINC429/MIL-STD-1553) runs in parallel with a JPEG encoder. The captured image is encoded, and is transmitted to an avionic sub-system using an appropriate communication protocol selected during runtime via a FMC interface.

Ideally, the data has to be transmitted to external sub-systems. However, for testing purposes, we have done an external FMC loop-back to verify if the transmission is correct as shown in Figure 4. Once the encoding starts and while the result is being written into the buffer, the Microblaze reads from the buffer simultaneously and sends the frames to the configured IP core for transmission.

All the above mentioned characteristics give our system some key advantages. First, it makes it very compact since only one avionic protocol is present at a time. Next, using runtime reconfiguration along with the industrial state-of-the-art FMC interface for transmission, makes the system modular and obsolescence proof. Finally, doing computation and communication at the same time, exploits the hardware parallelism, unlike the industrial state-of-the-art solutions which uses a system only for a specific purpose (i.e. video board, communication board, processing board etc.). In the following, we describe briefly the ARINC429 and MIL-STD-1553 protocols.

1) ARINC429 and MIL-STD-1553 BUS: The ARINC429 is an application-specific technical standard for the avionic data bus used on most higher-end commercial and transport aircrafts. It defines electrical characteristics, word structures and protocol necessary to establish an avionic bus communication. For ARINC429, messages are transmitted at a bit rate of either 12.5 or 100 Kbps to other sub-systems. The design supports up to 16 Transmit and 16 Receive channels, although the protocol permits up to 20 channels. The architecture of ARINC429 is shown in Fig. 5 (left). The standard uses a self-clocking, self-synchronizing data bus protocol where transmit (Tx) and receive (Rx) are on separate ports. Data words are 32 bits in length and most messages consist of a single data word. The physical connection wires are twisted pairs that
carry balanced differential signaling. The transmitter transmits either 32-bit data or the NULL words. A single wire pair is limited to one transmitter and up to a maximum of 20 receivers. The protocol allows for self-clocking at the receiver end, thus eliminating the need to transmit clock information. ARINC429 is an alternative to MIL-STD-1553 standard.

The MIL-STD-1553 is originally a serial military standard protocol that defines the mechanical, electrical, and functional characteristics of a serial data bus. It is now also being used in spacecraft On-Board Data Handling (OBDH) subsystems, both military and civil. The architecture of the bus system consists of a Bus Controller (BC) controlling multiple Remote Terminals (RT) all connected together by a data bus providing a single data path between the bus controller and all the associated remote terminals. The RT is used to interface with other user-defined subsystems. There can also be one or more Bus Monitors (BM); however, they are not allowed to do any data transfers, and are only used for recording the data for analysis. The protocol also supports several data buses to provide multiple redundant data paths up to a maximum of 4. The protocol follows very strict timing constraints and requirements and provides a maximum bandwidth of 1 Mbps. We have developed our own IP core according to the MIL-STD-1553 specification and used it to evaluate our system. The architecture of the MIL-STD-1553 is given in Fig. 5 (right).

C. Telecommunication Domain

As a third application domain, we aim to design a communicating platform dedicated to the intelligent video processing. It can perform a large number of applications in various fields such as tracking a moving target between different locations or detection of groups and their informations: time, location, number of individuals, etc. In this work, our platform will be fixed for tracking a person or an object moving between different locations. For example, in a public place, when an individual has a suspicious behavior, several indices are used to trigger an alarm. In addition, users can choose one person or object to be tracked during its movement between different areas.

The reconfigurable program part can be coupled with high-speed RF transmission modules through the FMC interface by ensuring the security of information, so we will couple the FPGA platform with an RF I/O based FMC cards. The distributed architecture of the system is illustrated in Fig 6. It consists of a number of nodes installed in a restrained locations.

The proposed system has the advantage of autonomy, it integrates a number of self communicating nodes and then it can achieve its mission for long periods without human intervention. Otherwise, each node is composed of a camera, an FMC150 RF interface, and the Xilinx ZC706 board. The communication between two nodes is done through a wireless network with a frequency band of 60 GHz and a throughput of 1GB/s.

About the camera used, it is the VITA-2000 camera. This camera delivers HD frame format with a resolution of
rely on digital buses in our use-cases. Second, implementation related to the automotive and avionic applications as they require a specific number of FPGA I/O pins to communicate with the external sub-systems via FMC. The number of pins required is dependent on the protocol selected and configured during runtime. Initially, the partial bitstreams are stored in the Compact Flash memory and are read when requested by the application. A C program is used to initialize and to interact with Microblaze or the Cortex A9 and thus the underlying hardware in order to configure the registers and initiate data transfers. A communication protocol runs in parallel with a JPEG encoder. The captured image is encoded, and is transmitted using an appropriate communication protocol selected during runtime via a FMC interface.

A. automotive and avionic applications

1) FPGA resource utilization: Table I summarizes the area utilization of each protocol with different configurations. While considering only one IP core active at a time (i.e. a design with only an ARINC429 16 channels or a CAN Bus 16 channels), the consumed area is about 37% of the logic blocks in the Virtex6 FPGA of the ML605 board (with respect to Flip Flop utilization). However, when 3 IP cores are implemented at the same time (in this scenario ARINC429 16 channels, CAN Bus 16 channels, and MIL-STD-1553), it consumes over 70% of the resources on the Virtex6 FPGA which is about 50% excess comparing to one IP core active at a time. This is a waste of hardware resources considering the fact that the FPGA can host also intensive signal processing applications. Using Partial Reconfiguration (PR) is very relevant because it clearly eliminates the need for having multiple I/O systems for the lack of hardware resources.

2) Transmission times: Figure 8 shows the performance of our IP cores in terms of time taken for transmitting different number of frames. The system performance has been evaluated up to 1000 frames. The time measurements were done in the software using a Microblaze timer. The transmission and reception are synchronized according to the protocols’ internal clocks in pre-programmed frequencies. However, the time measured, also takes into consideration the overhead of configuring the registers, the overhead caused due to the transfer of status words and the idle time between each transaction. From the graph shown in Fig. 8, it is seen that MIL-STD-1553 is the fastest in transmitting the frames. Although MIL-STD-1553 and CAN Bus have the same maximum bandwidth, the fact that MIL-STD-1553 is able to pack more data into a single message and to operate with minimal status feedback, gives it an extra edge in transmitting efficiently.

3) Number of I/O pins and channels: Each protocol requires a specific number of FPGA I/O pins to communicate to the external sub-systems via FMC. The number of pins

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Slices</th>
<th>FFs</th>
<th>LUTs</th>
<th>BRAMs</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARINC429-2 channel</td>
<td>1,912</td>
<td>3,198</td>
<td>7,844</td>
<td>2</td>
</tr>
<tr>
<td>ARINC429-16 channel</td>
<td>47,920</td>
<td>44,267</td>
<td>59,400</td>
<td>2</td>
</tr>
<tr>
<td>CAN Bus-2 channel</td>
<td>689</td>
<td>1,016</td>
<td>2,754</td>
<td>0</td>
</tr>
<tr>
<td>CAN Bus-16 channel</td>
<td>10,176</td>
<td>15,437</td>
<td>41,096</td>
<td>0</td>
</tr>
<tr>
<td>MIL-STD-1553- Dual Redundant</td>
<td>1,232</td>
<td>3,432</td>
<td>8,453</td>
<td>16</td>
</tr>
</tbody>
</table>
required is determined by the number of communication channels in the design. Table II shows the number of FPGA I/O pins required for each protocol according to the number of I/O channels. However, some data in the table are not shown because, CAN bus standard does not support more than 16 channels and MIL-STD-1553 is never used beyond 4 redundant buses. As seen in Fig. 9, the number of pins linearly increases with respect to the increase in the number of channels. This parameter is important for the following reasons. It is quite clear that packing all protocols at the same time requires about 230 user I/O pins on the FPGA. However, the number of FPGA user I/O pins that can be allocated for these communication protocols is restricted by the number of FPGA ports and FMC slots. While a mid-size FPGA (as used by many avionic systems) may have anywhere around 300 to 500 pins, using all these pins for FMC I/O will not leave sufficient pins for the other FPGA peripheral devices (i.e. Ethernet, SFP, LCD, memory, back planes, front panels, etc.). On the other hand, a high-end FPGA which may have up to 1200 user I/O pins are usually chosen to pack dense computational logic, hence using them for multiple communication protocols wouldn’t be very cost effective.

4) Scalability: The scalability of the system is given in terms of number of transmission channels each core can accommodate, which is derived from the protocol specification. ARINC429 is capable of scaling up to a maximum of 20 channels and CAN up to a maximum of 16 channels. However, MIL-STD-1553 is only dual redundant. It is quite important to note that, scalability of a protocol does not increase the bandwidth of the system. The bandwidth of the bus is dictated by the standard itself. However, the number of channels only enables to the system to simply communicate with more number of sub-systems.

5) Reconfiguration latency and application profile: We measure the time taken to dynamically reconfigure the system with our communication protocols. All the timing are measured from the software part. Partial bitstreams are stored in the Compact Flash and read when requested. It is quite obvious that the bigger the size of the bitstream the longer the reconfiguration latency. Bitstreams of size less than 500 KB require less than a second to be reconfigured. Configuration stream of size 645 KB which is the size of our bitstream (the IP cores with 2 channels), requires roughly about 1.3 seconds while bitstreams larger than a 2 MB (corresponds to the IP core with 16 channels) have reconfiguration latency of few seconds. The read queue in the XPS_HWICAP controller buffers the configuration data before it is fed to the ICAP. However, we see that the throughput of the ICAP controller is less than the theoretical maximum because of the disk access overhead caused by the Compact Flash.

The reconfiguration latency can be completely hidden in many scenarios as shown in the profile of the application in Fig. 10. From this illustration, it is quite obvious that transmission time is quite negligible compared to execution time of the application (JPEG). It is also seen that the reconfiguration time become also negligible with significant processed data. As Partial Reconfiguration means the ability to dynamically modify blocks of logic by downloading partial bit files while the remaining logic continues to operate without interruption\(^3\), the reconfiguration phase can be anticipated during the application processing.

6) Power estimation: The total power consumption of the design depends on several attributes of the overall design. But it is safe to assume that bigger the design more the power consumption is, unless special power saving mechanisms such as clock gating is applied, among other factors. From Fig. 11, the bigger the design, the greater the overall power consumed. We are comparing the overall power consumption of each design with respect to the design with ARINC429 16 channels, CAN Bus 16 channels, MIL-STD-1553 and JPEG application at the same time. The maximum power savings are about 400 mW when only one IP is present at a time. Although this difference may not be huge, as mentioned earlier, other industrial protocols such as AFDX are far more complex and power-hungry with several Gigabit transceivers operating at the same time.

\(^3\)Xilinx: http://www.xilinx.com/tools/partial-reconfiguration.htm
B. Telecommunication application

The implementation of FMC 150 I/O consists in controlling the transmission and the reception of data between two ZC706 boards. It uses a module operating at 245 MHz for A/D and 122 MHz for D/A conversion.

Table III shows the used resources by the module FMC150 on the programmable logic of the Zynq-7000 SoC. According to this table, the implementation of FMC150 module consumes limited resources in terms of slices, slice registers and slice LUTs. However, area utilization is 16% for the RAMB36E1/FIFO36E1s since the I/O hardware module uses many FIFO memories needed for data storage in transmission and reception modes. Concerning the BUFG and BUFGCTRLs, which are global clock buffers, occupy 31% of the available resources because the I/O IP core uses many clock signals to control D/A and A/D converters. For the power consumption of a communicating node, the measured values are as follows: 200 mW for the ARM Cortex A9 processor, 60 mW for the Programmable Logic, and 190 mW for the I/O FMC connector.

![Fig. 10: Application profile](image1)

![Fig. 11: Total power consumption of each design](image2)

<table>
<thead>
<tr>
<th>Resources</th>
<th>Used</th>
<th>Available</th>
<th>Utilization (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slice Registers</td>
<td>6,391</td>
<td>437,200</td>
<td>1.4%</td>
</tr>
<tr>
<td>Number of Slice LUTs</td>
<td>4,992</td>
<td>218,600</td>
<td>2.3%</td>
</tr>
<tr>
<td>Number of occupied Slices</td>
<td>2,594</td>
<td>54,650</td>
<td>4.8%</td>
</tr>
<tr>
<td>Number of RAMB36E1/FIFO36E1s</td>
<td>92</td>
<td>545</td>
<td>16%</td>
</tr>
<tr>
<td>Number of BUFG/BUFGCTRLs</td>
<td>10</td>
<td>32</td>
<td>31%</td>
</tr>
</tbody>
</table>

TABLE III: Area utilization of the I/O IP core

VI. Conclusion

In this paper we have presented a new FPGA-centric communication platform for several industrial fields. Combining Partial Reconfiguration feature of FPGAs with FMC as a communication-centric component in such systems eliminates obsolescence issues at the architecture level. This also means that it is no longer required to use a system for a specific purpose, but use one system for several purposes with guaranteed uninterrupted operations, thereby removing huge NRE costs and time to market. However, the challenges do not stop here. The FPGA families vary in their sizes and cost. Applications requirements may change depending on the cost, size of the device, etc. Changing the FPGA device for each application will re-invent the wheel of problems described above. Our future work will not only address how to solve such obsolescence issues at the hardware level, but also improve the runtime reconfiguration speeds.

REFERENCES


