Integrating DSP and FPGA evaluation modules for building high performance computing platforms

Ilan Sousa
Federal University of Pará
Rua Augusto Correa, nº 1
Belém, Pará, Brazil
55 91 32017674
ilan@ufpa.br

Aldebaro Klautau
Federal University of Pará
Rua Augusto Correa, nº 1
Belém, Pará, Brazil
55 91 32017674
aldebaro@ufpa.br

ABSTRACT

The so-called Evaluation Modules (EVMs) are largely present in the market and are often developed to serve as reference design to the main integrated circuit (IC) of the EVM. For example, it is common to find EVMs of microcontrollers, Field Programmable Gate Array (FPGAs), analog ICs and others. This work encourages building high performance computing systems by joining high end Digital Signal Processors (DSP) and FPGA EVMs, and proposes two schemes for division of the tasks in a system containing DSP and FPGA, which can take advantage of both to speed up the processing. This work also takes advantage of this variety of EVMs and encourages their usage to rapid design and build prototypes, which allows the designer to choose components which best fit the target application, with previous knowledge or even with the budget of the project.

Categories and Subject Descriptors

B.8.2 [Performance and reliability]: Performance Analysis and Design Aids; B.5.1 [Design] Data-path design.

General Terms

Algorithms, Performance, Design, Verification.

Keywords

Hardware and Software codesign, programmable logic, digital signal processing, computer architecture.

1. INTRODUCTION

There are applications in many areas that require high computational power which can be implemented either on dedicated hardware or in general purpose processor. A good example is digital signal processing, because it often must process an amount of data in a determined time before the next data set arrives.

Another characteristic of these systems is that most of the time they are not conceived for a single task. For example, if a system has a microcontroller as the only device capable of performing computation, it may be responsible for communicating with other components of the system, storing data, run an operating system and so on. All these tasks run concurrently with the digital signal processing, which makes the completion of the latter harder.

In order to solve the problem of multiple tasks running in a single processing unit, many approaches are widely known on the literature, one of them is to exploit the usage of both CPU and dedicated hardware in the same system to accomplish them. The idea is to take advantage of both and divide the processing to speed up the result generation. For example, in [3] it is proposed an approach where two versions of the same algorithm are made: one for a processor and the other for dedicated logic, letting a compiler to use a standard modelling approach to choose which implementation is better for a certain application.

The usage of both DSP and FPGA has been early adopted by many system designers, which have developed methodologies as described in [6, 9]. As shown in [4], joint hardware and software development has been made since the beginning of the digital logic technology, but in a level that they are integrated in the same IC, which has a long development cycle and is expensive. And nowadays, with the advance of IC technology there is a very popular a type of IC which is called System-on-chip (SoC), which can be composed by a processor and several peripherals or coprocessors within the same IC to help the processing.

In this work we propose to take advantage of the high computational power of both DSP and FPGA to make up a high performance computing system, using EVMs and high speed serial communication. The DSP and FPGA together can be used to make up a system similar to SoCs, with a processor, coprocessors and a high speed serial communication to decrease the latency in the communication and the pin count required. The EVMs turn this system easier and faster to prototype than a traditional SoC, which is good for rapid design or even low quantity products.

There are many EVMs which can be used in a system like this, because it is a common practice of IC manufacturers to sell EVMs of their products as a reference design to application designers. Consequently, this strategy provides an abundance of available options for choice, allowing the designer to choose its preferred platform or a more suitable to a certain application. Moreover, these evaluation platforms can be sometimes cheaper than the IC itself and they have the advantage of skipping a printed circuit board (PCB) design and development phase in a project.

This work shows an example system composed of two high end EVMs from Xilinx and Texas Instruments; for the dedicated hardware part of the system it is used a Xilinx Virtex-6 FPGA ML605 Evaluation Kit [8] and for the processor it is used a Texas Instruments TMS320C6670 Evaluation Module [1], which exchange information through a high speed Serial RapidIO link.

Both EVMs can be examples where the price is almost the same of the IC itself, as shown in Table 1, which summarizes and compares the prices of the main IC present in the board and the whole EVM, where it is possible to realize that the IC and its EVM have almost the same price. This way, in an application which do not have strict requirement of power or size, EVMs can be easily used instead of designing a specific board. Another advantage of using evaluation boards is that they are largely used systems, often with several threads in discussion forums which is a good option for support, besides the manufacturer support itself and others examples largely available on the internet.

Table 1. Single IC and EVM prices comparison. All prices are in US dollar.

	Single IC	EVM
TMS320C6670	270	400
ML605	1400	1800

This work is organized as follows. Section 2 proposes two schemes for connecting and dividing the processing between the EVMs. Section 3 presents and quickly discuss about the two EVMs used as examples in this work. Section 4 presents an application using the example system of this paper; and Section 5 presents the conclusions.

2. Task Division

This section proposes two options for the tasks division between the processing units of the system, which depends on how the designer is going to use the system or on the application requirements. The division will also impact on the type of messages to be exchanged between them.

The task division between FPGA and DSP can be made by taking advantage of strong points of each one. For example, if an operation can be executed faster if more concurrency or more efficient bit-level processing can be exploited, then this operation is more suitable to be performed by a FPGA. On the other hand, if an operation can take advantage of built-in features of the DSP, such as multiply-and-accumulate (MAC) structures or more RAM memory, the DSP is a good choice for executing that operation.

2.1 Data path

The first division composes a chain or data path with the two EMVs, where data is acquired or generated by one of the EVMs, some processing might be made and then the resulting data is forwarded to the next component of the system. In this scheme, the division of the tasks can also take advantage of characteristics of one EVM for performing an algorithm, but the division depends mainly on what stage the algorithm needs to be performed.

For example, FPGAs due to its reconfigurable nature have more general purpose pins (GPIOs) than the DSP, which makes the FPGAs more suitable for interface with analog-to-digital (ADC) and digital-to-analog (DAC) converters. Then, in this example, the FPGA receives the ADC data, and makes some processing, for example to classify if the received samples must be sent to the DSP, which saves the communication link bandwidth and DSP processing. This example is closely related to communication systems, where an ADC is continuously sampling, but only in a determined time the received samples make up a signal from the communication partner which really contains information. In order to classify the input data, the FPGA can use techniques such as cross-correlation and the one described in [7]. After the classification, the FPGA sends to the DSP only samples containing information. Clearly, it saves DSP processing time because looking for a known signal normally involves large amount of data and CPU time, and the FPGA is capable of making classification by the time each single sample arrives.

The lines above explain the receiving chain of the system, whereas the transmitting chain uses data generated or acquired by the DSP. Commercial DSPs normally have means to interface with other systems, this way, for this example the DSP receives data from an *Ethernet* interface and modulates the data using some digital modulation, and sends the resulting data to the FPGAs, which in turn makes some processing before sending to the DAC. Figure 1 illustrates the system example given above.

Another example uses the FPGA as an intermediate point between the DSP and a peripheral, because it is common to be included in the FPGA EVMs audio and video interfaces, which can be used to extend the DSP set of peripherals and also use the data path scheme to save DSP processing time or even to complement the DSP set of peripherals with one that is not present in its EVM. In this case, the ADC and DAC of Figure 1 are the peripheral interface, or the converters present in the peripheral.

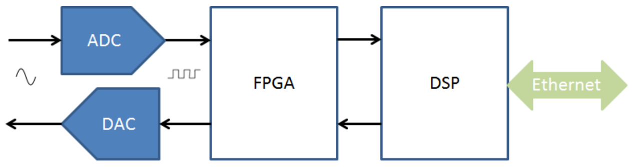

Figure 1. Example of FPGA and DSP composing a data path system.

2.2 Hardware accelerator

The second division scheme proposed in this work uses the FPGA as an extension to the DSP peripherals or coprocessors. In this scheme the FPGA can be programmed to contain implementations of several specific algorithms and the DSP acts as a client, that sends an amount of data and after the operation is completed, the FPGA sends the results to the DSP.

The algorithms to be implemented in the FPGA can be one with high computational requirement and high potential of parallelization, such as correlation or Fast Fourier Transform (FFT). Moreover, the FPGA vendors provide a large variety of Intellectual Property (IP) cores, which are freely available to the user or available under some software or tool license, which can be included in the FPGA to be available as a coprocessor to the DSP. Examples of IP cores commonly found are FFT, floating point unity, filters, Up/Down converters, etc.

This way the designer is only responsible for providing an infrastructure to include several coprocessors in the FPGA, in case all the required coprocessors are implemented as IP cores. Figure 2 shows how FPGA provides the IPs interface to the DSP, where the FPGA code is responsible for receiving an amount of data with an identifier to which coprocessor to be used, and by the end of processing, responding to the DSP with the resulting data.

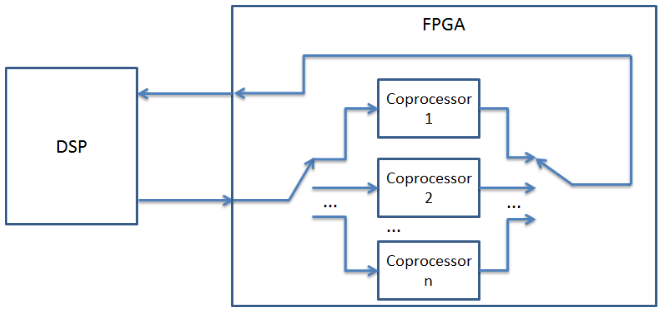

Figure 2. Example of the FPGA as coprocessor system to the DSP.

It is common in some SoCs to have processor and several coprocessors in the same IC. Thus, case the chosen DSP EVM is this kind of system, this scheme can extend the DSP set of available coprocessors, with other operations implemented in the FPGA. However it is not restricted to extend with different operations, because if a coprocessor within the DSP becomes busy during its execution, the DSP can use another coprocessor within the FPGA to perform the same operation, this way the DSP can send two requests in parallel.

3. Example system

This section presents with more details the example system mentioned before. Basically this system is composed of two high end EVMs: the TMDSEVM6670 and ML605 from Texas Instruments and Xilinx, respectively. The former is the DSP and the latter is the FPGA EVM. These EVMs exchange information through a Serial RapidIO link.

3.1 DSP

The DSP platform chosen to this example is a Texas Instruments multicore fixed and floating point SoC, the TMS320C6670, which has several accelerators available for common physical and medium access layer algorithms, of which the FFT Coprocessor (FFTC), the Network Coprocessor (NETCP) and the Serial RapidIO (SRIO) are examples.

Besides having a rich set of coprocessors, the TMS320C6670 SoC also is a quad-core system, where all cores have simultaneous access to all coprocessor and peripherals. This way, it is easy to happen a situation where one coprocessor being used by a core when another core try to use it, then to have the same operation available in the FPGA can avoid the second core to be blocked waiting for the coprocessor to be free.

3.2 FPGA

The FPGA board used in this example is the Xilinx Virtex 6 FPGA ML605 Evaluation Kit, which is composed of a Virtex 6 LXT FPGA and several peripherals. The FPGA, due to its reconfigurable nature, can be programmed to perform a broad variety of hardware functionalities, such as controllers for the peripherals present in the board or any arbitrary hardware. Moreover, Xilinx also provides IP cores which perform several different tasks; they are usually available as open or encrypted Hardware Description Language (HDL) code.

The IP cores are used to speed up the development, since they are extensively tested by Xilinx and they cover a large range of algorithms and functionalities. IP cores can be used for standardized functionalities, and the main example is the Xilinx Serial RapidIO core, which implements the specification version 2.2 of the protocol, that is used for exchanging data with the DSP in this example. The in-house developed HDL code performs application-specific tasks, such as data formatting and signaling which cannot be found in IP cores, because of their specific functionality.

3.3 RapidIO

The RapidIO is a packet-switched, high speed serial communication standard, which is divided in parallel and serial parts. This example uses the Serial RapidIO, because it is the protocol that achieves the highest data rates among the supported between the two EVMs. Moreover, it provides connectivity between several components by using a switch or between two components directly, this second option is used in this example.

3.4 Analog interface

For the analog interface of this example system an Analog Devices AD-FMCOMMS2 EVM [2] is used, which is a host board for the high integrated radio frequency (RF) transceiver AD9361 IC. Its main features are the capability of operating in a wide range of frequencies and the implementation of a complete RF interface in a single IC. It is composed by power amplifiers to drive the antennas, low noise amplifiers to receive signal, up and down converters, oscillators, etc.

4. Example application

This section presents an example application of a generic modem using the system of Section 3 and the scheme of data path presented in Section 2. The technique for modulation is Orthogonal Frequency-division Multiplexing (OFDM), which is a popular technique used in communication systems such as *xDSL*, *WiFi* and others. It has a relatively high computational requirement, because it needs a FFT operation and all the processing normally needs to be done within a strict timing. A good explanation about OFDM principles can be found in [5].

Basically, this system is composed by the two digital EVMs presented in previous sections connected by a Serial RapidIO link, and the AD-FMCOMMS2 EVM for performing up and down conversion, filtering and sampling.

In this example application there are two systems composed by the EVMs shown in this work, which exchange information through a wireless link using OFDM. They use the same frequency band for transmitting and receiving data. Thus, a scheme to avoid both modems sending at the same time is needed. This is done by implementing an arbitration of the channel, where a modem needs to wait its turn before sending data, which is called in the literature Time Division Duplexing (TDD). Figure 3 illustrates this arbitration using a state machine, where each state represents the operation being performed by the modem. In *Tx*, the modem is transmitting, in *Wait 1* it is waiting for its communication partner to send data, in *Rx* the communication partner has already started sending data and the modem is saving the samples and finally in *Wait 2* the modem simply is waiting before sending again, which corresponds to the guard interval used in commercial technologies, such as *WiFi*.

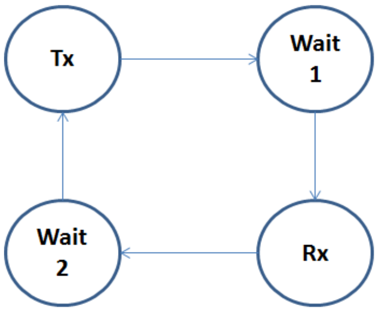

Figure 3. TDD arbitration used in the example.

The arbitration presented before works well when the modems have some means to keep their synchronization which is not an easy task. Then, to keep things easy to implement, it is more suitable to send a preamble before sending any data signal. Figure 4 illustrates the signal transmitted in state *Tx*, where before sending the data signal the modem sends a preamble to inform the communication partner about the beginning of valid data signals.

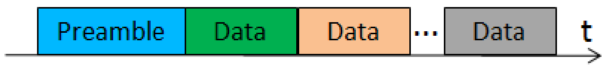

Figure 4. Structure of the signals of the example.

The preamble is a signal known by both the transmitter and the receiver, and the receiver in state *Wait 1* is actually looking for it, and once the receiver tracks the preamble it knows that the next samples are valid and can be saved.

The preamble can be anything even noise, the only requirement in this case is that the receiver knows what is going to be sent, and then the detection can be made by performing cross-correlation between the received signal and the reference signal in the receiver. However cross-correlation is a computational intensive operation which uses a long time of the CPU in case it is implemented in the DSP or many resources in case it is implemented in FPGA. There are many related work in the literature aiming to overcome this problem, such as the one described in [7], which generates a signal with two identical halves in time domain. This way the receiver does not need to store the preamble and it just looks for a signal with two identical halves, which can be implemented in a simpler way then cross-correlation.

4.1 Processing division

A simple division is made because the DSP has a FFT coprocessor, thus this coprocessor is used to limit the area where each EVM works. This way, the DSP is responsible for making frequency domain processing whereas the FPGA performs time domain processing.

The transmitting operation starts when the DSP receives data from an *Ethernet* interface and performs generation of OFDM symbols, which is comprised of several steps and the last one is the inverse FFT (iFFT) to generate a time domain signal to be sent to the FPGA. In the transmit chain, the FPGA makes the cyclic prefix (CP) extension and sends the preamble before sending the symbol in a transmission opportunity described in Figure 3.

In the receive chain the FPGA is continuously looking for a preamble in state *Wait 1*, and once it finds, preamble and CP are removed and the samples are forwarded to the DSP, which makes the processing of the OFDM symbol to recover the *Ethernet* packet, which is coded within this symbol.

In this example application the DSP generates signals to be transmitted, by feeding an implementation of a modulator with packets coming from the *Ethernet* interface, and after the resulting signal is sent to the FPGA which responsible for making time domain processing and to arbitrate the data exchange according to Figure 3.

5. Conclusions

This work presented and encouraged the adoption of EVMs to rapid building prototypes, which has the main advantage of avoiding printed circuit board design and implementation, which normally requires long time for design and testing and is normally expensive due to the low quantity required in these kinds of products. Moreover it is also good as a resource for teaching in areas that involve computer architecture, telecommunications, algorithms and many others. It should be noted that the EVMs manufacturers normally provide several example designs and maintain discussion forums regarding their products, which are huge sources of information.

6. REFERENCES

[1] Advantech, TMX320C6670 Evaluation Module, accessed April 22, 2014, http://www.advantech.com/Support/TI-EVM/6670le of.aspx.

[2] Analog Devices, ”AD-FMCOMMS2-EBZ User Guide”, 2014 - accessed April 22, 2014, http://wiki.analog.com/resources/eval/user-guides/ad-fmcomms2-ebz.

[3] B. Grattan, G. Stitt, and F. Vahid, “Codesign-extended applications,” in Hardware/Software Codesign, 2002. CODES 2002. Proceedings of the Tenth International Symposium on, 2002, pp. 1–6.

[4] J. Teich, “Hardware/software codesign: The past, the present, and predicting the future,” Proceedings of the IEEE, vol. 100, no. Special Centennial Issue, pp. 1411–1430, May 2012.

[5] National Instruments, “OFDM and Multi-Channel Communication Systems,” 2013 - accessed April 2014, http://www.ni.com/white-paper/3740/en/.

[6] Sanjaya Kumar, James H. Aylor, Barry W. Johnson, and William A. Wulf, “A framework for hardware / software codesign.,” IEEE Computer, vol.26, no. 12, pp. 39–45, 1993.

[7] T.M. Schmidl and D.C. Cox, “Robust frequency and timing synchronization for ofdm,” Communications, IEEE Transactions on, vol. 45, no. 12, pp. 1613–1621, Dec 1997.

[8] Xilinx, UG534 - ML605 Hardware User Guide, 2012 (accessed April 22,2014), http://www.xilinx.com/support/documentation/boards and kits/ug534.pdf.

[9] Y. Takeuchi, K. Shibata, and H. Kunieda, “Codesign methodology on programmable hardware and software system,” in Circuits and Systems, 1994. APCCAS ’94., 1994 IEEE Asia-Pacific Conference on, Dec 1994, pp. 182–187.