www.ioirp.com

International Journal of Innovative Research in Technology, Science & Engineering (IJIRTSE)
ISSN: 2395-5619, Volume – 4, Issue – 4. April 2018

MULTI CORE ARCHITECTURE ON FPGA USING ALTERA NIOS II

ASHOKKUMAR K
Research Scholar
Department of ECS
Hindustan CAS, Coimbatore, TN, India
krjashokcbe@gmail.com

Dr GOWRISHANKAR P
Prof and Head
Department of ECS
Hindustan CAS, Coimbatore, TN, India
hicaselectronicshod@hindusthan.net

ABSTRACT

In the past, processor design trends were dominated byincreasingly complex feature sets, higher clock speeds, growingthermal envelopes and increasing power dissipation. Recently,clock speeds have tapered and thermal and power dissipationenvelopes have remained flat. However, the demand forincreasing performance continues which has fueled the move tointegrated multiple processor (multi-core) designs. This paperdiscusses this trend towards multi-core processor designs, thedesign challenges that accompany it and a view of the researchrequired to support it.This paper describes a FPGA-baseddesign methodology to implement a rapid prototype of parametric multicore systems. A study of the viability of making the SoCusing the NIOS II soft-processor core from Altera is also presented.

Keywords : FPGA, SOC, NIOS II, Silicon, EDA

I. INTRODUCTION

In April 1965, Gordon Moore wrote an article for Electronicsmagazine titled "Cramming more components onto integratedcircuits" [1]. He predicted that the number of transistors on a chipwould double every 12 months into the near future. Although thisexponential trend has "tapered" to doubling transistors every 18months, it remains the driving force behind the integrated circuitsindustry including memory, microprocessors, and graphicsprocessors and has become known as Moore's law.

This law overthe years has provided a roadmap for product designers as theyplan efficient and effective usage of the transistors at theirdisposal. It has stood the test of time predicting an exponentialtrend for over 40 years. Process scaling has been the underlyingenabler of this trend starting in the early 90's and continuing totoday. Starting with 0.8 um process circa 1992, process scalingenabling feature size reduction by a factor of 0.7 has occurredapproximately every 24 months. This trend along with Moore'sLaw has spawned a number of exponentials all in the name ofincreasing performance. However, not all these residualexponentials can claim this longevity and many are not nearly asbenign as Moore's Law.

This paper is organized as follows. Thenext section briefly introduces the multicore system withfocus on the proposed FPGA-based design methodology. the main characteristics of the NIOS II softcoreprocessor are presented. The methodology followed to adaptthe NIOS II IP to the parallel SoC requirements is alsodescribed.

II.EXPONENTIALS PRIOR TO MULTI-CORE

Consider clock frequency which was on an exponential trend inthe mid 90's. From about 1993 with the Intel® Pentium®processor and continuing through mid 2003 with the Intel®Pentium® IV processor, clock frequency doubled every 18months to 2 years.

This was a driving force for increasingperformance of microprocessors during this timeframe. However,due to increased dynamic power dissipation and designcomplexity, this trend tapered with maximum clock frequenciesaround 4GHz. Along with increased dynamic power, static powercontinues to increase due to transistor source to drain leakagealong with gate leakage.

This leakage has been exponentiallyincreasing with scaling but has only recently been a concern as itbecame a significant portion of the overall power budget. Circuitdesigners have used stacked gates, body bias, and sleep transistorsto mitigate the S-D leakage problem and high K dielectrics toaddress the gate leakage problem.Power density is another exponential closely associated withpower dissipation and decreasing feature size.

Both powerdissipation and power density trends have essentially flattened byrequiring designers to remain within a particular power budgetand relaxing density requirements.Voltage scaling began in the early 90's when processor supplyvoltages began to deviate from the 5V standard.

This scaling wasrequired to avoid oxide stress as oxide thicknesses scaled withtransistor feature size. Gate capacitance scaled with feature size aswell resulting in overall lower power dissipation and counteractedthe adverse effects on power dissipation by increased clockfrequency.

Threshold voltage was also required to scale withpower supply in order maintain transistor performance. However,decreasing the threshold voltage led to increased sub thresholdleakage requiring designers to rethink the tradeoff betweenincreased performance with lower thresholds and increasedleakage leading to larger static power dissipation.
In the end,voltage scaling was short lived as was threshold voltage scaling tobetter control leakage and overall static power dissipation.Design complexity also continued on an exponential trend.Although it is very difficult to define a metric which encapsulatesthe resultant design effort, the increased design complexity hasrevealed itself in the past via exponential design team growth.Consider the feature set progression over the last several yearsincluding speculative execution followed byspeculativeexecutionand branch prediction.

Soon after, dynamic execution wasintroduced which included the aforementioned features plus superscalar and out of order execution. Then came multi-threadingfollowed by hyper-threading.All of these features were added toincrease the performance of the microprocessor at the expense ofincreased design and validation complexity.Low-end and embedded processors have historically trailed inmicro-architectural and performance enhancements. Many ofthese enhancements such as increased pipelining and increasedcache size exhibit diminishing returns in the light of increasedarea and power consumption. This has allowed low-end andembedded processors to close the overall performance gap whileproviding a reasonable tradeoff of area and power. Whenconsidering the limitations associated with voltage supply scaling,threshold scaling, and clock frequency scaling, along with designcomplexity increases, companies were already looking for analternative to the single core paradigm. Multi-core was thereforethe natural next evolutionary step in staying on the ever increasingperformance driven curve. In the next section, the multi-coredesign will be discussed along with how it addresses the need forincreased performance while abiding by strict power dissipationguidelines.

III.NIOS II-BASED MULTICORE ARCHITECTURE OVERVIEW

A..ArchitectureDescription.

TheNIOS II embedded softcoreis a reduced instruction set computer, optimized for AlteraFPGA implementations.
A block diagramof the NIOS II coreis depicted in Figure 2 [2].It canbeconfiguredatdesign time (datawidth, amountofmemory, included peripherals, etc.) in order to be optimizedfor a given application.
The most relevant configurationsare the clock frequency, the debug level, the performancelevel, and the user-defined instructions. The performancelevel configuration enables the user to choose one of thethree provided processors:

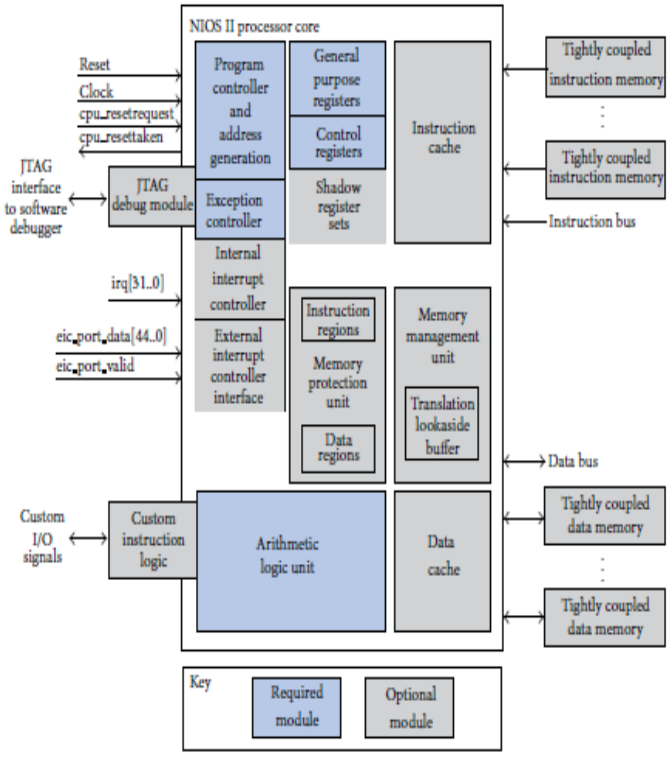

FIGURE 2: NIOS II processor core block diagram.

the NIOS II/fast, which results ina larger processor that uses more logic elements but is fasterand has more features; the NIOS II/standard, which createsa processor with balanced relationship between speed andarea and some special features; the NIOS II/economic, whichgenerates a very economic processor in terms of area, but verysimple in terms of data processing capability.In this work, we use the NIOS II economic version inorder to put a large number of processors in one FPGAdevice since we target a multicore SoC. NIOS II processorsuse Avalon bus for connecting memories and peripherals.
So, the implemented network interface is based on Avaloninterface. The Avalon interface is basically an interface thatcreates a common interface from different interfaces ofall memory and peripheral components of the system. Inthis work, we use the Avalon-Memory Mapped interface(Avalon-MM) [3]. It is an address-based read/write interfacesynchronous to the system clock. A wrapper, considered asthe network interface, has been implemented in order tobe able to integrate the NIOS with other components ofthe architecture, mainly the neighborhood network and theglobal router. This wrapper is a custom component which hastwo interfaces: one Avalon slave interface to be connected tothe NIOS and one conduit interface (for exporting signals tothe top level) to be connected to the networks. To assure communication,we only need addresses and data coming fromprocessors as well as the read/write enable signal to indicateif it is a read or write operation.

The NIOS-based wrapper isresponsible of transferring data and addresses of the attachedprocessor to the appropriate network (dependently on theaddress coding). Moreover, it communicates the data andaddresses received from networks to the processor dependingon its identity number.To design systems which integrate a NIOS II processor,Quartus II has as complement a SW called Qsys [4]. Qsysis used to specify the NIOS II processor cores, memory, andother components the system requires. Qsys automaticallygenerates the interconnect logic to integrate the componentsin the hardware system. The NIOS II-based SoC synthesisflow is summarized in Figure 3.In the implemented architecture, each processor is connectedto a timer which provides a periodic system clock tick,a system ID peripheral which allows uniquely identifying thePE, and to a local data memory. In addition, the controller iswith the NIOS core through the USB-Blaster download cableand providing debug information. Each processor is also connectedto a custom peripheral which integrated the networkinterface, allowing transferring the processor requests to theother components in the multicore architecture.

A network interface is usedto send and receive data transmitted over the network. Eachsubsystem contains the processor and some peripherals oraccelerators.The total number of PEs, defined by the numberof rows and columns, can be specified before synthesis.The design steps followed to implement the multicorearchitecture show the simplicity of the replication methodologyto be applied on any softcore processor and theconfigurability of the proposed SoC prototype. To build amulticore system, two steps have to be performed: to adaptthenetworkinterface tobe connectedtothe chosenprocessorand to define the corresponding communication instructionsbased on the processor instructions since the HW networksare managed by the SW program. Thelatter step is moredetailed in the next paragraph.

B. SWProgramming.

The main important instructions thatdepend on the used softcore are communication instructions.Such instructions are defined to assure communicating datathrough both networks. In this work, they are based onstandard C and specific NIOS instructions. Accessing andcommunicating with NIOS II peripherals can be accomplishedusingtheHWabstractionlayer (HAL) interface of theNIOS II processor. In fact, the HAL provides the C macrosIORD (IO Read) and IOWR (IO Write) that expand to theappropriate assembly instructions [5].

These instructionsallow accessing and communicating with different peripheralsconnected to the NIOS II processor. Each communicationinstruction consists in writing or reading data from thedefined address. In our case, the data has a 32-bit lengthand the address has a 12-bit length. The address field isdifferent depending on the communication network. In thecase of the global router the most significant bit (the bitat the 12th position) is set to "1," whereas in the case ofthe neighborhood network it is set to "0." To assure globalrouter communications, we distinguish different instructionsas illustrated below.

(i) MODE instruction allows setting the needed communicationmode and is executed by the controller:IOWR (NI BASE, 0x800, data), where data is thevalue that corresponds to the global router communicationmode (PE-PE, PE-Controller, PE-I/O Peripheral,or Controller-I/O Peripheral) as defined in themulticore architecture configuration file.

(ii) SEND instruction allows sending data through theglobal router: IOWR (NI BASE, address, data), whereNI BASE is the address of the controller networkinterface based custom component when the instructionis executed by the controller or the PE networkinterface based custom component when it is executedby the PE. The address field (11 bits) containsthe following:
(1) the identity of the PE receiver (mode PE-PE,mode Controller-PE, and mode I/O Peripheral-PE);
(2) "00000000000" (mode PE-Controller andmode I/O Peripheral-Controller);
(3) the peripheral address (mode PE-I/O Peripheraland mode Controller-I/O Peripheral).

(iii) RECEIVE instruction allows receiving data throughthe global router: data = IORD(NI BASE, address). Itanalogously takes the same address field as the SENDinstruction.

In the case of the neighbouring communication, SENDand RECEIVE instructions are also encoded from IORDand IOWR NIOS macros. compared to the global routerinstructions, they only differ in the address field, whichcontains the direction to which the data will be transferred(0: north, 1: east, 2: west, 3: south, 4: northeast, 5: northwest,6: southeast, and 7: southwest).Altera provides the NIOS Integrated Development Environmenttool to implement and compile the parallel SWprogram. The following section presents the implementationand experimental results of the proposed multicore design.

IV. CONCLUSION

This paper provides an overview of the reasons for moving tomulti-core designs along with the design challenges and a view ofthe research directions for design automation. We are entering aperiod of dramatic change. The end of direct performance-scalinghas led to an exciting time for system architects, who areinventing new ways to maintain the trend of the historic advancesin system performance.

Their need to explore more novelarchitectures provides a great need and opportunity to applydesign automation technology to the early, pre-RTL, phase ofdesign. This is the beginning of the long-awaited expansion ofdesign automation to the system-level, but the key enabler is theintegration of performance, power and physical analysis. We needto take advantage of the reuse in multi-core designs and theemerging standards to enable this next advance in EDA.

In this paper, we have presented a softcore-based multicoreimplementation on FPGA device.The proposed architectureis parametric and scalable, which can be tailored accordingto application requirements such as performance and areacost. Such a programmable architecture iswell suited for dataintensiveapplications from the areas of signal, image, andvideo processing applications. Through experimental results,we have demonstrated better performance gains of oursystem in comparison to state-of-the-art embedded architectures.The presented architecture provides a great deal of performance as well as flexibility.

7. REFERENCES

[1] Gordon E. Moore, Cramming More Components ontoIntegrated Circuits. *Electronics*, April 19, 1965.

[2] Altera, "AvalonMemory-Mapped Interface Specification," 2006,http://www.cs.columbia.edu/~sedwards/classes/2007/4840/mnlavalon spec.pdf.

[3] Altera,QuartusIIHandbookVersion 11.1,Volume 1:Design andSynthesis, 2011, http://www.altera.com/literature/hb/qts/archives/quartusii handbook archive 111.pdf.

[4] J. O. Hamblen, T. S. Hall, and M. D. Furman, *Rapid Prototypingof Digital Systems*, Springer, New York, NY, USA, 2008.

[5] Charlie Johnson, Jeff Welser, Future processors: flexible andmodular. In *Proceedings of the 3rd IEEE/ACM/IFIPInternational Conference on Hardware/Software Codesignand System Synthesis*, CODES+ISSS 2005, Jersey City, NJ,USA, September 19-21, 2005, 4-6

[6] http://ramp.eecs.berkeley.edu/