VOL. 11, NO. 23, DECEMBER 2016 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

THE VANTAGE OF UTILIZING FPGA IN THE DESIGN OF AN EMBEDDED MULTIPROCESSOR

Mays Q. Sedeeq, Muataz H. Salih, Omar F. Yousif and Nada Q. Mohammed
School of Computer and Communication Engineering, Universiti Malaysia Perlis (UniMAP) Perlis, Malaysia
E-Mail: muataz@unimap.edu.my

ABSTRACT

There are recent needs to design an embedded multiprocessor that will overcome the limitations in the performance of a uniprocessor. The deputation to achieve real time deadlines and overcome area and power restrictions on a single chip opened the door wide open for Multi-Processor Systems on Chip(MPSoC) as a solution. Designing MPSoC are very challenging, not only because of their complexity, but because of challenges like debuggers which have typically not been tightly coupled. The second issue is how the platform will provide synchronisation among the connected processors, which is the core factor in the success of such systems. The time consumed in the development and the verification of the design is also an issue where the need for a significant tool to accomplish those important phases in design life cycle at minimum time. It is known that the selection of the right platform, development tools and operating systems could be the difference between success and failure. Recent FPGA devices have increased their performance and gate capacity giving the ability to implement complex logic systems on a single programmable device. FPGA supported with NiosII processor empowered by the Qsys delivers unprecedented flexibility for cost-sensitive, real-time and applications processing needs. FPGA board was validated through utilising it in constructing an enhanced embedded concurrent processor. NiosII Embedded Evaluation Kit (NEEK) was utilised to construct the proposed design. The proposed design was tested by implementing a demanding application which was the Mandelbrot-set. The end design showed significant performance enhancement compared to a single processor. The result showed an enhancement with frequency that reached 1GHz and exceeded 20 times the speed of that in single processor. The obtained images was updated every frame while it was updated every five frames in the case of single processor.

Keywords: embedded processor, FPGA system design, MPSoC, multiprocessor, shared memory.

INTRODUCTION

The trend towards building a whole system on one chip is rising as the need for high performance and less area and power consumption. The complexity of such systems created an urgent need for an adaptive platform that will carry out the faced challenges. Designing embedded multiprocessing systems are challenging, not only because of their complexity, but because issues like debuggers which have typically not been tightly coupled [1]. Further complicating the debugging is that the multiprocessors might be co-located on a single die, share a single printed circuit board, or be on two (or more) separate circuit board. The ability to quickly modify the selection of internal signals (without the need to execute a very time consuming recompile), and the ability to probe internal signals (without the need to use FPGA fabric and potentially introduce timing violations) are major advantages when debugging FPGA designs.

The synchronisation of the connected multi processors will rise as a second issue when processors in a multiprocessing system communicate between each other generally via shared resources such as peripherals; memory, etc. miss coordination of the processors work while their access to the shared resources will cause the problem of thread safety [2]. The protection of shared resources is provided by NiosII processor through the access to the hardware mutex core. The hardware mutex core makes sure that the ownership of the mutex at any given time is yielded to only one processor. The hardware mutex core is a simple Qsys component and not an internal feature of the NiosII processor. Cooperating processors and through the mutex are allowed to accept access permition of only one processor to a certain hardware peripheral at a time. This mechanism is considered to be very helpful in eliminating thread safety issue that can take place as different processors attempt to use the peripheral at the same time.

When designing large scale systems especially with large number of connected processors the development and the verification of the end design could be time consuming [3].The Qsys is a system integration tool that saves significant time and effort in the FPGA design process, by automatically generating interconnect logic to connect intellectual property (IP) functions and subsystems. The Qsys has the power to start simulation faster with automatic test bench generation and through using the verification IP suite.

In the proposed design an enhanced embedded concurrent multiprocessor built on one chip is presented. The design is constructed of three NiosII processors in a master slave hierarchal style. NiosII/f is the selected version since it addresses performance providing high speed processor. The result will be gained in two scenarios; the first is when implementing the chosen application on one processor and displaying the result on the LCD touch screen. The second is when implementing the application on the proposed enhanced embedded concurrent multiprocessor where the result is displayed on the attached LCD touch screen and VGA port.

In the aspect of validating the proposed design and performing an application the selected one is Mandelbrot-set (M-set) which has a very simple formula

presented below. Despite that the M-set involves only addition and multiplication it presents at the same time very interesting results.

$$Zn+1= (Zn)^2 + C \quad (1)$$

C in the above formula is a fixed constant residing in the complex plane and n has the values of 0, 1, 2…reaching towards infinity. According to the above formula, Z_0 is the value of Z_1 in the former calculation. This application has repeated manner in calculating the magnitude of Z which may remain finite or it may diverge in to infinity. As a given point shows divergence into infinity the point is considered to be out of the M-set, or if it doesn't diverge then it is in the set. What made this Application to be considered demanding is that there are number of calculations needed to be performed for each pixel on the displaying screen. Since the utilised LCD screen has the resolution of 800*480 this means that the function will be executed for 384000 times which is time consuming and a challenge for any computing system.

The proposed design showed an enhancement that exceeded 20 times the speed of that in single processor with frequency that reached 1GHz. The displayed result was updated every frame while it was updated every five frames in the case of single processor.

RELATED WORK

The field-programmable gate array (FPGA) is an integrated circuit designed to be assembled by a customer or a designer after manufacturing. The FPGA configuration is typically defined using hardware description language (HDL).Recent FPGA devices has increased their performance and gate capacity giving the ability to implement complex logic systems on a single programmable device

The true parallel nature of FPGA made parallel and pipelining processing easily implemented. An embedded multiprocessor core with its full architecture is proposed by [4] designed for realistic with an ability to perform arithmetic, logical, shifting and bit manipulate operations. Homogeneous Embedded RISC processors are contained in the proposed quad processor corealong with pipelined processing units, multibus organization and I/O ports and with the other functional elements needed for the implementation of embedded SoC solutions. The performance issues of the designed Quad core processor like area, speed and power dissipation and propagation delay are analysed at 90nm process technology using FPGA Xilinx tool.

In the design of an embedded processor on chip, the ability to achieve area saving is very important. [5] Presented a Multi-Threaded (MT) soft processor for area reduction in SoPC implementations. An MT processor permits multiple programs to access the same IP without the necessity for the logic replication or the replication of entire Processors. The first design was a single-threaded processor that is instruction-set compatible to Altera's NiosII soft processor. The designed processor is approximately the same size as the NiosII Economy version, with equivalent performance. An assurance was made that the designed processor has 4-way interleaved multithreading capabilities. Compression of the area usage and performance of the MT processor versus two CMP systems. The results showed that there was achievement in area savings of about 45% for the processor itself, in addition to the area savings due to not replicating CI logic blocks.

Employing commercial evaluation boards and their design templates provide the designer an ahead start in the development cycle and can finally reduce development time. [6] Presented the hands-on demonstration of an FPGA evaluation board running a simplified example of a control system interface. Explanation was made that the advances in development tools, will allow designers to bridge the gap between traditional FPGA-based designs and embedded processor development.

The retain age of programmability as an advantage in FPGA can be even cost effective, while obviating the risks in producing silicon was a key feature recognised by [7].Demonstrating the effectiveness of FPGA based soft-multiprocessors for high performance applications. Deploying IPv4 packet forwarding on a multiprocessor on the Xilinx Virtex-II Pro FPGA. The design achieved a 1.8 Gbps throughput and loses only 2.6X in performance (normalized to area) compared to an implementation on the Intel IXP-28OO network processor. Developing a design space exploration framework using integer linear programming to explore multiprocessor configurations for an application. With the utilisation of the developed framework, led to achieve more efficient multiprocessor design surpassing the performance of the proposed hand-tuned solution for packet forwarding. This led to the fact of that with the existence of high-performance programmable platform for an application niche, then it will be a cost effective implementation medium.

The high performance is a must be feature in any platform specially when dealing with application requiring large calculations such as image processing. In such domains high processing power, limited size, limited power consumption, limited production volumes, procurement issues have to be conciliated. This is important to guarantee shipping and support during long lifecycles. There is also a great need to evolve the design during the lifecycle to remain competitive. A promising solution is described by [8] where a new generation of computation intensive signal and image processing systems is implemented. Compared to classical FPGA implementations, the high performance of FPGAs did not only allow the dramatical reduction of development and evolution cost but also gave the ability to make systems more intelligent through the access to sophisticated data dependent algorithms.

In the design of a multiprocessing system there is a need for an adaptive platform that will provide an interactive tools that make designing such systems less time consuming. [9]Proposed a complete flow, composed by a programming model and template architecture. The

framework allowed writing a parallel application by utilizing a shared memory model. It dealt with the consistency of shared data, with no need of hardware coherence protocol, but used a software model to properly synchronize the local copies with the shared memory image. The estimation could be applied both to a scratchpad based architecture or a cache-based one. The architecture was synthesizable with standard IPs, such as the soft cores and interconnects elements, which may be found in any commercial FPGA toolset.

FPGA VS OTHER PLATFORM

Since the success of any embedded system begins right from the evaluation phase, selecting the right platform is a significant step in the design of any embedded system. The following paragraphs shall present the strength and weakness of different platforms and why FPFAs are considered the best in the design of MPSoC.

A. FPGA vs ASIC

When system designers at the former stages of designing their system and starting to select the medium of implementation, they often make a choice between two platforms, which are either FPGAs or Application Specific Integrated Circuits (ASICs). First the base of making a selection between those platforms often is comparing the cost that is very much related to area. Second is compromising between the two important factors which are performance and power consumption. Along the excitants of both platforms there have been some efforts to measure the differences [10]. Different values are supplied to designers through FPGAs and ASIC (Application-Specific Integrated Circuit). Generally, a careful assessment should take place before selecting any one of them over the other. Lower speed, complexity and volume designs were the reason behind selecting FPGAsin the past, nowadays 500MHz performance barrier is successfully pushed by recent FPGAs. With the increase of new logic density and the host of other characteristics, such as embedded processors, DSP blocks, clocking, and high-speed serial at ever lower price points, FPGAs are an obliging proposition for nearly any type of design. Some of the trends that make FPGAs a better choice than ASICs for the growing number of higher-volume application:

- Increasing IC design costs.
- FPGA offers time-to-market advantage.
- Weak economy asking for low-cost technologies
- FPGA can be re-programmed in the field to fix bugs.
- Reusability, and lower non-recurring engineering costs.
- Some FPGAs have the potentiality of partial re-configuration that lets one part of the device to be re-programmed while other portions continue running.

The design flow of FPGA eradicates complexity and time-consuming floor planning, place and route, timing analysis, and mask / re-spins stages of the project while the design logic is already synthesized to be placed onto an already asserted, characterized FPGA device. In Figure-1 shows FPGA vs. ASIC design flow comparison. Modern FPGAs has the ability be reprogrammed at "run time," leading to the idea of reconfigurable computing or reconfigurable systems (CPUs that reconfigure themselves to suit the task at hand). However, FPGA has the disadvantage of higher power consumption compared to ASIC and other issues like:

- Not a right device for high volume applications.
- Costlier than custom silicon.
- No on-chip analog functions.

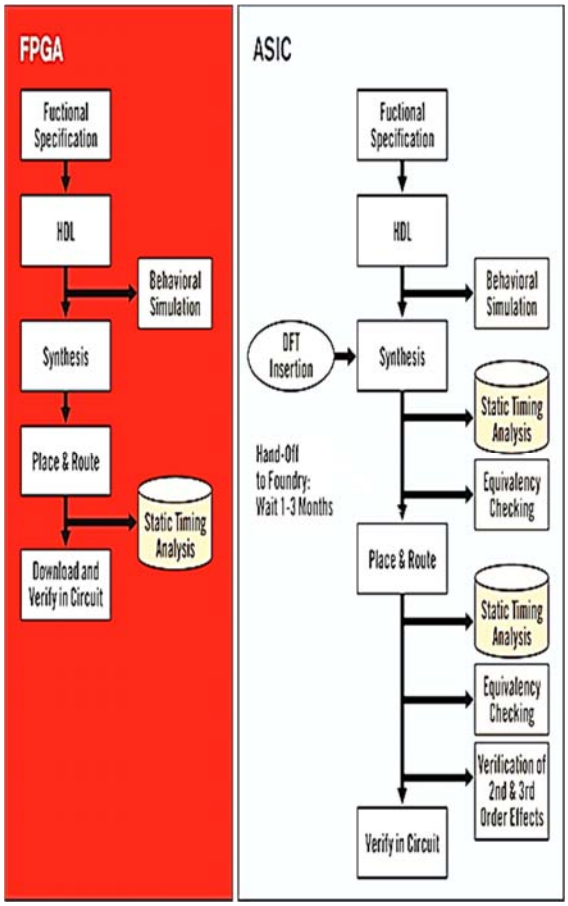

Figure-1. FPGA vs. ASIC design flow comparison [10].

B. FPGA vs. CPLD

Making a decision on which platform to utilise, either FPGA or Complex Programmable Logic Devices (CPLD) have a major dependency on the goals of a design. [11].Designers consider several factors when selecting a logic solution for use in handheld applications including time to market, design flexibility, standby power consumption, board space, and system integration options. While reviewing some of the techniques for designing with FPGAs and CPLDs used in designing a large-scale multiprocessor various consequences regarding high-speed design has been covered. Consequences like choosing the appropriate programmable logic family, finding high speed circuits with handed families, and enhance the

synthesis software. The widespread use of counters and controllers from simple digital alarm clocks to computer memory pointers, made designers seek for a best platform to successfully implement them. In [11] made adscription of the techniques that have been utilised where FPGAs and CPLDs are used to design controllers and counters. The conclusions were that CPLDs provide simple implementation of a high speed counters [12]. Yet in equivalence, FPGAs are better in terms of speed and more suitable for this purpose. One of the foundations was that with very little cost in area would enhance the counter design leading to performance and routability improvements.

However, those gains cannot be realized without intimate knowledge of the FPGA architecture and a great deal of manual work on the part of the designer. As a result FPGAs and CPLDs are both famous digital logic chips. But in the aspect of the internal architecture, there are very obvious differences.

- The number of logic blocks held in FPGA goes up to 100,000 on the other hand CPLD holds about few thousands. For the mentioned reason the FPGAs are considered best suited for more complex applications. In the other hand CPLDs and for their less complex EEPROM-based it suits more for small gate count design where the delays are much predictable and it is non-volatile.
- FPGA is suited for timing circuit because they have more registers, but CPLD is suited for control circuit because they have more combinational circuit.
- When synthesising the same code for FPGA for many times, you will find out that each timing report is different. But it is different in CPLD synthesis, you can get the same result.
- In terms of the amount of resources, FPGAs are generally much larger in equivalence to CPLDs.
- Adders, multipliers, memory and other complex embedded functions are more contained inFPGAs.
- Embedded flash is contained in CPLDs where it's utilised as configuration storage. On the other hand FPGAs generally and not all the time, demand external none-volatile memory.

C. FPGA vs. Microcontroller

In this aspect designers use microcontrollers due to their simplicity and ease while dealing with them and due to their low cost. In [13] the designers implemented a multiprocessor configuration of 8051 microcontroller Chip. In a proposed hardware part, a presentation of a single wire bus system. The system is built with microcontrollers in a master-slaves architecture. The mode of the utilised UARTs of each microcontroller is in mode 3, which is multiprocessor mode. An example of multiprocessor having two slaves is represented in Figure-2. The receiver (Rx) and transmitter (Tx) pins of all the microcontrollers are connected together. A single bus is used to connect all the microcontrollers where the transmitting and receiving of the data cannot be done at the same time. This single strategy provided limited ability for data transfer across the system. The situation of having majority of idle slaves ignoring the transfer of data, while only few slaves are able to receive, raises the issue of wasting resources through having idle processors

The other experienced issue was that, despite the 8 bits provided for addressing giving theoretically the ability to connect up to 256 slaves but through practice showed that too many slave devices would cause loading effects [13]. This is one of the characteristics of microcontroller which is related to small applications and quantities otherwise the system will suffer from loading effects. Since the microcontroller has many modes and types, the type of the master and slave controller and whether they are similar or not,where it should be in the mind of the designer to choose the right mode otherwise it will reduce the efficiency of the system.

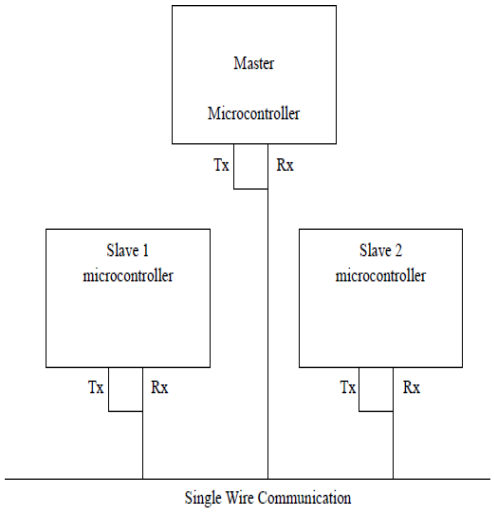

Figure-2. Multi-microcontroller configuration with master slave connection [13].

In reality, there has been a discussion to make a fair comparison of what to utilise, an FPGA or a conventional hard IP microcontroller. The great enhancements in power efficiency and the handed flexibility tend FPGAs as a platform that will prevail in future.

- FPGAs are more suitable for real time applications for the concurrency provided by these platforms.
- FPGA are flexible to the contrary of the rigid microcontrollers. The former has the flexibility add/subtract the functionality.
- FPGA is preferred in military application for two primary reasons, mainly the hard wired bases of FPGAs and secondly is the life time FPGA base development is longer.

On the other hand, microcontroller alters too frequently. To spare the design from being obsolete, huge amount of rework is demanded to keep step with changing technology. However, microcontroller up till now

isefficient and low cost compared to FPGAs but particularly for small applications and small quantity.

D. FPGA vs. DSP

The ability yielded by the FPGA in designing multiprocessors are compared with that provided by DSP (Digital Signal Processor) platform in [14]. The TNETV3020 multicore based DSP shown in Figure-3has been chosen to be compared with that based FPGA platform. The researchers presented the TNETV3020 multicore as a very efficient design and how the chosen platform enhances the performance.

The TNETV3020 DSP was presented as system capable to do the required complex math in real time and satisfy application high demands [14]. Yet, the presented multi-core DSP architectures showed many challenges in hardware architectures, the organization and management of the memory, operating systems, platform software, compiler designs, and tooling for code development and debug. In the matter of programming multi-core DSPs it still very challenging, where the unsolved issue of optimally partitioning a piece of sequential code across the multiple cores not missing the lack of debugging a design and visibility in DSP platform.

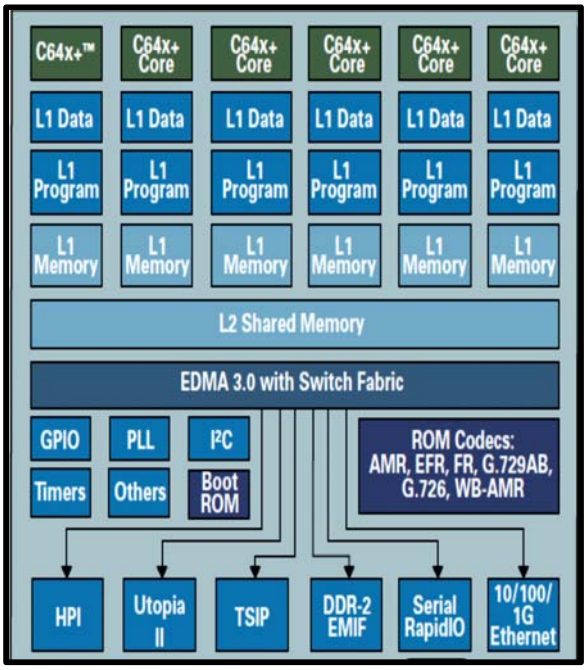

Figure-3.TNETV3020 Multicore DSP platform [14].

The development of efficient and easy code to be used along with tools utilised for real-time debug is very crucial as the chance for bugs rise up. This case takes place especially once starting to deal with both time and space. Answers are set to satisfy multiprocessor debugging challenges for more rapid expansion which makes vantage of modern processor devices with multiple cores and heterogeneous systems based on FPGAs. Therefore, it can be distinguished that the two main obstacle facing developers of large DSP systems are the management and debugging of these complex configurations and recognizing the important effect of this on time to market.

In general, DSPs are considered to be a specialised microprocessor typically programmed in C, and for better performance sometimes with assembly code (low level language). It is most suitable to extremely complex maths-intensive tasks, with conditional processing. But it is limited in performance by the clock rate, and the limited number of useful operations that can be done per clock

On the other hand, FPGA is an uncommitted "sea of gates". Programming these platforms is achieved through connecting the contained gates together to make multipliers, registers, adders and so on. The blocks in FPGAs can possibly be very high level with a range that starts from a single gate to an FIR or FFT. Multipliers have been especially included in modern FPGAs to efficiently perform DSP tasks [15]. As a matter of fact, the majority of systems are constructed of lots of blocks, where FPGA is the best to implement some of them, other blocks in DSP. DSP is best suited approach for lower sampling rates and increased complexity. While FPGA is best suited for higher sampling rates, particularly aggregated with fixed, repetitive tasks.

FPGA vs. SBC

Another utilised platform in embedded system design is the Single Board Computer (SBC). This platform tends its self as a complete computer on an individual board, along with one or more microprocessors, memory, I/Os and some features that is considered required in functional computing. Based on SBC there is the Single Board Computer Multiprocessor (SBCM) shown in Figure-4 and the enhanced one Enhanced Single Board Computer Multiprocessor (ESBCM) shown in Figure-5, which is a hybrid architecture "based upon empirical analysis using discrete event simulations and Monte Carlo techniques" [16]. The ESBCM design shows performance improvement and scalability characteristics over the former SBCM design.

A point-to-point links where added between the processors of an SBCM to create the communication network which complemented the system bus. The result of this addition was the Enhanced SBCM (ESBCM). ESBCM showed ability to reduce bus traffic. The enhancements were expanded scalability of the system and second the overall performance was improved.

Yet, the issue introduced by the point to point links is, they are not area efficient. And almost each existing SBC model has major flaws considering it as hardware with no capability to function without running a non-free program [17]. As a matter of fact customizing the SBC is impossible as the processor CPU and I/Os are already fixed. These platforms cannot be scaled to accommodate latest processors in future, as the CPU and I/O sections are integrated on a single board.

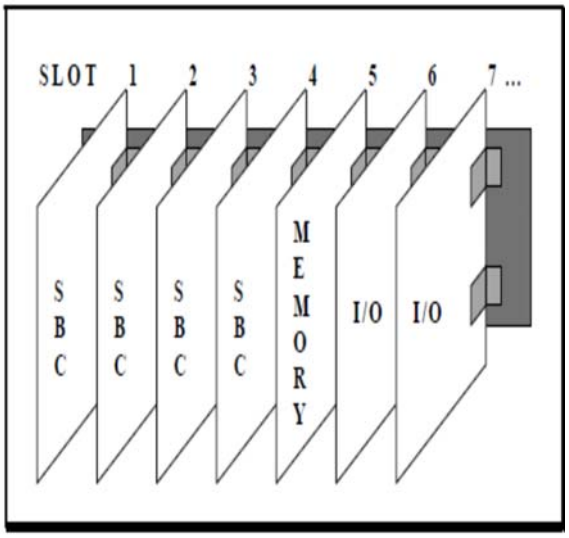

Figure-4. General SBCM hardware configuration [16].

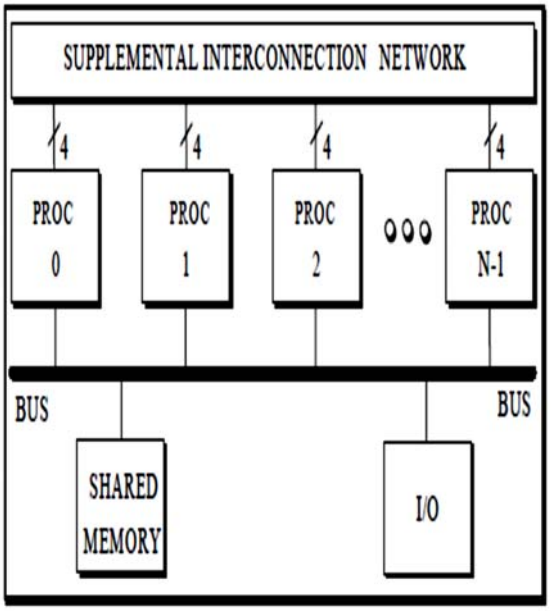

Figure-5. General design of an ESBCM cluster [16].

Table-1. Comparison of embedded platforms.

Platform	Adv.	Dis.	Proramm. Language
FPGA	True parallel nature. Reconfigurable. Best suited for real time application. Flexible functionality. FPGA can be re-programmed in the field to fix bugs. High logic density. Obsolescence protection. Fast in time to market	High power consumption. Expensive. Limited size options.	VHDL Verilog
ASIC	Low power consumption.	Increasing IC design costs Don't provide flexible programmability.	BASIC, GW-BASIC
TNETV3020 DSP platform	Low power consumption. Low cost. Hard real time processing.	The clock rate limits the performance. Debugging and visibility challenges. Fixed design not reconfigurable. Very challenging in programming. General purpose processor	C Assembly
SBC	Low power consumption. Low cost. High computational performance	Not customizable. Fixed design not reconfigurable.	C
CPLD	Low power consumption. Low cost. Programmable. High speed	High power consumption. Limited no. of blocks. Much less resources than FPGA.	VHDL, Verilog

PRE DESIGN CONSIDERATIONS

An efficient methodology for designing embedded multiprocessor systems is to base the design completely or partially on an FPGA evaluation board. This idea has the advantage of that evaluation boards come with design examples and templates for the development suite, which can decrease development time. The evaluation board is a launched working and affirmed hardware system that might already hold various components that will be included in the final design. Schematic and layout design files are sometimes supplied by the board vendor enabling modification of the design to create a partially customized board. Since there are at least 30 distinct FPGA evaluation boards commercially available, it is most beneficial to match the potentiality of the evaluation board to the I/O and interface necessities of the application. This scheme tailors the design to the application rather than adjusting to a bus standard. If there

is not a commercial evaluation board that fits the application requirements, an FPGA chip can be chosen for the foundation of an entirely custom board design. Eventually the board or chip has been chosen, the form factor of the board and the enclosure can be specified. At last creating a project using the development platform tools to design the FPGA hardware and embedded processor software.

SUGGESTED METHODOLOGY

A fast design of multiprocessor systems that share peripherals on FPGA is safely achieved by using NiosII and Qsys tool. The NiosII processor system generally denotes to a system with a processor core, a set of on-chip peripherals, on-chip memory, and interfaces to off-chip memory all implemented on a single device [18].Qsys in the other hand is a system development tool with an ability to create FPGA designs including processors, peripherals, and memories. TheQsys tool also provides an easy tuning and modification of the hardware which provides optimal system performance. System complexity can be reduced through the construction of a multiprocessor that share the mutex peripheral in a hierarchal style. The experienced complexity reduction is achieved through classifying the design into discrete subsystems. A user-define interface is exported by each subsystem providing the demanded the hierarchal linking. Writing software for the connected processors is considered as one of the faced challenges in building a multiprocessing system where assurance must be given that they don't conflict with each other and work efficiently. The hardware mutex should be included in such systems to prevent any kind of interposition among multiple processors. The mutex core allows processors in a multiprocessing system to own the shared peripheral for a certain amount of time. The shared peripheral is protected through the temporary ownership of a peripheral by a single processor. Mainly, this protection is demanded with possibility of corruption by the actions of another processor. This corruption is prevented through writing a software that before accessing the peripheral it waits to acquire the mutex, making sure of mutual exclusive access. There are number of features included in NiosII SBT for Eclipse which can be very helpful in the development of software for multiprocessor systems. Performing simultaneous debug for multiple processors by NiosII SBT for Eclipse is the most notable feature. Figure-6 shows multiprocessor system with shared memory.

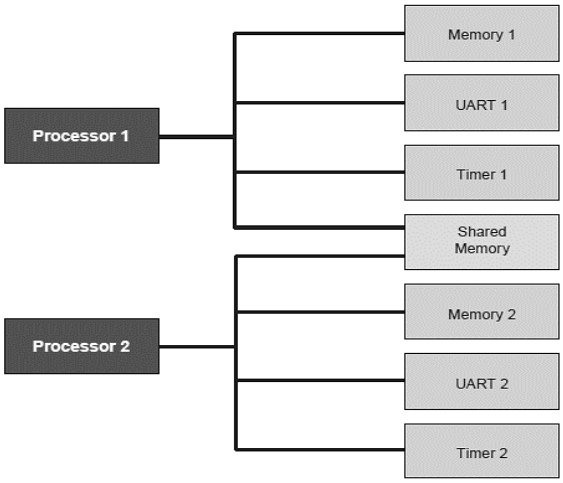

Figure-6. Multiprocessor system with shared memory.

VALIDATING FPGA IN A MULTIPROCESSOR DESIGN

In an attempt to validate FPGA in designing multiprocessor system, a comparison is made between single processor and FPGA based multiprocessor performance. And this will be achieved through two phases. The first phase is to enhance the implementation of an embedded concurrent processor using a tightly coupled shared memory completely implemented on FPGA where the selected board is the NEEK board. The second phase is to select an application that will be implemented on both processors where the results will be compared.

FPGA BASED MULTIPROCESSOR

The utilisation of FPGA board handed the core functionality of the proposed design which is the concurrent processor function unit shown in Figure-7. Each task was subdivided into subtasks and theses subtasks where executed concurrently on the connected processors. Assurance was given to distribute the handed tasks across the processors evenly performing a suitable load balancing algorithm which provided the ability to minimise the processing time and enhance the throughput.

ARPN Journal of Engineering and Applied Sciences

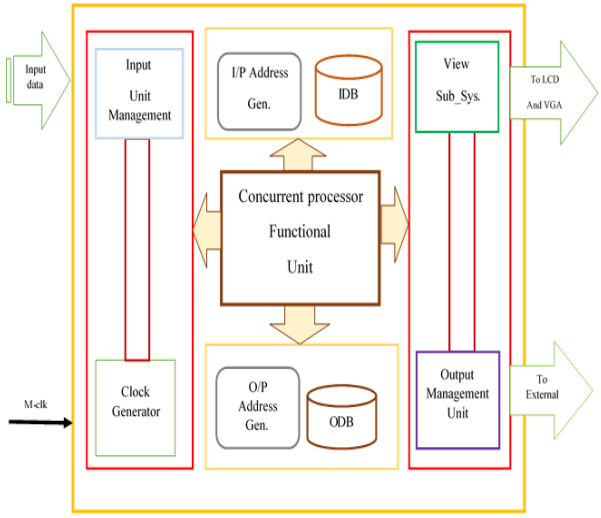

Figure-7. System top level design.

The process of organising the data entry can be handled by the memory management unit. Any entered data can be processed concurrently by the concurrent processor functional units. In an array style the processed data will be stored in the Output Data Buffer (ODB) where with the assistance of the O/P address generator the processed data will be provided with addresses. The demonstration of the processed data will be the responsibility of the Output Unit Manager to either LCD and/or VGA through also view sub-system unit.

The core of the proposed design is the Concurrent Functional Unit shown in Figure-8. This unit is constructed of three NiosII/f processor connected in a hierarchal master slave style through the Qsys. The top level of the design is constructed of one processor "master", JTAG UART and system ID which is a register that holds information about each component in the system. The proposed multiprocessor is shared peripheral type which in this case as shared memory multiprocessor. The shared memory is the on-chip RAM also resides on the top level and under the control of the master. The slaves in the subsystem level access the shared memory under the protection of the mutex to avoid the issue of thread safety. What is available through the Qsys is the Avalon Memory Mapped interface family which was utilised in this design. Avalon-MM actually defines appropriate interface for high speed data streaming, writing and reading registers and memory along with controlling on-chip devices which happens to be very significant to the performance of the end design. The slaves in the subsystem are connected through the Avalon bridges in a point to point style along with the mutexes.

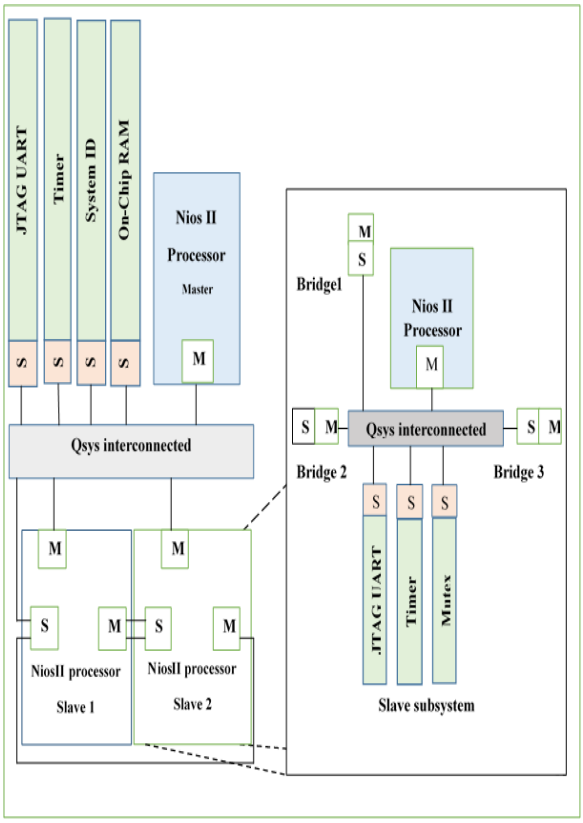

Figure-8. Concurrent functional unit.

To build and download the software part and for safe running of multiprocessors on the shared memory, in independency each processor's exception address was set in the Qsys. The NiosII software for Eclipse where started, and the projects where imported to Eclipse. These projects are:

- app-monitor
- bsp-monitor
- app
- bsp

The app-monitor and bsp- monitor are to be run on the master processor since monitoring and coordinating the work of the two connected slaves is the master's responsibility. While the app and bsp are to be run on the slaves in the subsystem. The positioning of the code is very important there for it is the first step to be taken so that the on-chip memory was partitioned as shown in Figure-9. Through the NiosII SBT the memory was partitioned to allow the multiple connected processors to run their software from different part of the same physical memory. Those partitions where made so that each processor will run its software from its own eight KB of the on - chip memory. Each processor's partition is separated from the others by other eight KB. The boundary of the code region ends at the foundation of the other processor's exception address in the physical memory. There are five linker sections (listed below) for

every utilised processor, where the linking and the allocation of the processors software where ensured through the SBT to be at set addresses in the memory. Theses linker sections are:

- (.text) refers to an executable code.
- (.rodata) refers to the utilization of whatever read-only data, while executing a code.
- (.rwdata) refers to the storage of read write variables and pointers.
- (.heap) refers to the location of the dynamical memory.
- (.stack) refers to the storage of function-call parameters and other temporary data.

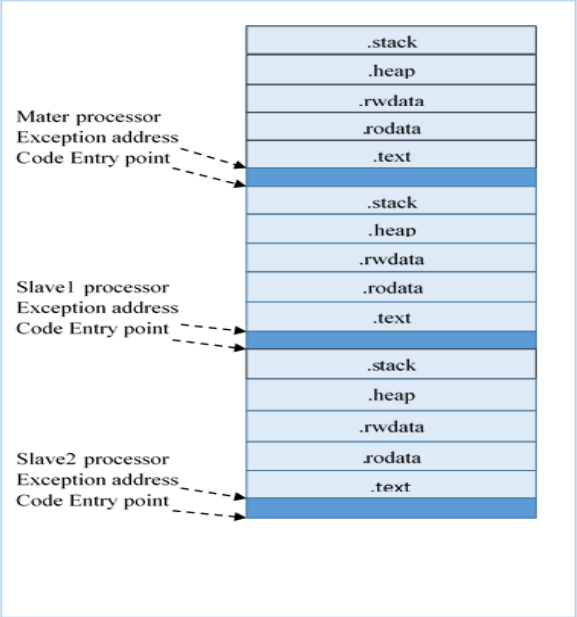

Figure-9. Partitions of the on-chip memory.

To make sure that the multiprocessor will boot from the exact bit of the executable code at the exact address, the booting memory was also partitioned shown in Figure-10. The multiple processors were permitted to program their bootable code into one flash device through the NiosII flash programmer.

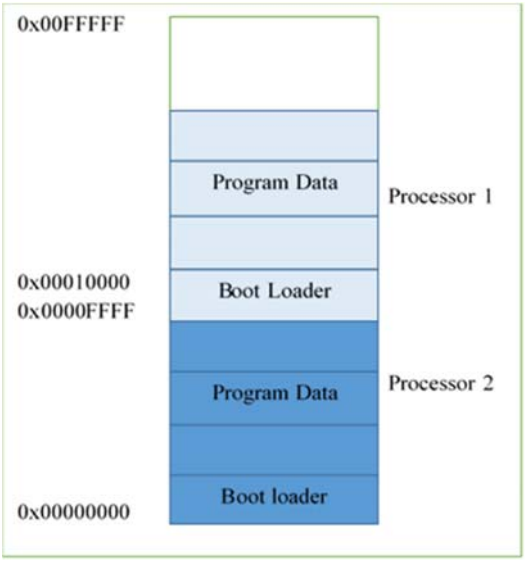

Figure-10. Two processors booting to 1MB flash memory.

The final and significant step is debugging the design where it was obvious the outstanding features provided in NiosII SBT for Eclipse to set the debugging environment and starting to debug the software. As the NiosII SBT for Eclipse was started, launch configuration are created for every target processor and debugging is started with these configurations. It was noticed the remarkable ability handed by the NiosII SBT for Eclipse to launch and stop the debugging sessions independently. As the design enhancement was completed the Mandelbrot- set was applied as an application to validate the end design. Realising how beneficial load balancing and its major effect on enhancing the performance of the multiprocessing system, the load balancing step was considered significant. The selected load balancing algorithm was Central Queue algorithm. Through implementing this algorithm the requests handed by the slave processors for a process was stored in a cyclic FIFO queue. The master was working as a manager that would distribute the workload and assign the process to the slaves that had less loads. So an immediate serving was provided by the slaves to any given task and coordinated by the master. This step will make full advantage of the utilised processors eliminating the case of having idle processor at a given time and wasting the power of a resource that would eventually decrease the performance of the whole system.

RESULT

After completing the design stages and staring the testing part, the result will be obtained in two scenarios as shown in Figure-11. The selected application will be processed by either one processor where the result is displayed on the LCD touch screen, or the second scenario where the application will be processed by the enhanced embedded concurrent processor and the result will be displayed on both LCD and VGA port. The concluded results will show the difference in performance of the both

processors in concern of processing speed and the quality of the obtained image.

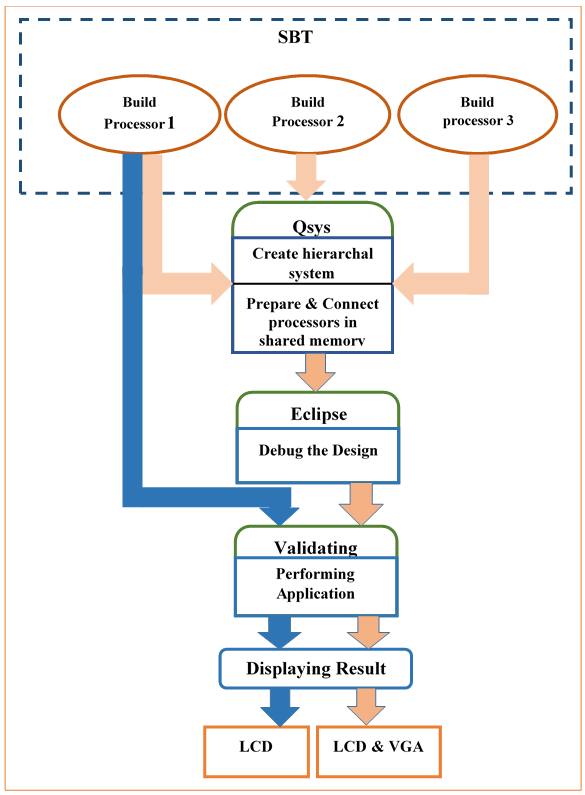

Figure-11. Observational platform diagram.

When the M-set is implemented on the single processor which is referred with the blue path in Figure-11, the speed of the obtained result was very low and this is shown in Figures 12 to 15.

Figure-12. M-set plot displayed after 5 seconds.

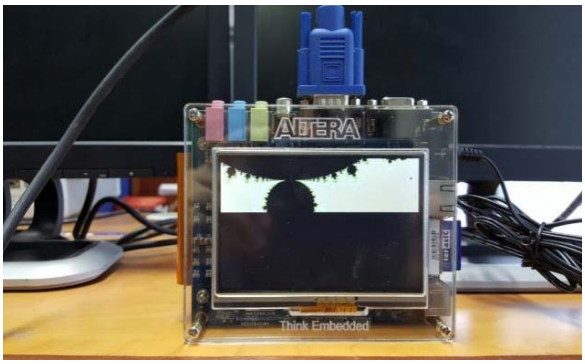

Figure-13. M-set plot displayed after 10 seconds.

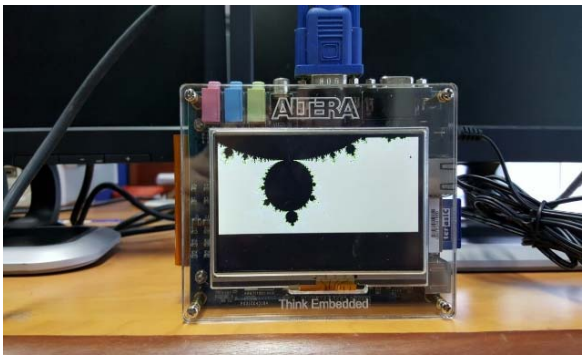

Figure-14. M-set plot displayed after 15 seconds.

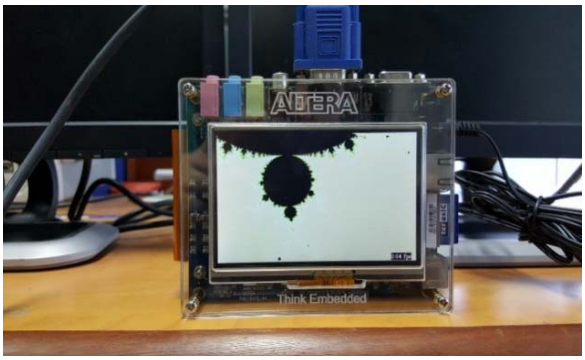

Figure-15. M-set plot displayed after 20 seconds.

It's obvious that the speed of the rendered result is slow, where the display of the output data is updated every five frames.

When M-set was applied to the proposed design, the computing tasks were divided over the two slave processors coordinated by the master processor. As one of the slave finishes a task and its workload is less than its threshold load it sends a request to the manager which is the master in the proposed deign. The handed requests are kept in a cyclic FIFO queue and the master will assign each request in it turn to a processor. This immediate serving of a given task by the connected processors made concentration on the generated pixels possible, where Mandelbrot function for the next frame was computed all in time. The master processor is also responsible of the displaying of the processed data on both VGA port and the

LCD. Figure-15 shows the M-set plot as an output of the enhanced embedded concurrent processor.

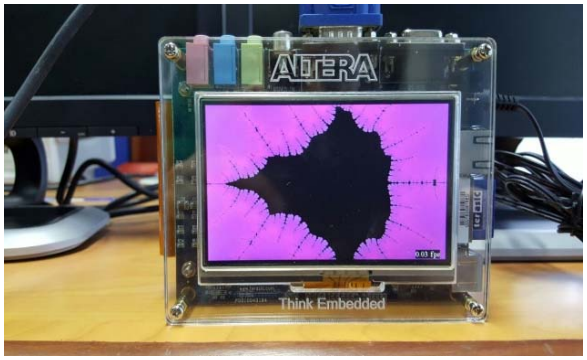

Figure-15. M-set plot on the enhanced embedded concurrent processor.

The NEEK board yielded very simple modification ability to the design through the touch panel, where the modes, colour pallets can be chanced along with even pausing the displaying of design by simply tapping the touch screen. Figure-17 and Figure-18 show coloured iteration of points out M-set implemented by embedded concurrent processor displayed on both LCD and VGA port respectively.

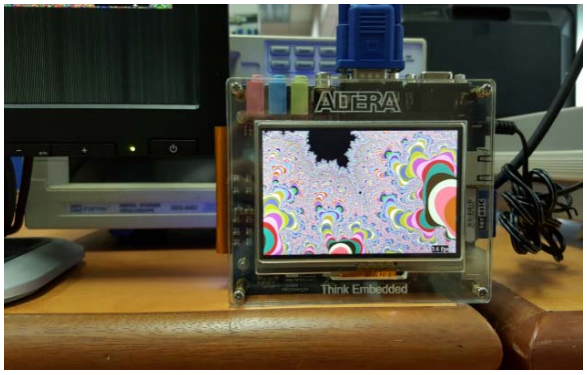

Figure-17. Coloured iteration of points out m-set implemented by embedded concurrent processor displayed on LCD.

Figure-18. Coloured iteration of points out m-set implemented by embedded concurrent processor displayed on VGA port.

To determine the amount of efforts spent on each pixel to make a determination if it's a part of the M-set or not, this really depends on the maximum number of iteration. This pixel calculation is based on the next code:

```
inline int int_mandelbrot(long long cr, long long ci,
int max_iter)
{
long long xsqr=0, ysqr=0, x=0, y=0;
int iter=0;
ci = ci<<28;
cr = cr<<28;
while( ((xsqr + ysqr) < 0x0400000000000000LL) &&
(iter < max_iter) )
{
xsqr = x * x;
ysqr = y * y;
y = ((2 * x * y) + ci) >> 28;
x = (xsqr - ysqr + cr) >> 28;
iter++;
return(iter);
```

The loop in the previous code will proceed till the number of iteration reaches the value of X2+Y2. This function will be executed for each pixel and since the screen has the resolution of 800*480 this means that the function will be executed for 384000 times which is time consuming and a challenge for any computing system.

It's very important to point out that the efficient capacity of the utilised board, through highlighting the amount of the used logic elements and compare it with that available on the board. Table-2 shows that despite the design complexity where large amount of logic elements are demanded, the NEEK board was adequate to yield the design with the needed resources.

Table-2. FPGA resources consumption.

	Utilised logic elements	Available logic elements	Percentage
Total logic elements	21,359	24,624	87%
Total combinational functions	16,527	24,627	67%
Dedicated logic register	14,649	24,624	59%
Total registers		14811	
Total pins	152	216	70%
Total memory bits	251176	608,256	41%
Embedded multiplier 9-bit element	8	132	6%
Total PLLs	1	4	25%

The enhanced results were significant as the data was updated every frame, and this is an increase in the number of frames/sec compared to that in one processor which was updated every five frames. The enhancement was made in the time needed to calculate the function, where the reached frequency is 1GHs the speed improved by 20 times in rendering an image compared to that in one processor case. Table-3 summarises the achieved enhancements by making a comparison between the obtained results from the enhanced embedded concurrent processor and the results from the single processor.

Table-3. Result Comparison between enhanced embedded concurrent processor vs. single processor.

	Enhanced embedded processor	Single processor
Frame update	Every 1 frame	Every 5 frames
Frequency	1 GHz	
Speed	Fast speed (20 times faster in image render)	low speed (slow image render)
Image quality	Clear, smooth with multi-coloured image	Clear, only two coloured image)

CONCLUSIONS

The trend towards embedded multiprocessors is rising especially for application that combines the need for programmability and performance. Selecting the FPGA was after the comparison made with other boards, where its reconfigurable and partial reconfigurable ability and the real parallelism handed along with all the supporting tools like the Qsys all had the upper hand in this design. There are lots of features within the Altera® Nios II Embedded Evaluation Kit, Cyclone III Edition have been reined in to implement this work suchlike the LCD touch screen beside the VGA port supported on this board. In concern to the proposed design an important feature is provided by FPGA board which is the ability to deal with mutex that protected the shared peripheral in this design from data corruption addressing the thread safety issue faced in multiprocessing systems. Finally the SBT for Eclipse also supported on this board for debugging, had the final touches in accomplishing the designing stages before testing the systems performance. The proposed design made remarkable improvement in the number of the displayed frames in every second with a frequency that reached 1GHz rendering speed improvement exceeded 20 times compared to that in one processing element. The firm smooth edges of the M-set plot displayed by the attached LCD touch screen and the VGA port made the given result to be clearly observed and analysed.

REFERENCES

[1] Nios I. I. 2006. Hardware Development Tutorial, Altera, December 2009 Altera Corporation Website.

[2] Asokan V. 2007. Designing multiprocessor systems in platform studio. White Paper: Xilinx Platform Studio (XPS). 1-18.

[3] Levy M. and Conte T. M. 2009. Embedded multicore processors and systems. IEEE micro. (3): 7-9.

[4] Sekhar K. R. and Prasanth V.2013. FPGA Implementation of Quad Processor Core Architecture for Concurrent Computing.

[5] Fort B., Capalija D., Vranesic Z. G.and Brown, S. D. 2006. A multithreaded soft processor for SoPC area reduction. In Field-Programmable Custom Computing Machines, 2006. FCCM'06. 14th Annual IEEE Symposium on (pp. 131-142). IEEE.

[6] Weber J. M.and Chin M. J. 2006. May. Using FPGAs with embedded processors for complete hardware and software systems. In: Beam Instrumentation Workshop 2006(AIP Conference Proceedings Volume 868). 868: 187-192.

[7] Ravindran K., Satish N., Jin Y. and Keutzer K. 2005. An FPGA-based soft multiprocessor system for IPv4 packet forwarding. In: Field Programmable Logic and Applications, 2005. International Conference on (pp. 487-492). IEEE.

[8] Bonnot P., Lemonnier F., Edelin G., Gaillat G., Ruch O. and Gauget P. 2008. Definition and SIMD implementation of a multi-processing architecture approach on FPGA. In: Proceedings of the conference on Design, automation and test in Europe (pp. 610-615). ACM.

[9] Tumeo A., Monchiero M., Palermo G., Ferrandi F. and Sciuto D. 2007, March. A design kit for a fully working shared memory multiprocessor on FPGA. In: Proceedings of the 17th ACM Great Lakes symposium on VLSI (pp. 219-222). ACM.

[10] Kuon I.and Rose J. 2007. Measuring the gap between FPGAs and ASICs. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on. 26(2): 203-215.

[11] Zilic Z., Lemieux G., Loveless, K., Brown S. and Vranesic Z. 1995. Designing for High Speed-Performance in CPLDs and FPGAs. In The 3rd Canadian Workshop on Field-Programmable Devices (FPD'95) (pp. 108-113).

[12] Brown S.and Rose J.1996. FPGA and CPLD architectures: A tutorial. IEEE design and test of computers. (2): 42-57.

[13] Kumar N. S., Saravanan M. and Jeevananthan S.2011. Microprocessors and Microcontrollers. Oxford University Press, Inc.

[14] Karam L. J., AlKamal I., Gatherer A., Frantz G. A., Anderson D. V. and Evans B. L. 2009. Trends in multicore DSP platforms. Signal Processing Magazine, IEEE. 26(6): 38-49.

[15] He J., Chen W., Chen G., Zheng W., Tang Z. and Ye H. 2011. OpenMDSP: Extending OpenMP to Program Multi-Core DSP. In: Parallel Architectures and Compilation Techniques (PACT), 2011 International Conference on (pp. 288-297). IEEE.Marković D. 2012. FPGA vs. ASIC.

[16] Ricks K. G. and Wells B. E. 1998. An Analysis of an Improved Bus-based Multiprocessor Architecture.

[17] https://libreplanet.org/wiki/Group:Hardware/Single_Board_ Computers.

[18] Altera Embedded Processor Solutions,http://www.altera.com/technology/embedded/emb-index.html.