Scientific Journal of Impact Factor (SJIF): 4.72

e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 4, Issue 11, November -2017

Effect of Convergence on Hardware Design Flow of Digital Signal Processing

Laxman Perika

Asst. Professor, Dept. of ECE , Jagruti Institute of Engineering & Technology

Abstract:- *Design of real-time development, memory and processor in digital signal processing systems for decades have been fulfill on GPP. Various efforts to improve the performance of these systems resulted in the use of specific digital signal processing devices like DSP processors of electronics design - Application Specific Integrated Circuits. The advance of RAM-based Field Programmable Gate Arrays has changed the DSP design flow. Software algorithmic designers can now take their DSP algorithms right from inception to hardware implementation, thanks to the increasing number of C/C++ hardware design flow. This has led to a demand in the industry for graduates with good skills in both electrical engineering and electronics engineering. This paper evaluates the impact of technology on DSP-based designs, hardware design languages, and how graduate/undergraduate courses have changed to suit this transition.*

Keywords: *Digital Signal Processing, Application Specific Integrated Circuits, Field Programmable Gate Array, Augmented C/C++ design-flow*

I. INTRODUCTION

FAST CHANGING STATE-OF-THE-ART

Technologies have had a great impact on electronic system design issues. The 1970s was the decade of semiconductors, which enabled the digital generation. This was followed by dynamic memory (DRAM) in the 1980s and then the microprocessor era in the 1990s, which made the word "Megahertz (MHz)" a common term in our everyday language. The new millennium has made Digital Signal Processing the technology of focus with an expected exponential growth. Digital Signal Processing (otherwise referred to as DSP) is the branch of electronics that involves the representation, processing and manipulation of real-world signals in digital form. DSP as a technology has been available to engineers for over two decades and was initially used in application areas where cost and performance were the ultimate goals. This special technology can be found in almost all aspects of our daily lives; including but not limited to security systems (home or industrial), medical systems, telecommunications, consumer electronics, defense and aerospace. DSP systems for the past decade have been implemented using the ubiquitous general-purpose micro-processors. The increasing demand by consumers of such systems for high-speed, resulted in the use of dedicated Digital Signal Processors also referred to as DSPs; a special type of general-purpose processor optimized for signal processing algorithms. The processing power of general-purpose processors has increased over the past years while their price de-creases; thanks to Moore's law. This trend of increasing the clock-rate of the processor has two major drawbacks:

1. Inability to meet the performance demand of today's electronic systems, as depicted in Figure 1.

2. High power consumption, unsuitable for battery powered systems.

As the clock-rate of general purpose processor in-creases the processing requirement of DSP systems increases as well. This trend has overstretched the performance of DSPs to a point that they can no longer support the demand of today's systems – Hook's law applied to DSP. Power dissipation is important in almost all DSP-based consumer electronic devices; hence the high-speed, power-hungry DSPs become unattractive. For hardware acceleration and low power consumption, DSP designers had no option, other than the then heavyweight champion of electronic design, ASIC, sacrificing flexibility for performance.

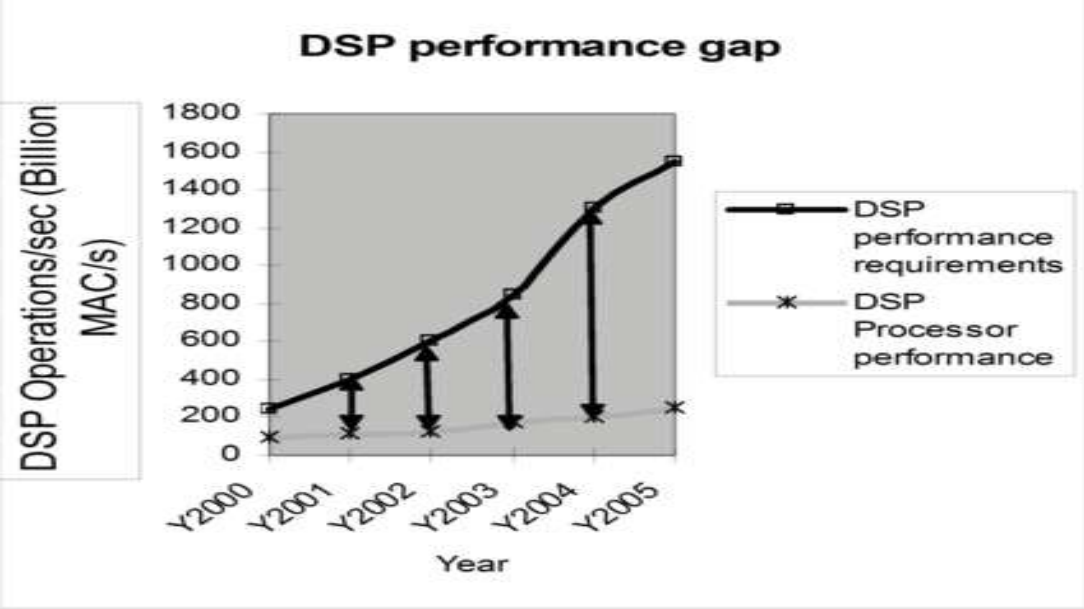


Figure 1: DSP Performance Requirements vs. General-Purpose DSP Processor Performance. Top-Down DSP Design Flow to Silicon Implementation.

Dedicated Digital Signal Processors and Application Specific Integrated Circuits have high disparities in term of flexibility and performance. Algorithmic designers in the DSP world have long been software engineers with little or no knowledge in hardware design, but rather software programming and mostly high-level mathematics needed for DSP. The only way software engineers could have their design implemented in hardware, for increased performance, was to pass their algorithms and specifications over to their hardware colleagues. These hardware engineers are much interested in the "how" of a technological phenomenon rather than the "why" and hence do not have the deep understanding of high-level mathematics needed in DSP. This makes the hardware implementation from algorithmic level, through the register transfer level (RTL) to its final or silicon implementation error prone. A hybrid reconfigurable computing architecture like Field Programmable Gate Array (FPGA) offers the necessary computing resources needed for today's DSP. There is an increasing need for DSPs to become more reconfigurable to meet future needs. "The responses also make us believe that programmable logic is likely to dominate the digital design domain over the next decade" Kevin Morris in a survey conducted by Xilinx in September 2004. The latest Dataquest reports show that 40% of ASIC designs in 2003 had less than 1 million gates, running at 500 MHz or less. Current FPGAs from key players like Xilinx and Altera have as many as 8 million gates, running at 500MHz or more. Unless power usage is very critical or production volumes are extraordinary, it is becoming increasingly hard to justify the use of ASIC over FPGA.

Technological advancements in Field Programmable Gate Arrays (FPGAs) open new paths for DSP design engineers. FPGAs maintain the advantages of the high specificity of the ASIC while avoiding the high development costs and inability to make design modifications after production. FPGAs add adaptability and design flexibility with device utilization conserving both onboard system power and space, which is often not the case with DSP chips. When the design demands more than 100 MIPS, time-to-market is critical or design adaptability is crucial, the use of reconfigurable architecture becomes inevitable.

New implementation platforms call for new design processes for efficiency and accuracy. As the time-to-market has become very critical in the design and implementation of DSPs, the conventional method of implementing DSP designs on ASICs is highly unacceptable. Designing with multimillion-gate FP-GA for time constraint systems, therefore require a new breed of engineers with unique DSP skills. This is in line with the optimistic view on the impact of new technologies on society and employment pattern. To the optimists, new technologies come with new jobs that replace existing ones and new skill requirements.

II. DSP DESIGN OPTIONS

Digital Signal Processing can be implemented on four major platforms: the use of conventional micro-processor, dedicated DSP chip, reconfigurable Field Programmable Gate Array and dedicated Application Specific Integrated Circuit. Clock speed or instruction cycle time is not the only contributing factor for the evaluation of the total performance of a particular DSP platform. Rather the bus architecture, in-put/output (IO) hardware and the software algorithm upon which the implementation is based must be considered. Table 1 gives a comparison between these platforms in terms of flexibility and performance. DSP design typically starts with the system specification or algorithmic design, which is followed by a decision on a platform and finally the actual implementation. Each platform requires a specific design route. Knowledge of the final implementation architecture is required for a successful and highly optimized DSP design.

Designing DSP-based systems with a microprocessor or dedicated DSP chip is cheaper and much easier than the other methods, see figure 2. This starts with the design of the actual algorithm followed by simulation and verification of the algorithm with a high-level programming language. It is then followed by a translation of the algorithm into code acceptable to the final DSP platform, usually C or assembler. The functionality of the DSP code is verified with the high-level program as reference. The design is then ported to the final hardware for final debugging and verification.

	Flexibility		**Performance**		
	Programmability	**Reconfigurability**	**Area utilization**	**Power consumption**	**Computational throughput**
ASIC[1]	Low	Low	Low	Low	High
DSP	Medium	Medium	High	High	Low
MicroP (P)	Medium	High	High	High	Low
FPGA	Medium	High	Low	Medium	Medium

Table 1: Signal Processing Implementation Platform Comparison

Source : Jing Ma, *Signal and Image Processing Via Reconfigurable Computing* , (University of New Orleans: Workshop on Information and Systems Technology - May 2003)

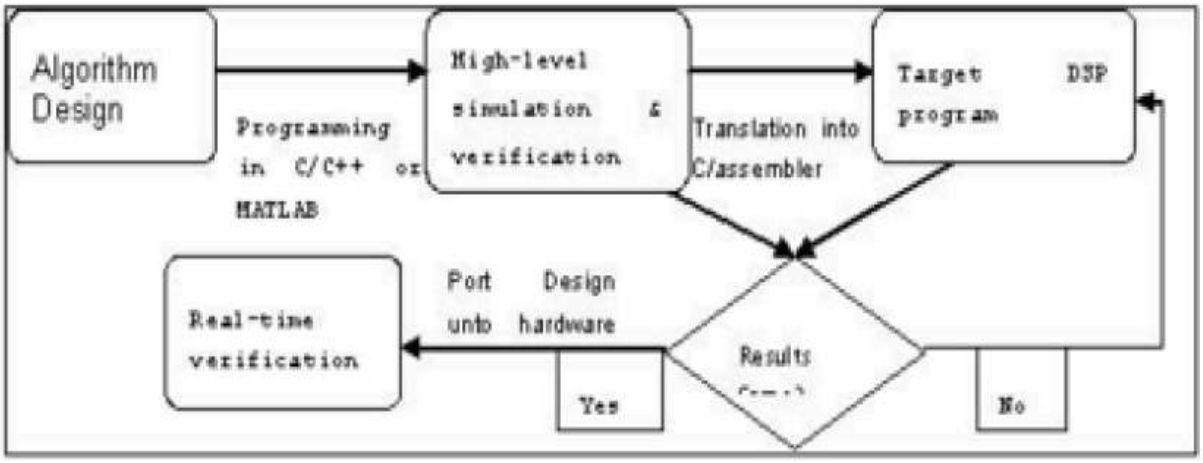


Figure 2: Microprocessor/DSP chip based DSP Design-flow

This design flow is very short and prone to fewer errors, as the same person who designs the algorithm does the hardware implementation. The performance of the resulting system in term of speed makes them unattractive.

Designing for Application Specific Integrated Circuits (ASIC) comes in two disjoint levels, see figure 3. A system or algorithmic designer handles the first level while a hardware engineer handles the second level. The algorithmic designer describes and simulates the implementation in a high-level programming language like C/C++ or MATLAB. This design is then sent to a hardware designer who will then convert the specification into Register Transfer logic (RTL) model, by converting all floating-point operation into fixed-point. The model is then simulated for similarity with the high-level design. This design flow is not only error prone but time consuming as well.

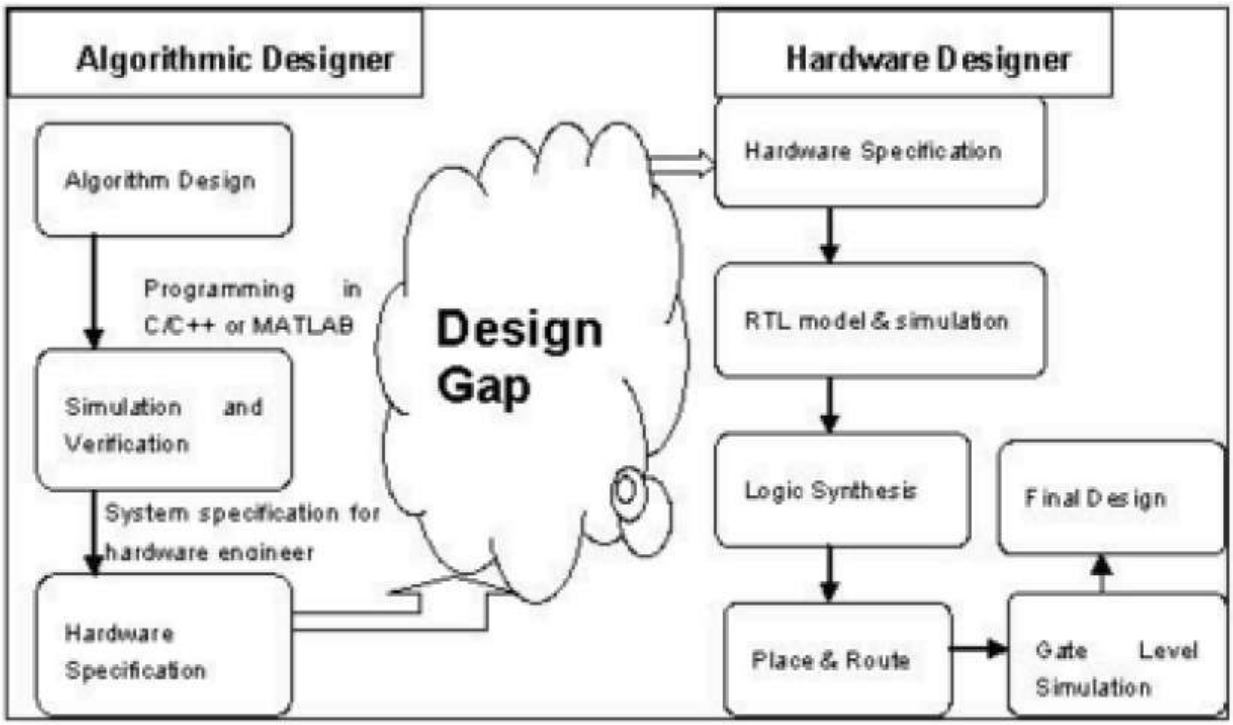


Figure 3: A Typical Application Specific Integrated Circuit (ASIC) Design Flow

Many hardware designers have taken to FPGA computing for their DSP-based designs for the following important reasons , [Xilinx]:

1. Fast time-to-market
2. Non-recurring engineering cost
3. Efficient use of time and
4. Satisfying human wants.

The low price/performance ratio of today's high-end FPGAs makes them very attractive for the development of DSP systems. Until recently, the FPGA design solutions have been unavailable to software or algorithmic designers with little or no skills in computer hardware.

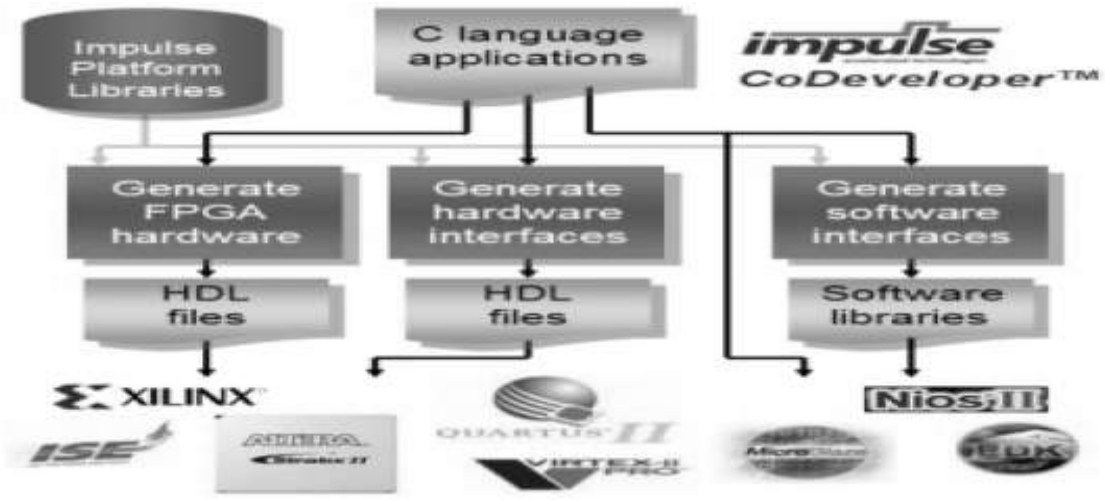
Many design tools have floored the EDA market, taking advantage of the short FPGA design flow. Tools based on high-level languages like C/C++ and MATLAB very common to DSP designers, have been available on the market recently. This has helped DSP algorithmic designers to take full advantage of FPGA solutions without a steep learning curve. The C

programming language has been the language of choice for most DSP applications, for its portability and compactness. C as a typical language for general-purpose computers lacks the specialized features like parallelism, wires, clocking and delays, required in hardware design. To extend the capabilities of C/C++ for hardware designs, many tool developers have turned to augmented C/C++ ; tools which extend the capabilities of C for hardware purposes. Typical examples are SystemC, SystemVerilog, Impulse-C, Handel-C and AccelChip (based on MATLAB).

Impulse-C is designed to simplify the development of highly parallel, processing-intensive applications, including applications that require a mix of hardware and software processing resources, or applications in which specific processes must be interfaced to other hardware and/or software components to form a complete system. The benefit of using Impulse-C is that it gives you more opportunities to experiment with alternative algorithmic approaches, a faster path to a working prototype and greater opportunities to improve sys-tem-level performance through iterative design methods.

Another big player in the C/C++ augmentation for ASIC and FPGA design capture, simulation and synthesis is Celoxica. Celoxica's Handel-C has special statements to support concepts like clocks, pins, parallelism, interfaces, RAM and ROM. The numerous libraries that come with the language makes it much easier to interact with their RC-series FPGA boards. Designing code with Handel-C is fairly straightforward to any conventional C program-mer. This makes it easier for any software engineer, adopt and use Handel-C for hardware development. Handel-C is seen as a programming language rather than a hardware description language like VHDL and Verilog AccelChip DSP synthesis tool, which is based on MATLAB design flow, has made significant impact in the FPGA-DSP design environment. The AccelChip DSP synthesis tool directly reads in MATLAB (a popular DSP algorithm design tool from The MathsWorks Inc.) models and automatically outputs synthesizable RTL models and simulation test-benches in VHDL or Verilog. This provides DSP designers with a significant reduction in design labour and time, elimination of misinterpretations and costly design rework, automatic verification of the hardware implementation, and the ability of systems engineers and algorithmic developers to perform architectural exploration in the early phases of their development cycle.

The simple and short design flow exhibited by Impulse-C and Handel-C is shown in figure 4. The entire design is described in C; taking advantage of the added hardware support feature as well as the libraries supplied with the EDA tool. A typical ex-ample is the PAL library that comes with Celoxica's DK suite for interacting with devices, external to the Field Programmable Gate Array. The design is then compiled into a Hardware Descriptive Language (HDL) or an Electronic Design Interchange Format (EDIF) suitable for a particular vendor's place and route tool. Pessimists of new technologies describe the emerging short-hardware-design flow for DSP applications as detrimental. They argue that there will be an overall decline in job opportunities and thus, leading to large-scale and possibly permanent unemployment. This is the case, when a single engineer with the aid of automated design tools is able to accomplish the task of two engineers; in this case the algorithmic designer and the hardware designer. The use of EDA tools also results in deskilling, as technology downgrade hardware design engineers.



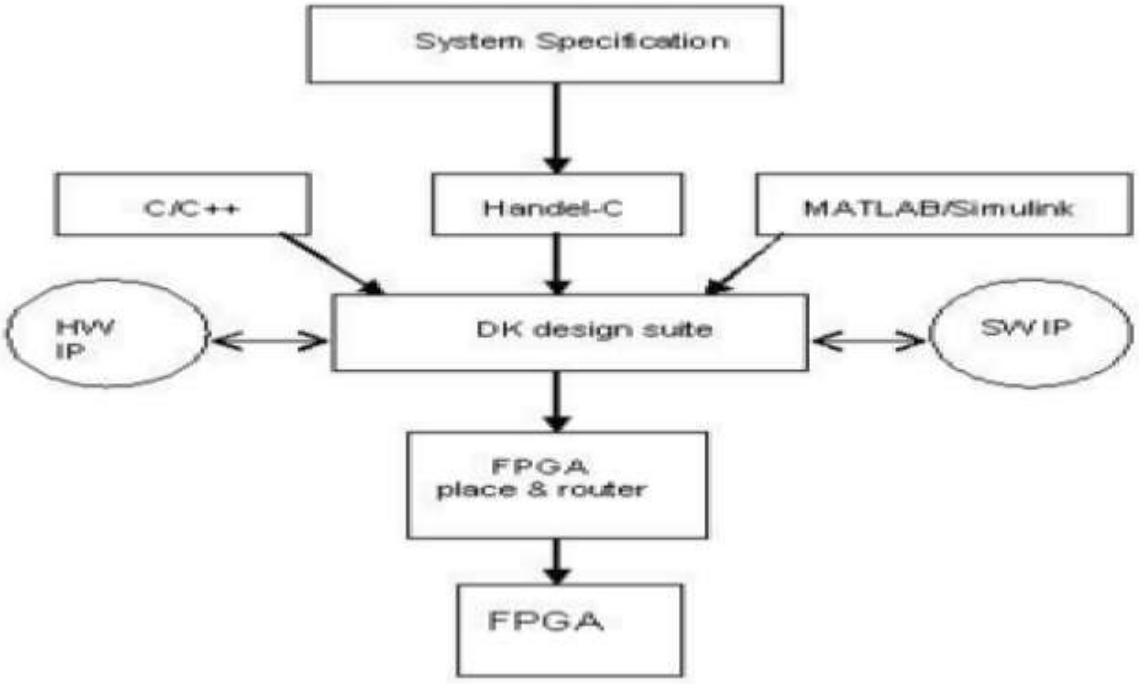


Figure 4: Impulse –C and Handel-C Hardware Design Flow

FPGA-BASED DESIGN REQUIREMENTS

Designing FPGA based systems, using Hardware Descriptive Languages (HDLs) like VHDL and Verilog is very similar to building hardware or arrays of gates as compared to programming in software. Thus the designer or developer is expected to focus on using hardware components rather than software constructs, to avoid using too many gates and highly inefficient chip and logic layouts. It is becoming increasingly hard to justify an inefficient hardware implementation of an algorithm over the same algorithm written in C and running on today's high-speed microprocessors. Thus simply implementing an algorithm in hardware does not necessary make them faster than their software implementation.

Finding a way to help FPGA designers skip the long and time consuming HDL-based design without missing a beat in productivity is a key to gaining acceptance. Even though many high-level languages have been proposed as system design languages (C/C++, Java, and MATLAB), most of these languages would produce very large logic designs, if the designer has very little knowledge of the hardware architecture. Most of these soft-ware/hardware tool developers argue that in this era where chips are very cheap, hardware efficiency is no longer important. Unfortunately, this is not always true, as most FPGA designs would have to meet timing, power and architectural constraints. There-fore, a uniquely skilled DSP/logic designer is generally required to construct an FPGA design.

Design and implementation of high-speed and low-powered DSPs to meet the short time-to-market requirements of today's increasing consumer electronics require a broad knowledge in areas tradition-ally not covered in a single discipline. These areas include Computer Science, Electrical and Electronic Engineering, Business, Social Science, Mathematics and others. This has made it very difficult over the past years to train students in a single discipline to effectively design and implement high-speed Digital Signal Processing systems.

Attempts by various academic institutions in training students for the always-changing DSP industry resulted in a number of disciplines with no clear bounds between them. This means students involved in such disciplines had much more workload. Computer Science is concerned with the theoretical and algorithmic aspects of computing. Electrical Engineering is concerned with the physical aspect of electronics whiles Software Engineering is an engineering discipline that is

concerned with all aspects of software production from early stages of system specification to maintaining the system after it has gone into use. Thus Digital Signal Processing designs, call for a cross-pollinated engineer with a multidisciplinary design skills. Such an engineering discipline will embody the science and technology of design, construction, implementation, and maintenance of software and hardware components of modern computing systems and computer-con-trolled equipments.

The Computer Society of the Institute for Electric-al and Electronic Engineers (IEEE-CS) and the Association for Computing Machinery (ACM) by means of drawing a clear boundary between the related disciplines (Computer Science and Electrical Engineering) established a Joint Task Force on Computing Curricula to undertake a major review of the curriculum guidelines for undergraduate programs in computing. The three-year degree program that resulted from this Task Force's report (Computing Curricula for Computer Engineering - CCCE) is computer engineering, which lies between computer science and electrical engineering.

Computer engineering is solidly grounded in the theories and principles of computing, mathematics, science, and engineering and it applies these theories and principles to solve technical problems through the design of computing hardware, software, net-work, and processes. The CCCE report published in 2004(CCCE-04), requires all computer engineers to exhibit the following characteristics:

- Professionalism – as the public has entrusted in engineers a level of responsibility based on the systems they design, engineers have to exercise the utmost conscientiousness when designing.
- Ability to design – to seek and exploit new and improved products.
- Breadth of Knowledge – that enables the engineer to span the entire fields of study.

Engineers in other fields like electrical and electronic engineering are also expected to possess the same characteristics, with the exception of the breadth of knowledge require by a computer engineer. To clarify the similarity in workload of a computer-engineering student and an electronic engineering student, we evaluate on a common computer related course like computer organization and architecture. Typically electrical engineers spend time in resolving complex communication and synchronization issues when dealing with concepts of parallelism using array of processors. A computer engineer resolving a similar issue for the purpose of DSP will use the widespread Field Programmable Gate Array (FPGA) without having to worry about complex communication and synchronization issues. FPGAs, which were originally used primarily for glue logic, now have distributed memory and multiplication re-sources that are capable of handling computational tasks directly. An electrical engineer will typically program in Hardware Descriptive Language (HDL) like VHDL (Very high-speed integrated circuit Hardware Descriptive Language), which is much harder as compared to a computer engineer who will program in a high-level system design language like Handel-C. Unlike professions such as law and medicine, engineering generally does not require an advanced degree for employment in the field. As a means to allow engineers and scientists from fields other than computer engineering to take full advantage of new technologies and their design requirements, several universities have started graduate degree course in System-on-Chip (SoC) Design. The Institute for System Level Integration (ISLI) started the first postgraduate program in system-on-chip aimed specifically at system level integration. The emergence of new generation of design technologies and methodologies, like development of Electronic Design Automation (EDA) tools, systems partitioning between software and hardware and design verification have made this postgraduate course very popular.

The SOCWARE design cluster including the Royal Institute of Technology (KTH), Linkoping University and the Lund University all in Sweden have for the past four years been running masters programs in System-on-Chip Design. The key aspects of these courses have been the design methods, architectures and circuit design towards system level integration (SLI). The content of the program is been driven by:

- Interdisciplinary approach – ranging from deep submicron and digital noise issues to formal techniques and system modeling.
- Practical competence in the form of small projects undertaken in worlds-leading hightech companies and hands-on laboratory sessions for industrial practice and academic expertise.

Similarly, the Graz University of Technology, Austria in 2002 started a postgraduate programme in SoC Design. Students of this program take courses in concurrent systems, hardware/software co-design and how to model hardware-software systems. Their modules also emphasize on the use of IP cores as well as software libraries, as today's systems-on-silicon contain more and more software. The post- graduate course in Embedded Systems Design at the Advanced Learning and Research Institute, Lugano in Switzerland and the newly designed master's programme in SoC Design at the faculty of engineering, University of Sheffield in England are no different from the above programmers. We are beginning an era where the system designer is the Integrated Circuit (IC) designer, the digital designer, the analogue designer and the system integrator; this therefore calls for graduates who are well prepared for career challenges ahead of them and hence has spark the design of new undergraduate/graduate programmes. Stewart also pointed out the importance of integrating electrical engineering and computer science courses to aid advanced DSP designs in both industrial and educational institutions.

III. CONCLUSION

Digital Signal Processing is widely used in almost all aspects of our daily life, yesterday's wants have become today's needs and this trend in expected to continue throughout this millennium. The in-creasing need to meet human satisfaction coupled with the high competition amongst electronic design companies has triggered the use of FPGAs in the DSP industry. The use of sophisticated EDA tools for FPGA designs give pessimists the impression that new technologies results in deskilling as well as unemployment. Optimists have the view that these technologies create new jobs, which in effect replaces old ones and hence the employment rate remains unchanged. Concerning skill requirement, optimists are of the view that new graduates as well as skilled engineers are required to effectively utilize the cap-abilities of these technologies and thus causing "re-skilling". A third group with mixed views on new technologies has emerged as a result of convergence between the optimists and the pessimist. Long-term optimists or the short-term pessimists believe that new technologies will cause considerable displacement in certain sectors they maintain it should be for short term.

Programming FPGA with high-level languages like C, Java and MATLAB have changed the DSP design flow and also added to the skill requirements of today's DSP development. We have explained in this paper the requirements in today's DSP industry and how several universities have restructured their curriculum to teach courses in DSP designs and hardware architectures, to prepare graduates for the always-changing industries. A typical example is the new curriculum, computer engineering which has emerged as a convergence between computer science and electronic engineering. The software/hardware designs or the software-hardware co-designs have significantly shortened the DSP design flow.

REFERENCES

[1] "Computing Curricula Computer Engineering, Final Report", *Report of the IEEE-CS/ASM Joint Task Force on Computer Engineering*, 2004.

[2] Dick, Chris. "FPGAs: The high-end alternative for DSP applications", *DSP Engineering* Spring 2000. ED Online. "FPGA High-Level Design Methodology Comes Into its Own", June 1999.

[3] Feist, Tom. "What's the right language for DSP system-level Design?" *FPGA and Programmable Logic Journal*, November 2003.

[4] Edwards, John. "DSP Technology gains importance in all major market segments", *Boards and Solutions* March 2003. Frantz, Gene A. "Preparing EE's for the 21 st Century with a 20th Century Education", *Signal Processing Education Workshop* – 2000.

[5] Wood, Sally L. and Chris Dick. "Concepts of Parallelism in an Introductory Computer Architecture Course with FPGA Laboratories", *ASEE/IEEE Frontiers in Education Conference* – October 2004.

[6] Young, Duncan. "FPGAs' impact on next generation sensor digital signal processing", *Military Technology Insider* June 2004.

