McDonald et al., J Comput Eng Inf Technol 2016, S1
http://dx.doi.org/10.4172/2324-9307.S1-e101

Journal of Computer Engineering & Information Technology

Editorial

A SCITECHNOL JOURNAL

Current Trends in Applications for Software/Hardware Integration

McDonald[1]*, Andel[2], Mahoney[3] and Russ[4]

In today's modern computing world, software and hardware work together to produce the wide range of advanced system functions that support industry, science, and commerce. Historically, software and hardware have been seen to be distinct entities with clear boundary lines, depicted in Figure 1. Traditional software consists of code in the form of high-level programming language commands or low-level assembly instructions, targeted specifically at a general purpose processor. Traditional hardware has evolved in the form of general central processing units (CPUs) and microprocessor chip families such as Intel and Motorola consistent with the original vision of the von Neumann architecture. In its historic use, compilers and interpreters that guaranteed compliance with underlying architecture guaranteed software/hardware integration. General purpose processor (GPP) architecture and code have thus formed the bulk of our modern application base.

The notion of software/hardware integration in modern parlance often alludes to the realm of embedded systems where microprocessor design crosscuts between physical hardware and operating system software that manages it. Cyber physical systems (CPS) extend this notion further to encompass systems that have real-time collaborating computational elements that control or are informed by physical devices and sensors [1]. At a higher level, software/hardware integration might involve the collaborative effect of networking appliances (the hardware) with the underlying control applications (the software) that must work together to produce client-server and multi-tier infrastructure. At a lower level, software/hardware integration can involve system-on-chip (SoC) devices that allow high configurability of both reprogrammable logic and functionality, where devices like field-programmable gate arrays (FPGAs) constitute the hardware and logic programs constitute the software [2-5].

As Figure 2 illustrates, advancements in hardware technology have produced a revolution in computing power by offering task-specific processing components that can be leveraged by some underlying piece of software. For example, graphics cards and graphics processing units (GPUs) are nothing more than a specialized hardware for multimedia display and there are always associated software libraries that must be used in order to take advantage of the specific hardware features offered. Application specific integrated circuits (ASICs) have also become standard systems building blocks that provide specific functions leveraged by other hardware and software in a systems design. In many cases, physical hardware in the form of either ASICs or FPGAs can replace a slower software/GPP based solution as in the case of cryptographic ciphers like RSA and AES or with network security monitors [6]. FPGAs themselves can contain software-based or virtual processors, allowing agile configuration of the same piece of hardware to have almost unlimited programmability. With the advent of reprogrammable and specialized device technology, the line between software and hardware itself has become blurred [7].

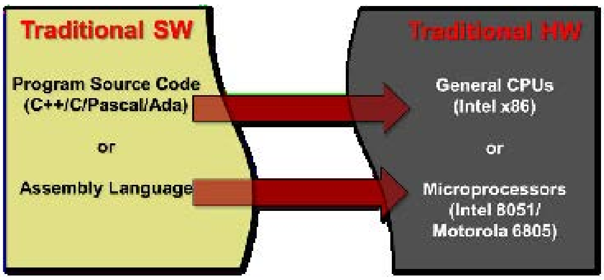

Figure 1: Traditional Notion of Software/Hardware Integration.

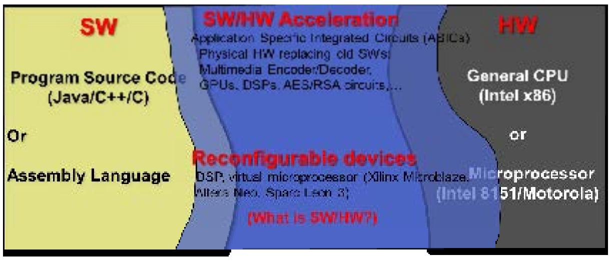
SW
Program Source Code
(Java/C++/C)
Or
Assembly Language
SW/HW Acceleration
Application Specific Integrated Circuits (ASICs)
Physical HW replacing old SWs:
Multimedia Encoder/Decoder,
GPUs, DSPs, AES/RSA circuits,...
Reconfigurable devices
DSP, virtual microprocessor (Xilinx Microblaze,
Altera Neo, Sparc Leon 3)
(What is SW/HW?)
HW
General CPU
(Intel x86)
or
Microprocessor
(Intel 8151/Motorola)

Figure 2: Modern Notion of Software/Hardware Integration.

This special issue on current trends in applications for software/hardware integration highlights recent advancements, novel usage, and future concepts for how applications are constructed to realize the full extent of integration between these two worlds. For example, software applications that utilize specialized processors through custom or proprietary libraries already embody this concept. Likewise, software that utilizes FPGA-based logic functions to improve performance of otherwise time-intensive GPP code is in essence married to the hardware at a deeper level than a traditional GPP [8]. Without the underlying hardware present, the software would not function correctly or would not perform at its most efficient speed.

The issue will look at advancements in tools, techniques, frameworks [9], methods, and practice for the prototyping, co-design, test, and verification of software/hardware integrated applications. Applications of full SW/HW integration can offer potential benefits to performance [10-13], security, agility, and configurability while also creating more complicated development and design issues. The special issue will also highlight advancements in languages [14,15], development environments, and industry support for this new paradigm of computing. Of particular interest are case studies of both

*Corresponding author: J Todd McDonald, Professor of Computer Science, University of South Alabama, USA, Tel: (251) 460-7555; Fax: (251) 460-7274; E-mail: jtmcdonald@southalabama.edu, tandel@southalabama.edu

Received: August 10, 2015 Accepted: August 12, 2015 Published: January 19, 2016

successful and failed approaches, delineation of problems and issues [16], and potential solutions for SW/HW integrated applications.

References

1. Luo Y, Chakrabarty K, Ho T (2015) Hardware/Software Co-Design and Optimization for Cyberphysical Integration in Digital Microfluidic Biochips. Springer International Publishing, Switzerland.
2. Huang T and Mar P (2011) Hardware/Software integration: Closing the gap. EE Times: Connecting the Global Electronics Community.
3. Vuleti M, Pozzi L, Ienne P (2005) Configurable Computing: Fabrics and Systems Seamless Hardware- Software Integration in Reconfigurable Computing Systems. IEEE Design & Test of Computers 22: 102-113.
4. Vercauteren S and Lin B (1997) Hardware/Software Communication and System Integration for Embedded Architectures. Design Automation of Embedded Systems 2: 359-382.
5. Garcia P, Compton K, Schulte M, Blem E and Fu W (2006) An Overview of Reconfigurable Hardware in Embedded Systems. EURASIP J Embedded Syst 1:1–19.
6. Chen H, Chen Y, Summerville DH (2011) A Survey on the Application of FPGAs for Network Infrastructure Security. IEEE Communications Surveys Tutorials, 13: 541 –561.
7. Vahid F (2007) It's Time to Stop Calling Circuits "Hardware". IEEE Computer Magazine 40: 106-108.
8. Arora D, Ravi S, Raghunathan A, Jha NK (2005) Secure embedded processing through hardware-assisted run-time monitoring. Proceedings of Design, Automation and Test in Europe 1: 178–183.
9. Abu BAI, Atan R, Yaakob R (2011) Middleware framework for integration of heterogeneous hardware and software. Proceedings of the 2011 IEEE 5th Malaysian Conference in Software Engineering (MySEC) 66 – 69.
10. Guinde NB and Lohani RB (2011) FPGA Based Approach for Signature Based Antivirus Applications. In Proceedings of the International Conference-Workshop on Emerging Trends in Technology (ICWET'11) 1262–1263.
11. Li S, Torresen J, Soraasen O (2003) Exploiting Reconfigurable Hardware for Network Security. In 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 2003) 292-293.
12. Lockwood JW, Moscola J, Kulig M, Reddick D, Brooks T (2003) Internet Worm and Virus Protection in Dynamically Reconfigurable Hardware. In Military and Aerospace Programmable Logic Device (MAPLD) 10.
13. Mück TR and Fröhlich AA (2014) A metaprogrammed C++ framework for hardware/software component integration and communication. J Sys Arch 60: 816–827.
14. Hopf J, Mawson L and Kearney D (2003) Specification and integration of software and reconfigurable hardware using Hardware Join Java. Proceedings of the 2003 IEEE International Conference on Field-Programmable Technology (FPT), 379 – 382.
15. Edwards SA (2005) SHIM: A Language for Hardware/Software Integration. Synchronous Languages, Applications, and Programming.
16. Schirrmeister F, McNamara M, Melling L, Bhatnagar N (2012) Debugging at the hardware/software interface. Embedded Computing Design.

This article is published in the special issue **Current Trends in Applications for Software/Hardware Integration** and has been edited by Dr. J. Todd McDonald, University of South Alabama, USA

Author Affiliation

[1]Professor of Computer Science, School of Computing, University of South Alabama, USA
[2]Professor of Computer Science, School of Computing, University of South Alabama, USA
[3]Associate Professor of Information Assurance, Acting Director, School of Interdisciplinary Informatics, University of Nebraska at Omaha, USA
[4]Associate Professor and Interim Department Chair of Computer Engineering, College of Engineering, University of South Alabama, USA