FPGA-Based Accelerator Development for Non-Engineers

David Uliana
National Instruments Corporation
Austin, Texas 78759, USA
Email: david.uliana@ni.com

Peter Athanas and Krzysztof Kepa
Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061, USA
Email: {athanas, kepa}@vt.edu

***Abstract*—In todays world of big-data computing, access to massive, complex data sets has reached an unprecedented level, and the task of intelligently processing such data into useful information has become a growing concern to the high-performance computing community. However, domain experts, who are the brains behind this processing, typically lack the skills required to build FPGA-based hardware accelerators ideal for their applications, as traditional development flows targeting such hardware require digital design expertise. This work proposes a usable, end-to-end accelerator development methodology that attempts to bridge this gap between domain-experts and the vast computational capacity of FPGA-based heterogeneous platforms. To accomplish this, a development flow was assembled, targeting the Convey Hybrid-Core HC-1 heterogeneous platform and utilizing an existing graphical design environment for design entry. The efficacy of the flow in extending FPGA-based acceleration to non-engineers in the life sciences was informally tested at an NSF-funded summer workshop, organized and hosted by a bioinformatics organization at a particular university. A group of five life-science-focused, non-engineer participants made significant modifications to a bare-bones Smith-Waterman accelerator, extending its functionality and improving performance.**

I. Introduction

From measurements of physical phenomena at CERN's Large Hadron Collider (LHC) to the billions of nucleotides in the human body's DNA, we have access to an unprecedented amount of stored data. It saturates the world in which we live, and while at one time the production of sufficient data was a primary challenge, the problem has now shifted to its consumption and intelligent processing at reasonable rates. The task of compiling such massive data sets into useful information falls to *domain-experts*, scientists with domain-specific knowledge of algorithms needed for such analysis. Due to the size of the data sets, these algorithms often have extensive computational requirements, and such experts resort to the use of large, data-center class computing systems for algorithm acceleration. The rigid, coarse-grain architecture of such systems, however, is intended for high-performance over a *general mix* of problems, and cannot take advantage of many application-specific optimizations. FPGA-based, heterogeneous platforms, such as the Convey Hybrid-Core series, provide the fine-grain flexibility needed to enable optimizations specific to an application's architecture.

Traditional tools targeting FPGA-based platforms are characterized by several shortcomings, the foremost being the prerequisite digital design experience and hardware description language (HDL) skills required by almost all FPGA development flows. The majority of non-engineers lack such expertise, resulting in a wide gap between domain experts' algorithm knowledge and the vast computational capacity of FPGAs. Furthermore, most traditional tool-flows exhibit limited support for high-level IP reuse, often resulting in redundant effort in the design phase. The interactivity—rapid feedback leading to design changes—of these tools is also often very limited, lengthening the time to solution by constraining the number of meaningful design iterations explored and verified per day. All of these problems raise the already high barrier to entry for application development targeting FPGA-based platforms.

While many non-traditional, text-based syntaxes have emerged in an attempt to bridge the gap between domain experts and FPGA-based, heterogeneous acceleration, they have yet to consistently produce efficient implementations without designer input of a nature requiring hardware design expertise (e.g. compiler directives). Some existing graphical tools present a more intuitive approach to hardware design, constraining the user to describe functionality on a highly-abstracted, inherently concurrent graphical canvas. These syntaxes are more usable by non-engineer domain experts without digital hardware design experience, and ease the translation from a high-level functional description to the hardware-level. Through the use of such an environment in the assembly of an end-to-end development environment targeting FPGA-based, heterogeneous platform, this work attempts to address the difficulties traditional FPGA tool flows present to non-engineers. A development flow was constructed, utilizing the National Instruments LabVIEW FPGA environment as a front-end to target the Convey HC-1 platform, including integration of the HC-1 system simulation framework into LabVIEW.

To demonstrate the efficacy of the two flows, an informal evaluation of its usability was conducted during a life-sciences summer workshop. A small group of students and professionals were given a bare-bones, systolic array-based implementation of the Smith-Waterman sequence alignment algorithm, which made heavy use of the interface abstractions provided with the flow. The group members were tasked with extending the functionally and performance of the implementation using one of the two flows, with limited involvement of technical experts.

This paper is organized as follows. Section II discusses the growing concern of big-data science to the field of HPC, and how heterogeneous computing with FPGAs can address these concerns. Included is an overview the current state of development tools targeting these platforms, with a focus on LabVIEW and the Convey HC-1. Section III presents *ConVI*,

978-1-4799-5944-0/14/$31.00

a seamless, end-to-end development flow targeting the Convey HC-1 heterogeneous HPC platform with LabVIEW FPGA as the design front-end. Last, Section IV contains the qualitative and quantitative results of this experiment.

II. Background

The current era is one characterized by an abundance of raw facts, as the size and availability of valuable data sets is rapidly increasing. This trend is reinforced by the existence of projects like the Large Hadron Collider (LHC), which contains 150 million sensors and will produce about 25 petabytes of data in 2013, even after discarding over 99.99% of the sensor output [4], [12], [28]. Another example—more pertinent to this work—is the construction of the genome of a single human individual, which is comprised of over three billion DNA base pairs. Producing such a genome using output from top-of-the-line next-generation DNA sequencers involves the complex process of assembling billions of 100-200 base-pair reads into a single sequence [23]. This is especially concerning given the expectation that, in the near future, next-generation sequencing machines will produce these reads in a matter of hours [14]. This flood of information necessitates exceptional computing performance in order to process such data sets within tolerable times.

At the forefront of big-data analysis efforts are domain experts—the brains behind extracting useful information from vast data sets. These include scientists who interpret the LHC sensor output and bioinformaticians who explore genome hypotheses by analyzing large quantities of genome data [16]. Such domain experts may have significant programming skill; however, their productivity is limited by the processing capabilities of available computing resources. For example, the architecture of data-center class computing platforms is designed for speed over a general mix of problems, and efficiently applying such platforms to domain-specific data structures and algorithms such as DNA/protein sequence alignment can be very difficult [24]. Heterogeneous computing machines, such as the Convey Hybrid-Core (HC) servers have potential to address this problem.

Heterogeneous computing can be defined as the application of diverse computing resources to the acceleration of computationally intense tasks with diverse requirements [11]. Moving a computing task from general purpose processing resource to a resource specifically designed for that task or class of tasks has potential to increase execution efficiency, specifically in terms of time and space/energy savings [13]. These benefits have resulted in the widespread use of domain- and application-specific computing resources, including graphics processing units (GPUs) and application-specific integrated circuits (ASICs), for HPC problems. When properly programmed, FPGAs outperform multicore CPUs and GPUs for many applications, due to support for non-standard data types and massive, fine-grain parallelism, features that dominate many bioinformatics applications.

A. Convey Hybrid-Core

Convey Computers Hybrid-Core (HC) machines are examples of FPGA-based, heterogeneous computing platforms [8]. These systems, the architecture of which is shown in Figure 1, provide application acceleration through the tight coupling of a general purpose processor and an FPGA-based application accelerator. The work presented in this paper makes heavy use of the Convey HC-1 Computer, a 2U chassis server partitioned into a Intel Xeon-based host board with up to 128 GB of memory, and a co-processor accelerator board, the focus of which is four user-programmable Xilinx Virtex 5 XC5VLX330 FPGAs Convey refers to as *Application Engines* (AEs). The co-processor board houses eight memory controllers, each providing access to two DDR2 DIMMs, achieving a peak aggregate throughput of 20 GB/s per AE, or 80 GB/s total. A unique configuration of these FPGAs is referred to as a *personality*, and provides a set of x86 custom assembly instructions to applications running on the host server [3], [22].

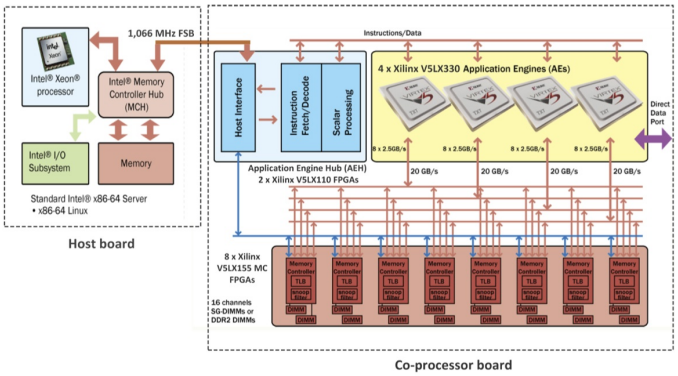

Fig. 1. Convey HC-1 system architecture. Convey Computer Corporation. Convey HC-1 Personality Development Kit Reference Manual, v.4.1, 2011. Used with permission of Convey Computer Corporation, 2014.

B. FPGA Design Productivity

Design productivity for FPGA-based computing has suffered from the contemporary ASIC "design productivity gap," in which the rapid rise in transistor density overwhelms the abilities of design tools and methodologies [1]. This gap especially hurts FPGA design productivity as these devices present unique needs and opportunities, such as reprogrammability (not available on ASICs). Contributors to high FPGA design productivity and challenges unique to non-engineer domain experts are discussed in this section.

High-level synthesis (HLS) is defined here as the compilation of a high-level, behavioral description language to a low, register-transfer level (RTL) hardware specification (e.g. Verilog and VHDL). These tools consume textual or graphical behavioral descriptions, which are analyzed to extract control dependencies and parallelism. By using a very high-level specification, HLS can drastically reduce the time required to arrive at a functionally correct solution when compared to RTL-based development. However, as the input description is abstracted, optimization of the hardware is shifted from the designer to the HLS compiler, often resulting in inefficient implementations. This effect can usually be mitigated by providing 'hints' (e.g. pragma statements in C-based HLS), to aid the compiler, though these hints are usually of a nature requiring digital design expertise.

Many early HLS tools made use of unconventional or tool-specific languages. In general, however, the use of a non-standard language as the entry format of an HLS tool creates

a barrier to widespread adoption [15]. To avoid this, numerous attempts have been made to use existing, widely adopted languages, such as C/C++/SystemC, Java, Python, OpenCL, etc., as the design entry point. Current HLS tools that accept C/C++/SystemC as input include Catapult (formerly Catapult C from Mentor Graphics) [6], Bluespec [5], ImpulseC [10], Xilinx HLS (formerly AutoPilot from AutoESL) [30], LegUp [7], and Synphony C [26]. Examples of OpenCL HLS tools include Altera's SDK for OpenCL [2], [9]. An excellent, albeit slightly outdated summary of HLS history is presented in [15].

C. Graphical Environments

Among the vast spectrum of HLS tools one can find the subset devoted to graphical development environments, distinguished from text-based HLS tools by the use of a graphical syntax. It can be argued that constraining the designer to indicating application behavior and structure graphically, rather than textually, forces him/her to think in terms of concurrent—rather than sequential—processes, which map well to the parallelism provided by FPGAs. Tools utilizing this approach include Xilinx System Generator [29], and National Instrument's LabVIEW platform with the LabVIEW FPGA Module [19], [20]. Both capture design input at a highly abstracted, behavioral level, compiling down to a hardware-level implementation.

D. LabVIEW

The LabVIEW platform, created by National Instruments, is a mature, widely adopted environment for system design, with a focus on instrument control/monitoring and data acquisition [17]. Design with the tool is achieved through drag-and-drop placement and interconnection of behavioral blocks in a flowchart-like manner to create Virtual Instruments (VIs), the functional building blocks in LabVIEW [18]. While programming a VI, designers have access to various control constructs, including for loops, time-line structures, and case statements, as well as an extensive library of reusable arithmetic, logic, and higher-level behavioral blocks. Each VI has a block-diagram and a front-panel, the former containing the VI's behavioral function, while the latter enables user input to the block-diagram and feedback of the VI at run time.

Alongside the standard development environment is the LabVIEW FPGA Module, a plugin that enables LabVIEW to target *supported* FPGA-based platforms [20]. The LabVIEW graphical language is used when targeting for FPGA targets; however, because some of LabVIEW's convenient control constructs and flexible data types cannot be mapped efficiently to FPGA fabric, they are not available when developing high-performance FPGA-based applications. These features include arrays of variable-size at run time, multidimensional arrays, and shared variables [21]. Despite these limitations, the tool maintains a high level of abstraction appropriate for non-engineers.

III. Approach: ConVI

This work, called *ConVI*, approaches the challenges discussed above through the assembly of a seamless, end-to-end development flow for the Convey HC-1 platform using the LabVIEW FPGA front-end environment. With ConVI, a user can design with a high-level, abstracted syntax, verify functional correctness through high-level local execution and system-wide HDL simulation, and deploy the implementation to a remote Convey HC-1 platform, all from within a single environment. The construction of this flow can be divided into three efforts:

- constructing of abstractions for low-level hardware such as memory controllers and control registers,
- incorporating the Convey HC compilation framework into LabVIEW's compilation process, and
- enabling front-panel control and design feedback during remote simulation and execution through a register access framework.

In addition to the assembly of the flow, a skeleton implementation of the Smith-Waterman algorithm, a common sequence alignment computation in the life-sciences field, was created and verified using *ConVI*. The above efforts and the Smith-Waterman implementation are discussed in this section.

A. Interface Abstractions

A typical Convey personality is comprised of complex building blocks, including memory interfaces, streaming interfaces, and host interactions. In ConVI, these interfaces are abstracted as simple VIs within LabVIEW, and further simplification of the memory interface is achieved by constraining the user to a streaming memory access model. Note that the decision to provide a streaming abstraction is based on its simplicity and intuitive behavior, and justified by the prevalence of applications in big-data HPC which can be mapped to a stream-based processing model (e.g. pipelined, SIMD processing). Software counterparts of these interface abstractions were created to ease interaction of the HC-1 host server and a LabVIEW-produced personality.

B. Compilation

LabVIEW FPGA includes an instance of Xilinx ISE for compiling VIs into FPGA configuration bitstreams. During assembly of *ConVI*, this implementation process was replaced by a set of scripts that extract the VI functionality and insert it into the AE's HDL hierarchy at compile time. The integration of the LabVIEW-generated HDL into the HC-1 AE hierarchy is shown in Figure 2. This is seamlessly integrated into the LabVIEW build process, such that the designer can remain in the friendly LabVIEW environment despite the outsourcing of the compilation process.

C. Front-Panel Control

Introducing a command line or waveforms to the environment when a design is in execution or simulation could reduce the flow's usability by a non-engineer; thus, to maintain front-end seamlessness, real-time control and diagnostic feedback was implemented for remote system simulation and execution on the HC-1. This was achieved through the creation of a local server on the development machine to forward data between the LabVIEW front-panel and a control register server on the HC-1. Using this connection, LabVIEW reads and writes hardware registers tied to the control and indicator ports in the VI, providing run-time control to the user. This architecture is better explained visually, and is shown in Figure 3.

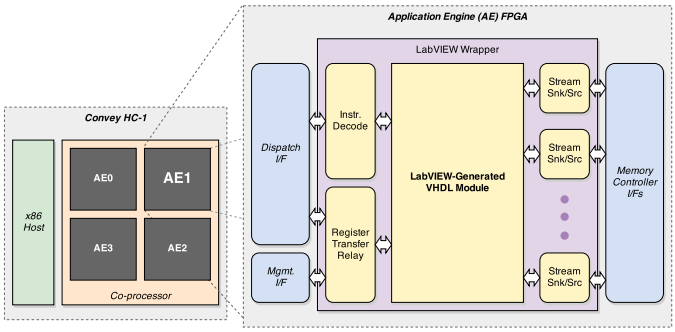

Fig. 2. Integration of the top-level LabVIEW VI into the existing HC-1 HDL framework.

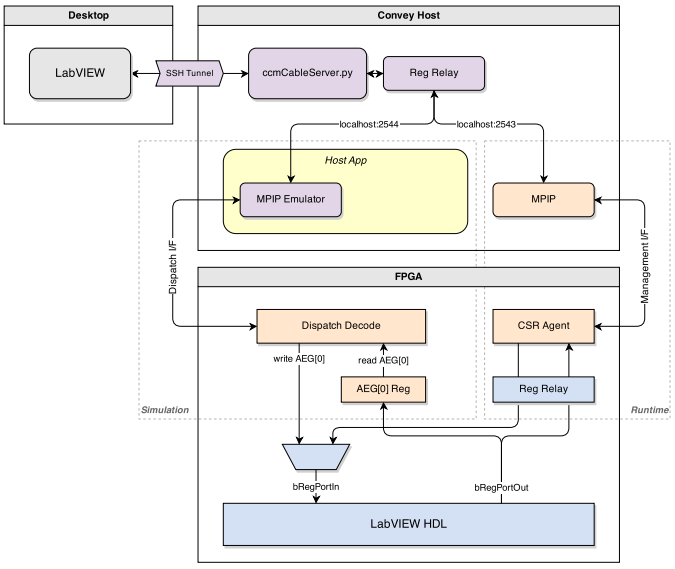

Fig. 3. Architecture of the register access framework that enables front-panel control during run time or simulation.

D. Smith-Waterman

The Smith-Waterman (SW) algorithm is a technique for comparing and aligning two sequences S and T with maximum sensitivity for a given parametrization. An alignment of two short sequences is shown in Figure 4. Proposed in 1981 to detect similar regions and compute optimal alignments of two sequences of DNA nucleotide or protein data, it is still widely used today [25]. The bottleneck of the algorithm is typically the computation of a large scoring matrix V of dimensions $|S|\times|T|$, as defined in the following equations, given alignment gap penalties α and β and substitution function σ.

$$V(i,j) = max \begin{cases} 0 \\ E(i,j) \\ F(i,j) \\ V(i-1,j-1) + \sigma(S[i],T[j]) \end{cases} \tag{1a}$$

$$E(i,j) = max \begin{cases} V(i,j-1) - \alpha \\ E(i,j-1) - \beta \end{cases} \tag{1b}$$

$$F(i,j) = max \begin{cases} V(i-1,j) - \alpha \\ F(i-1,j) - \beta \end{cases} \tag{1c}$$

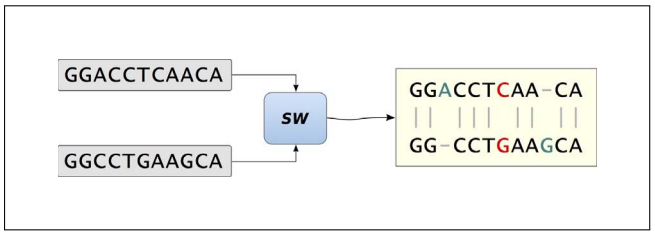

Fig. 4. Example of Smith-Waterman (SW) algorithm used to compare two DNA sequences. Note the mutations and gaps present in the resulting alignment.

Because of the structure of data dependencies in the scoring matrix, an entire anti-diagonal can be computed at each time step. Taking advantage of this, acceleration of the matrix computation is performed with a systolic array of processing elements, each computing a single column of the matrix in a wave-front pattern. In hardware, the systolic array is fixed with the contents of sequence S, and sequence T is streamed through the array.

The processing element (PE) implemented in this work is based on [31], each configured with one base from sequence S at run time. Every time step, each PE consumes a base of sequence T from its predecessor and computes the value of one cell of the scoring matrix. Because sequence T may be very large, it is infeasible to store the entire matrix contents in memory; rather, the systolic pipeline produces the maximum cell value per row of V, and downstream logic maintains one or more best alignment locations based on these values.

The top-level architecture of the Smith-Waterman implementation in ConVI is displayed in Figure 5. The accelerator is initialized by loading a query sequence S—the length of S is constrained to the number of PEs instanced in the accelerator—into the systolic array. Then, a large reference T is streamed through the array, and the best alignment locations are computed. Both sequences are loaded into coprocessor memory prior to execution, and control registers are used to provide the AEs with the length and location of each sequence, as well as send the alignment locations back to the host.

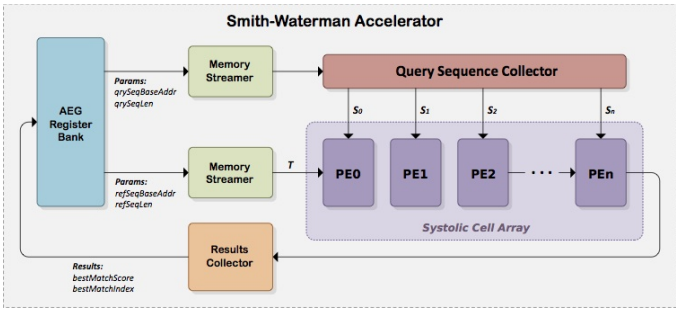

Fig. 5. Top-level architecture of the HC-1 accelerator for the Smith-Waterman matrix fill operation.

Within LabVIEW FPGA, approach to synchronous design is at a high level, incorporating abstractions such as for and while loops, which indicate periodic execution of all functions within the structure. Using these constructs, the SW systolic array could be implemented with great abstraction; however, performance requirements necessitated the use of a top-level

Single-Cycle Timed Loop (SCTL), within which was nested the majority of the implementation's functionality, including the systolic array. Data synchronization was achieved with "feedback nodes," which operated as simple registers, updating once every clock cycle. To assemble the systolic array, a LabVIEW script was created to automate PE replication and interconnection, enabling construction of systolic pipelines of arbitrary size. The content of each systolic cell is shown in Figure 6.

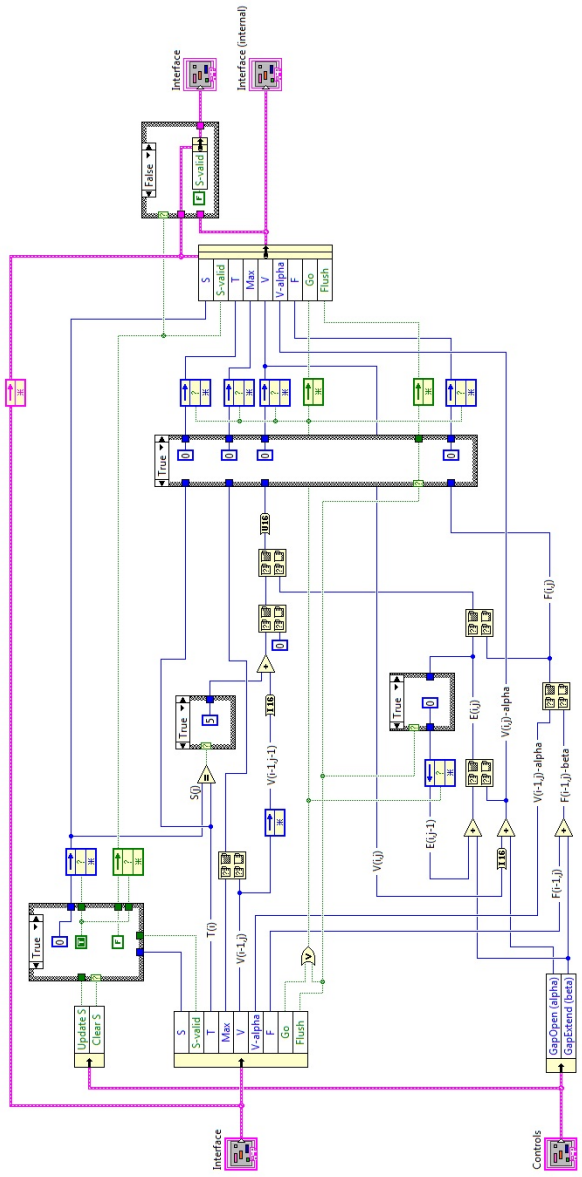

Fig. 6. Implementation of the Smith-Waterman processing element in LabVIEW.

E. VBI Workshops

To verify the usability and productivity that *ConVI* aims to provide, it were used in an NSF-funded summer program at a certain bioinformatics institute in 2013. A previous iteration of this work called *bFlow*, as described in [27], was tested at the 2012 institute, while *ConVI* was used in 2013.

Each occurrence of this two-week institute brought in 16 students from a diverse set of fields and levels of expertise. For the first week, all students attended workshops covering a wide range of topics related to HPC in the field of bioinformatics, and completed short assignments related to the topic. The students then split into four- or five-person groups, each spending the second week attempting to address a specific problem in the field. Prior to the event, the students in one of the groups were asked to review [31] to gain a basic understanding of the systolic array approach to implementing the accelerator. At the end of the first week, this group was given the skeleton SW implementation described in Section III-D, and tasked with extending its functionality and improving its performance. The results from these workshops are given in the next section.

IV. Results & Analysis

The constructed flow was given to participants of the NSF-sponsored workshop at a particular bioinformatics institute at some university. A group of five students was tasked with using *ConVI* to improve the performance and function of a skeleton Smith-Waterman accelerator. This section contains their results and a qualitative discussion of the usability challenges encountered by those using the flows.

The primary functional improvement initiated by the students was the tracking of multiple best alignment locations, in contrast to the previous year's group, which tracked only one location [27]. Several approaches were discussed, and two of the group members independently developed LabVIEW VIs to perform this function in an afternoon. This logic included the use of Boolean and numeric operations, as well as simple control structures. The seamless integration of the LabVIEW front-panel into Convey's system simulation framework allowed for relatively fast verification of their designs without significant technical expertise.

One of the more substantial drawbacks discovered was that the students designed primarily in LabVIEW under the *My Computer* or *Desktop* target, rather than the *FPGA* target, since only VIs under the former can be verified locally (i.e. without compiling the VI to HDL for system simulation). Because desktop-bound VIs have access to control and data constructs incompatible with FPGA targets, the students tended to design using FPGA-incompatible objects, and had to be reminded often to use only constructs and sub-VIs available to VIs under the FPGA target.

V. Conclusions

In summary, big-data in the sciences is a growing concern, especially in the field of bioinformatics, as the explosion of available genomic data is quickly outpacing the growth in usable HPC technology. FPGA-based heterogeneous systems, such as the Convey HC-1 platform, have the potential to address such growth through the creation of custom, high-performance accelerators; however, the use of existing development flows for such platforms (i.e. Convey Personality Development Kit) necessitates a heavy background in digital design concepts, forcing bioinformatics domain experts to influence accelerator development through a hardware engineer.

The contributions of this work has included the construction of *ConVI*, an end-to-end development flow based

on a graphical front-end design tool. Integrating full system simulation into the front-end, the flow enables domain experts without significant digital design experience to design, simulate, and deploy accelerators on the Convey HC-1 platform with minor to no intervention by a hardware engineer. An informal evaluation of the usability of *ConVI* was conducted at a summer bioinformatics workshop, indicating greater independence among non-engineer users when compared to a previous study.

VI. Acknowledgement

This work was supported in part by NSF I/UCRC IIP-1266245 via the NSF Center for High-Performance Reconfigurable Computing (CHREC) and by NSF Advanced CyberInfrastructure award 1124123.

References

[1] A. Allan, D. Edenfeld, W.H. Joyner, A.B. Kahng, M. Rodgers, and Y. Zorian. 2001 technology roadmap for semiconductors. *Computer*, 35(1):42–53, January 2002.

[2] Altera Corporation. Altera SDK for OpenCL. http://www.altera.com/products/software/opencl/opencl-index.html. [Online; accessed 1 Dec 2013].

[3] Jason D. Bakos. High-Performance Heterogeneous Computing with the Convey HC-1. *Computing in Science & Engineering*, 12(6):80–87, November 2010.

[4] Ian Bird. Computing for the Large Hadron Collider. *Annual Review of Nuclear and Particle Science*, 61(1):99–118, October 2011.

[5] Bluespec, Inc. Bluespec compiler. http://www.bluespec.com/high-level-synthesis-tools.html. [Online; accessed 1 Dec 2013].

[6] Calypto Design Systems, Inc. Catapult: Product Family Overview. http://calypto.com/en/products/catapult/overview/. [Online; accessed 1 Dec 2013].

[7] Andrew Canis, Jongsok Choi, Mark Aldham, Victor Zhang, Ahmed Kammoona, Jason H. Anderson, Stephen Brown, and Tomasz Czajkowski. LegUp. In *Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays - FPGA '11*, page 33, New York, New York, USA, February 2011. ACM Press.

[8] Convey Computer Corporation. Convey: Better Computing for Better Analytics. http://www.conveycomputer.com/products/hcseries/. [Online; accessed 19 Nov 2013].

[9] Tomasz S. Czajkowski, Utku Aydonat, Dmitry Denisenko, John Freeman, Michael Kinsner, David Neto, Jason Wong, Peter Yiannacouras, and Deshanand P. Singh. From opencl to high-performance hardware on FPGAS. In *22nd International Conference on Field Programmable Logic and Applications (FPL)*, pages 531–534. IEEE, August 2012.

[10] Impulse Accelerated Technologies, Inc. Impulse C. http://www.impulseaccelerated.com/products_universal.htm. [Online; accessed 18 Nov 2013].

[11] A.A. Khokhar, V.K. Prasanna, M.E. Shaaban, and C.-L. Wang. Heterogeneous computing: challenges and opportunities. *Computer*, 26(6):18–27, June 1993.

[12] Christiane Lefevre. LHC Guide, English version. A collection of facts and figures about the Large Hadron Collider (LHC) in the form of questions and answers. [Online; accessed 18 Nov 2013], January 2008.

[13] Kenli Li, Xiaoyong Tang, and Keqin Li. Energy-Efficient Stochastic Task Scheduling on Heterogeneous Computing Systems. *IEEE Transactions on Parallel and Distributed Systems*, PP(99):1–1, 2013.

[14] Lin Liu, Yinhu Li, Siliang Li, Ni Hu, Yimin He, Ray Pong, Danni Lin, Lihua Lu, and Maggie Law. Comparison of next-generation sequencing systems. *Journal of Biomedicine and Biotechnology*, 2012, 2012.

[15] G. Martin and G. Smith. High-Level Synthesis: Past, Present, and Future. *IEEE Design & Test of Computers*, 26(4):18–25, July 2009.

[16] L J McIver, J W Fondon III, M A Skinner, and H R Garner. Evaluation of microsatellite variation in the 1000 Genomes Project pilot studies is indicative of the quality and utility of the raw data and alignments. *Genomics*, 97(4):193–199, 2011.

[17] National Instruments Corporation. How Can I Use NI LabVIEW? - Application Areas. http://www.ni.com/labview/applications/. [Online; accessed 6 Dec 2013].

[18] National Instruments Corporation. Introduction to G Programming. http://www.ni.com/white-paper/7668/en/. [Online; accessed 7 Dec 2013].

[19] National Instruments Corporation. NI LabVIEW. http://www.ni.com/labview/. [Online; accessed 1 Dec 2013].

[20] National Instruments Corporation. NI LabVIEW FPGA Module. http://www.ni.com/labview/fpga/. [Online; accessed 1 Dec 2013].

[21] National Instruments Corporation. Unsupported LabVIEW Features (FPGA Module). http://zone.ni.com/reference/en-XX/help/371599J-01/lvfpgaconcepts/fpgamisc/. [Online; accessed 1 Dec 2013].

[22] Karl Savio Pimenta Pereira. *Characterization of FPGA-based high performance computers*. PhD thesis, Virginia Polytechnic Institute and State University, 2011.

[23] Michael A Quail, Miriam Smith, Paul Coupland, Thomas D Otto, Simon R Harris, Thomas R Connor, Anna Bertoni, Harold P Swerdlow, and Yong Gu. A tale of three next generation sequencing platforms: comparison of Ion Torrent, Pacific Biosciences and Illumina MiSeq sequencers. *BMC genomics*, 13(1):341, January 2012.

[24] Michael S Rosenberg. *Sequence alignment: methods, models, concepts, and strategies*. University of California Pr, 2009.

[25] Temple F Smith and Michael S Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[26] Synopsys, Inc. Synphony C Compiler Tool for High-Level C Synthesis. http://www.synopsys.com/Systems/BlockDesign/HLS/Pages/SynphonyC-Compiler.aspx. [Online; accessed 18 Nov 2013].

[27] David Uliana, Krzysztof Kepa, and Peter Athanas. Fpga-based hpc application design for non-experts. In *Rapid System Prototyping (RSP), 2013 International Symposium on*, pages 9–15. IEEE, 2013.

[28] Worldwide LHC Computing Grid. Worldwide LHC Computing Grid — WLCG. http://wlcg.web.cern.ch/. [Online; accessed 18 Nov 2013].

[29] Xilinx, Inc. System Generator for DSP, UG640 (v14.3). http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_4/sysgen_user.pdf, 2012.

[30] Xilinx, Inc. Introduction to FPGA Design with Vivado High-Level Synthesis, UG998 (v1.0). http://www.xilinx.com/support/documentation/sw_manuals/ug998-vivado-intro-fpga-design-hls.pdf, 2013.

[31] Peiheng Zhang, Guangming Tan, and Guang R. Gao. Implementation of the Smith-Waterman algorithm on a reconfigurable supercomputing platform. In *Proceedings of the 1st international workshop on High-performance reconfigurable computing technology and applications held in conjunction with SC07 - HPRCTA '07*, page 39, New York, New York, USA, November 2007. ACM Press.