The Advent of FPGA Accelerators in the Cloud

Suhaib A. Fahmy, Kizheppatt Vipin
School of Computer Engineering
Nanyang Technological University, Singapore
sfahmy@ntu.edu.sg

Abstract

Hardware accelerators implement custom architectures to significantly speed up computations in a wide range of domains. As performance scaling in server-class CPUs slows, we propose the integration of hardware accelerators in the cloud as a way to maintain a positive performance trend. Field programmable gate arrays represent the ideal way to integrate accelerators in the cloud, since they can be reprogrammed as needs change and allow multiple accelerators to share the communication infrastructure. We propose service levels for cloud accelerators and discuss some early efforts in this area, before identifying some important research directions.

Categories and Subject Descriptors CR-number [*subcategory*]: third-level

Keywords keyword1, keyword2

1. Introduction

Cloud Computing promises elastic access to unlimited compute resources, shared among many users, in much the same way as we view traditional utilities like the power grid. Virtualisation of these resources enables efficient scaling and sharing as user needs change. The established service models offer access to virtualised hardware, access to tuned application design platforms, or simply to full blown applications implemented on the cloud.

A key driver behind the increase in demand for cloud computing has been the exponential increase in use of mobile computing devices. As such devices often lack the computational power to complete complex tasks, such as voice recognition, they are offloaded to the cloud, where subsequent search tasks can also be run. Computation in the cloud benefits from server-grade CPUs with advanced datapaths, large caches, and multiple cores.

However, while the flexibility, scalability, and affordability of cloud computing are well established, performance remains a matter of concern. By virtue of the virtualisation of resources, the distributed communication among sometimes disparate nodes, and fluctuations in demand, running complex applications can be challenging [1]. Furthermore, the fundamental computational resources, CPUs, are simply not scaling in performance at the rates previously observed, especially at the enterprise level [2]. As a result, widespread adoption of the cloud in domains where performance is important remains on hold. Poor computational performance due to the overheads of virtualisation can severely impact response time and hence user experience.

One strategy discussed for overcoming this stalled performance scaling is to add heterogeneous resources to the cloud [3]. This would offer access to resources better suited to complex computation but maintaining the fundamental generality demanded by cloud computing. Some efforts have already emerged for integrating GPUs, as general purpose GPU (GPGPU) computing has gained in popularity [4, 5]. While GPUs can offer significant performance benefits in traditional computing systems, integration in the cloud is more troublesome, since the architectures are designed to be used monolithically. Hence, cloud providers who provide GPUs tend to offer them as a fixed resource using the Infrastructure-as-a-Service (IaaS) model, leading to potential under-utilisation and over-provision. Some recent work has explored methods for sharing GPUs among multiple applications [6] and this would be necessary for a cloud-friendly integration.

Though GPUs can be a useful resource for accelerating some types of application, this is not true for everything, and they remain highly power-hungry. Instead, we propose integration of FPGAs, that offer the ability to integrate custom hardware accelerators that can be changed on demand. While offering significant speedups in execution of a wide variety of tasks over CPUs, FPGAs are also significantly more power-efficient, resulting in a computational efficiency improvement in the orders of magnitude over both CPUs and GPUs [7–9]. While custom application specific inte-

Submission to SoCC '14, October, 2014, Seattle, WA, USA.

grated circuits (ASICs) can offer even higher performance and lower power consumption, the design effort is extremely high, and functionality is fixed once manufacturing is complete. So they would only make sense for very large corporations wishing to accelerate fixed functions over a long period.

A number of recent developments make it clear that cloud hardware accelerators are soon to take hold. First, at the server hardware level, manufacturers have recently announced changes that will ease integration of FPGAs in servers. IBM announced that their POWER8 processors would include integrated PCI Express Gen 3 support and the Coherent Accelerator Processor Interface (CAPI) [2]. These features significantly improve the coupling between the processor and a co-processing peripheral, and allow joint access to the memory space. Specific mention is made of FPGA accelerators as a motivation.

Intel also recently announced their XEON+FPGA solution, referencing the challenges we mentioned above [10]. They are integrating an FPGA with their XEON processors in a single chip package. They reference the 10× performance gains possible with FPGAs, and the further benefits of this tight coupling, offering low latency and coherency. And the key benefit of reprogrammability allows the FPGA to be used as benefits their clients.

Finally, a more complete demonstration of this approach was presented by Microsoft at ISCA 2014. Their Extreme Computing Group integrated FPGAs in some of their Bing servers, resulting in a doubling in performance for the affected tasks, at a small increase in power consumption [11].

Hence, it is clear that we may finally see the benefits afforded by FPGAs in mainstream computing. Though the reconfigurable computing community has long worked on integration of FPGAs with processors for high performance computing, integration in the cloud presents new challenges that require a close collaboration with experts in that domain. In this paper, we present some background on FPGAs: their capabilities, design, and integration. We then discuss existing related efforts in the integration of FPGAs with computers. Finally, we present our view of the relevant service models that can be applied for this paradigm of cloud computing.

2. Background

2.1 Field Programmable Gate Arrays

Field programmable gate arrays (FPGAs) are commercial off-the-shelf silicon devices (chips) that can be programmed to implement custom digital circuits. They emerged in the mid 1980s and were used primarily for glue logic—the functionality required to interface different components in a system at the board level. However, early on, the benefits of custom architectures for signal processing applications drove a more widespread adoption of FPGAs as accelerators for complex computational functions.

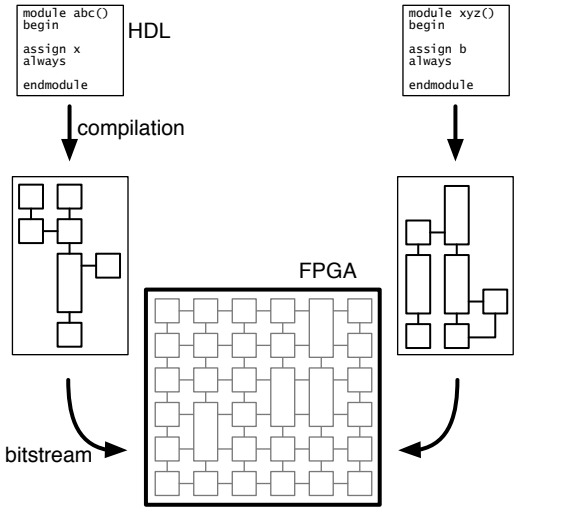

Figure 1. Simplified FPGA design flow.

Fundamentally, FPGAs consist of a mesh of basic circuit resources—at the core, lookup-tables (LUTs) which can store simple Boolean expressions, and flip-flops that allow state to be stored. These two fundamental components can be combined to create complex digital circuits. More recently, other resources such as small memory elements, and arithmetic components, have been added to further boost performance. While building a processor in an FPGA is possible, the key benefits are realised by designing custom computational datapaths suited to a particular application.

A designer describes a microarchitecture in a hardware description language (HDL) or uses high-level synthesis (HLS) tools to infer one from high level algorithmic code (e.g. C). Automated tools work out how to build this microarchitecture using the components in the FPGA, how they should be arranged on the grid of the FPGA and connected, finally generating a bitstream. This is a binary file which, when loaded into the FPGA's configuration memory, implements the circuit described by the initial description.

The performance advantage of an FPGA accelerator over a software implementation running on a processor is primarily as a result of exploiting parallelism inherent in the algorithm, and a spatial approach to computing rather than iteration of a single datapath. The key advantages over custom hardware in an ASIC is that there is no manufacturing process, turnaround time is significantly reduced, and the hardware design can be changed at any time simply by loading a new bitstream.

Within the cloud context, FPGAs represent an ideal hardware accelerator platform due to this application flexibility that preserves the generality of the resource. Furthermore, an FPGA can host multiple distinct, isolated accelerators through a mechanism called partial reconfiguration (see Section 2.3, where only part of the configuration memory is modified. This allows for a more fine-grained approach to using the hardware resource that fits perfectly with the

Figure 2. Partial reconfiguration of an FPGA.

ideas of scalability and sharing that are so fundamental in the cloud.

2.2 Coupling FPGAs with Computers

The reconfigurable computing community has explored coupling of FPGAs with general purpose processors for a long time, with numerous approaches proposed [12]. Traditionally, the FPGA has acted as a stand-alone accelerator with communication through I/O interfaces, resulting in high communication overheads. Over time, the FPGA has been brought closer to the CPU, resulting in higher communication bandwidth and lower latency [13]. In some cases, programmable logic has been proposed as an extension to a standard CPU to offer customised instructions. More recently, integrated systems on chip, combining both general purpose processors and FPGAs have emerged, such as the Xilinx Zynq architecture. These offer extremely high data bandwidth between the processor and FPGA, allowing for high performance applications with interleaved hardware and software execution.

Within a cloud context, this coupling is of great importance. Recent FPGAs support very high bandwidth serial communication interfaces, including PCI Express (PCIe). This allow designs to be targeted to different boards, the lower layers are built into the FPGA. While accessing these interfaces previously involved complex ad-hoc design, a number of open-source frameworks have recently emerged to simplify the process. As such, it is possible to almost saturate these interfaces, offering high bandwidth to the CPU memory space and on-board memory.

2.3 Partial Reconfiguration

Apart from the reprogrammability of FPGAs, partial reconfiguration is key to adoption in the cloud. Just as the virtual CPUs allocated to clients are abstracted from the physical CPUs, allowing scaling and sharing, accelerators in the cloud must support the same idea. Partial reconfiguration is where only a part of the FPGA is reconfigured instead of the whole device. This has the advantage of allowing all the communication infrastructure to remain in place while accelerators are reconfigured. It also allows us to host multiple independent accelerators with management of their reconfiguration not impacting each other.

This is done by outlining partially reconfigurable regions (PRRs) that are allowed to house different accelerators at runtime, as shown in Fig. 2. Partial bitstreams contain all the configuration information required to place each accelerator into a PRR. The static region is the part of the FPGA that is not reconfigured at runtime, and contains all the communication interfaces and reconfiguration management circuitry. Partial reconfiguration also has the benefit of smaller bitstreams, access to a faster reconfiguration port, and hence, much faster reconfiguration of accelerators in times of the order of 10 milliseconds.

3. Existing Work

While incorporation of FPGAs in the cloud is still a relatively new topic, there have been some efforts in this direction, and other work on high performance reconfigurable computing is also highly relevant.

One proposed application of FPGAs in the cloud is to provide better security and privacy by offloading sensitive data processing into hardware since possible attack vectors are limited [14, 15]. Netezza's data warehousing appliances perform complex data filtering on FPGAs, with additional compression/decompression performed by spare resources [16].

Microsoft's Catapult architecture [11], discussed earlier, represents the first detailed investigation of applying FPGAs within an enterprise-level datacentre application. They propose a number of FPGAs interleaved with the general purpose server resources. They offload the document ranking part of the Bing search engine to hardware. A custom design programmed in Verilog is split across 8 FPGAs within a rack, and they evaluate the full search stack performance on a deployment of over 1500 servers. They report almost doubled throughput in search ranking as a result of this integration, at a cost of only 10% increased power consumption and 30% increased total cost of ownership.

In [17], the authors present an integration of FPGAs in the cloud as standalone virtualised resources. They leverage extensions in OpenStack available as part of the Canadian SAVI Testbed to allow virtual FPGAs to be requested by clients. It is important to note that the FPGA resources are treated separately from the standard (server) resources in this setup, so the complete "application" should run in the FPGA. They also discuss the design of load balancers for this setup, and show that requested FPGA images can be brought up significantly faster than traditional virtual machines. Since they use apply traditional FPGA reconfiguration over a JTAG interface, extra cabling is required, and reconfiguration is slow. Hence, this approach would not work for applications that require dynamic switching of accelerators.

A longer line of work dealing with FPGAs for high performance computing also presents some relevant contributions. With large, complex applications and FPGAs that were significantly smaller, designers sought ways of virtualising FPGAs through the use of partial reconfiguration, allowing the applications to be completed with fewer resources [18, 19]. When considering the upper bounds on accelerators implemented in FPGAs instead of processors, they reported four orders of magnitude increases in performance, three orders of magnitude better power consumption, and two orders of magnitude in cost and size [20].

Generalised communication infrastructure has received significant attention recently, as more FPGA boards support

advanced interfaces. Open source interface frameworks have emerged allowing FPGAs to be integrated in PCs with communication throughput between the host and FPGA close to the capacity limits offered by modern PCIe interfaces [21]. Reconfiguration and communication over a single PCIe interface has also been demonstrated [22]. This overcomes the need for extra cabling and drivers required to configure FPGAs in the traditional manner, which can be problematic in a tightly managed datacentre environment.

Virtualisation of hardware resources has also been explored at different levels. While partial reconfiguration offers the benefits of time multiplexing, it still suffers from the fixed slots that must house accelerators, as discussed in Section 2.3. More generic communication infrastructures have been proposed to overcome this limitation and allow bitstreams to be moved around an FPGA [23], though this is more difficult with recent devices.. The management of reconfiguration at runtime has also been investigated. This has included integration within Linux [24] as well as customised operating systems [25]. Recent work demonstrated how such management could be done within a microkernel hypervisor, reducing overheads drastically [26].

Key challenges to be addressed for a cloud-centric integration of FPGAs are:

- Support for different accelerators as per application needs
- Fast reconfiguration of accelerators as requests are made
- Maximised usage of the FPGA resources at all times through efficient scheduling and allocation.
- Integration of accelerated tasks within software applications
- Easy to use tools allowing users to make use of accelerators
- Maintaining security of the platform against new attack vectors

4. Hardware in the Cloud Service Models

We present a set of different service models for incorporation of FPGAs in the cloud, and relate these to existing cloud service models.

4.1 Vendor Accelerators

For cloud applications in which clients are simply users, such as search engines or other Software as a Service (SaaS) deployments, the cloud provider can make use of FPGAs to offload complex computations, resulting in accelerated execution and increased (software) computational capacity in the CPU. Here, the FPGA is completely hidden from the clients, and the provider has full control. FPGAs have already been demonstrated as useful in a number of applications relevant to such deployments, including Memcached [27], and security management [14]. The benefits can be even more marked for more specific application domains and specialised SaaS solutions that incorporate complex algorithms well suited to FPGA acceleration, such as bioinformatics databases [28] or speech recognition [29].

Since the FPGAs are not being used directly by clients in this model, their presence is a way for the provider to improve efficiency, maximise performance, reduce latency, and improve overall user experience. An example of this service model is the Microsoft Catapult system discussed earlier [11]. The accelerator(s) could be fixed, as in Catapult, or partial reconfiguration can be used to load them dynamically. To facilitate this, the cloud resource manager (hypervisor) must be extended to add features for managing the FPGA including loading of bitstreams. Since the FPGA is entirely hidden from the clients, no additional security concerns arise.

Somewhat related to this service model are FPGA-based appliances which might be provided as an Infrastructure as a Service (IaaS) solution, such as those from Netezza [16].

4.2 Accelerators as a Service

In this model, the FPGAs are accessible, indirectly, to clients, for use as accelerators. This model makes most sense for Platform as a Service (PaaS) deployments. Here, the clients are not aware of the details of the integrated FPGA, nor do they provide their own accelerators. Rather, the provider makes available a library of accelerators that can be accessed by the client through a provided API. Their overall application can offload suitable computations to these accelerators through specific function calls.

Each FPGA-enabled server maintains a library of precompiled accelerator bitstreams. The FPGAs are split into multiple partially reconfigurable regions (PRRs) which are interfaced with the memory and PCI interfaces in a manner that maximises performance while maintaining fairness. When a request is made, the resource manager finds a suitable available PRR for the required accelerator, loads the partial bitstream, and then initialises the required data transfers.

Once again, as FPGA reconfiguration and communication are managed internally, no security concerns arise with this model. Providers can charge a premium for access to these accelerators, or even charge by data bandwidth used through them. Since the accelerators are maintained by the vendor, users may need to deal with some mismatch in exact requirements, but they should hence be designed to be as general as possible. This model will also have the widest reach since no FPGA experience whatsoever is required. vFPGAs are allocated and torn down regularly in this model, since they are only required when executing the associated functions.

4.3 Fabric as a Service

In this model the FPGAs are made available as a resource, allowing clients to to use their own accelerators. However, to meet cloud requirements, the physical FPGAs are not

accessible, but rather PRRs are available through a tightly managed implementation flow. The resource allocator still manages allocation, reconfiguration, and data movement. The client has access to virtual FPGAs (vFPGAs), i.e. PRRs, and can request multiple. These can also be combined into larger vFPGAs, mirroring the Amazon EC2 vCPUs that are each a hardware hyperthread on an Intel Processor.

The static part of each FPGA still hosts all the necessary communication and reconfiguration infrastructure and these are not visible to the user. Their own accelerator design must meet the interface standard for it to be compatible. Users design their own accelerator using a standard hardware design flow. Once complete, the generate the suitable bitstream and it is uploaded to the server to be stored within the user accelerator library. Similar functions to those for the Accelerator as a Service model are then used to manage loading and data transfer. Since these processes are managed through the provider's API, throughput and reliability are ensured.

Since clients now provide the bitstream to the server, security is a concern. Corrupt or malicious bitstreams might be used to cause physcical damage to the FPGAs. Hence, an authentication approach is necessary to ensure that the bitstreams have been correctly generated from supported tools. This is achieved through providing users with a customised implementation flow that watermarks accelerator designs and ensures bitstreams are generated for the available regions on the FPGA. This service model closely relates to IaaS, and the provider can charge for access to FPGAs as they do for other resources. In this model, users would typically have access to a vFPGA for a period of time, which may be mapped to different physical FPGAs.

4.4 Hardware Design as a Service

In this final model, the development infrastructure for building accelerators is also hosted in the Cloud. Users upload their hardware design source code using HDLs or HLS languages, and the implementation tools are run on the CPU in the cloud targeting multiple vFPGAs. Multiple tool instances can be run to try different hardware configurations and deduce the most efficient. Apart from testing using simulators, compiled designs can be tested in a vFPGA using test vectors provided by the user. Since all the tools run in the cloud, there are no security concerns. This approach combines IaaS for the vFPGA provision with PaaS for the tools access, and offers designers access to more compute power to improve their designs and meet complex constraints. An example of this approach is the suite of tools provided by Plunify [30].

5. Proposed Approach

We briefly discuss some of our recent work on building an FPGA-enabled cloud platform to explore the above deployment scenarios.

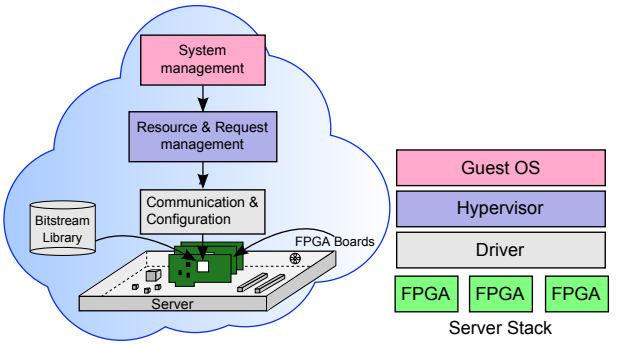

Figure 3. Proposed FPGA in the cloud architecture.

Two major obstacles to contend with are the lack of a general communication infrastructure and the low-speed external configuration interfaces. An ad-hoc approach to building communication infrastructure typically leads to low bandwidth between the software application and the hardware accelerator, and this can severely limit the acceleration benefits achievable, even with highly efficient accelerators. Similarly, high reconfiguration latency over traditional interfaces like JTAG, means reconfiguration time severely impacts efficiency. Since FPGAs support only a single reconfiguration operation at any point in time, this creates a bottleneck when multiple user requests arrive. Using external interfaces for reconfiguration also adds to the "cable spaghetti".

We have been working on open-source management infrastructure, which uses the same PCIe interface for both communication and reconfiguration management. This infrastructure is implemented in the FPGA static logic and can manage multiple accelerators concurrently. A built-in arbitrator guarantees fair communication bandwidth to every accelerator when multiple are communicating with the host server. We are able to achieve near maximum theoretical PCIe throughput. New accelerators are also configured over PCIe, providing superior reconfiguration performance and avoiding the need for external cabling.

To manage the platform, software infrastructure has been developed, managing low-level communication management through a driver and high-level virtualisation through a hypervisor. When a specific accelerator needs to be configured in the FPGA, the hypervisor decides the optimal PRR for it considering the least area variance. This ensures that larger PRRs are not blocked by smaller accelerators causing request rejections. The hypervisor also maintains a list of PRRs and the configured accelerators, to avoid unnecessary reconfiguration when a required accelerator is already present in the FPGA and not in use. An outline of our platform is depicted in Fig. 3.

A secure accelerator implementation framework targeting our hardware platform has also been developed. It generates the partial bitstream required for accelerator configuration from user provided HDL, while abstracting all the low-level steps in the FPGA implementation tool flow.

6. Research Directions

Integration of FPGAs in the cloud will entail work in a number of research directions. A key challenge remains the design of accelerators. Few non hardware experts are well versed in hardware design languages, yet domain experts are key to the design of relevant and effective accelerators. Recent efforts in high level synthesis promise to simplify the design of accelerators by offering automated mapping from high level languages like C. More recently, there have been tools under development to allow more parallel programs, in languages like OpenCL to be mapped to hardware automatically. Domain specific languages are also promising in this regard as they offer a more constrained mapping that can be more efficient.

Scheduling and management of FPGA resources in a shared on-demand environment also presents some challenges. THe aim would be to minimise resource wastage by always allocating accelerators to the most suitable PRRs, while also ensuring availability for subsequent requests.

Adding new resources to the cloud requires a re-evaluation of security and reliability aspects. We have shown how some of our proposed service models ensure security through only accepting accelerator bitstreams that have been generated by the provider. Management of hardware failures must also be tackled.

Integration of such resources within standard cloud management platforms like OpenStack is also essential for such work to find general applicability. This entails extending the OpenStack framework to deal with new types of resources and the specific allocation and management features required. One benefit of managing reconfiguration and communication over PCIe is that no fundamentally new low-level drivers are required for this integration.

7. Conclusion

We have discussed the potential of FPGA accelerators in the cloud. Integration of heterogeneous hardware resources offers an opportunity to significantly improve performance, and improve compute efficiency, overcoming the CPU performance scaling limits currently being faced. Hardware vendors have recognised this, resulting in next generation architectures supporting more efficient coupling of accelerators with server grade CPUs. FPGAs offer the advantages of high communication bandwidth, sharing among multiple accelerators, and dynamic loading of accelerators at run time through partial reconfiguration. We proposed a set of service models that offer a range of benefits to both the provider and client, and related these to existing cloud service models. Finally, we briefly discussed our prototyping platform that we hope to integrate within a cloud management framework.

A closer collaboration between researchers in the areas of reconfigurable computing and cloud computing can result in the significant benefits afforded by FPGAs finding mainstream adoption.

References

[1] K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, and N. J. Wright, "Performance analysis of high performance computing applications on the Amazon Web Services cloud," in *Proceedings of the IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2010, pp. 159–168.

[2] J. M. Tendler, "An introduction to the POWER8 processor," Presented at the IBM POWER User Group, Jan. 2014.

[3] G. Lee, B.-G. Chun, and R. H. Katz, "Heterogeneity-aware resource allocation and scheduling in the cloud," *Proceedings of HotCloud*, pp. 1–5, 2011.

[4] V. T. Ravi, M. Becchi, G. Agrawal, and S. Chakradhar, "Supporting GPU sharing in cloud environments with a transparent runtime consolidation framework," in *Proceedings of the International Symposium on High Performance Distributed Computing*, 2011, pp. 217–228.

[5] W. Zhu, C. Luo, J. Wang, and S. Li, "Multimedia cloud computing," *IEEE Signal Processing Magazine*, vol. 28, no. 3, pp. 59–69, 2011.

[6] K. M. Diab, M. M. Rafique, and M. Hefeeda, "Dynamic sharing of GPUs in cloud systems," in *Procedings of the IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2013, pp. 947–954.

[7] S. Kestur, J. D. Davis, and O. Williams, "BLAS comparison on FPGA, CPU and GPU," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2010, pp. 288–293.

[8] S. Asano, T. Maruyama, and Y. Yamaguchi, "Performance comparison of FPGA, GPU and CPU in image processing," in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL)*, 2009, pp. 126–131.

[9] D. B. Thomas, L. Howes, and W. Luk, "A comparison of CPUs, GPUs, FPGAs, and massively parallel processor arrays for random number generation," in *Proceedings of the ACM International Symposium on Field Programmable Gate Arrays*, 2009, pp. 63–72.

[10] D. Bryant, "Disrupting the data center to create the digital services economy," The Data Stack Blog (Intel), Jun. 2014.

[11] A. Putnam *et al.*, "A reconfigurable fabric for accelerating large-scale datacenter services," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2014.

[12] T. J. Todman, G. A. Constantinides, S. J. Wilton, O. Mencer, W. Luk, and P. Y. Cheung, "Reconfigurable computing: Architectures and design methods," *IEE Proceedings-Computers and Digital Techniques*, vol. 152, no. 2, pp. 193–207, 2005.

[13] S. C. Goldstein, H. Schmit, M. Budiu, S. Cadambi, M. Moe, and R. R. Taylor, "PipeRench: A reconfigurable architecture and compiler," *Computer*, vol. 33, no. 4, pp. 70–77, 2000.

[14] J.-A. Mondol, "Cloud security solutions using FPGA," in *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PacRim)*, 2011, pp. 747–752.

[15] K. Eguro and R. Venkatesan, "FPGAs for trusted cloud computing," in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL)*, 2012, pp. 63–70.

[16] P. Francisco *et al.*, *The Netezza Data Appliance Architecture: A Platform for High Performance Data Warehousing and Analytics*. IBM Redbooks, 2011.

[17] S. Byma, J. G. Steffan, H. Bannazadeh, A. Leon-Garcia, and P. Chow, "FPGAs in the cloud: Booting virtualized hardware accelerators with OpenStack," in *Proceedings of the IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2014, pp. 109–116.

[18] E. El-Araby, I. Gonzalez, and T. El-Ghazawi, "Performance bounds of partial run-time reconfiguration in high-performance reconfigurable computing," in *Proceedings of the International Workshop on High-Performance Reconfigurable Computing Technology and Applications*, 2007, pp. 11–20.

[19] ——, "Exploiting partial runtime reconfiguration for high-performance reconfigurable computing," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 1, no. 4, p. 21, 2009.

[20] T. El-Ghazawi, E. El-Araby, M. Huang, K. Gaj, V. Kindratenko, and D. Buell, "The promise of high-performance reconfigurable computing," *IEEE Computer*, vol. 41, no. 2, pp. 69–76, 2008.

[21] M. Jacobsen and R. Kastner, "RIFFA 2.0: A reusable integration framework for FPGA accelerators," in *Proceeding of the International Conference on Field Programmable Logic and Applications (FPL) International Conference on Field-Programmable Logic*, 2013.

[22] K. Vipin and S. A. Fahmy, "DyRACT: A partial reconfiguration enabled accelerator and test platform," in *Proceeding of the International Conference on Field Programmable Logic and Applications (FPL)*, 2014.

[23] M. Majer, J. Teich, A. Ahmadinia, and C. Bobda, "The Erlangen Slot Machine: A dynamically reconfigurable FPGA-based computer," *The Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, vol. 47, no. 1, pp. 15–31, 2007.

[24] J. A. Williams and N. W. Bergmann, "Embedded Linux as a platform for dynamically self-reconfiguring systems-on-chip," in *Proceedings of the International Conference On Engineering of Reconfigurable Systems and Algorithms (ERSA)*, 2004, pp. 163–169.

[25] E. Lübbers and M. Platzner, "ReconOS: An RTOS supporting hard-and software threads." in *Proceeding of the International Conference on Field Programmable Logic and Applications (FPL)*, 2007, pp. 441–446.

[26] A. K. Jain, K. D. Pham, J. Cui, S. A. Fahmy, and D. L. Maskell, "Virtualized execution and management of hardware tasks on a hybrid ARM-FPGA platform," *Journal of Signal Processing Systems*, 2014.

[27] S. R. Chalamalasetti, K. Lim, M. Wright, A. AuYoung, P. Ranganathan, and M. Margala, "An FPGA memcached appliance," in *Proceedings of the ACM International Symposium on Field Programmable Gate Arrays*, 2013, pp. 245–254.

[28] T. F. Oliver, B. Schmidt, and D. L. Maskell, "Reconfigurable architectures for bio-sequence database scanning on FPGAs," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 52, no. 12, pp. 851–855, 2005.

[29] E. C. Lin, K. Yu, R. A. Rutenbar, and T. Chen, "A 1000-word vocabulary, speaker-independent, continuous live-mode speech recognizer implemented in a single FPGA," in *Proceedings of the ACM International Symposium on Field Programmable Gate Arrays*, 2007, pp. 60–68.

[30] "Plunify pte ltd." [Online]. Available: http://plunify.com/