Accelerating the 3-D FFT Using a Heterogeneous FPGA Architecture

Matthew Anderson(✉), Maciej Brodowicz, Martin Swany, and Thomas Sterling

School of Informatics and Computing, Center for Research in Extreme Scale Technologies, Indiana University, Bloomington, IN 47408, USA
andersmw@indiana.edu

Abstract. Future Exascale architectures will likely make extensive use of computing accelerators such as Field Programmable Gate Arrays (FPGAs) given that these accelerators are very power efficient. Oftentimes, these FPGAs are located at the network interface card (NIC) and switch level in order to accelerate network operations, incorporate contention avoiding routing schemes, and perform computations directly on the NIC and bypass the arithmetic logic unit (ALU) of the CPU. This work explores just such a heterogeneous FPGA architecture in the context of two kernels that are driving applications in leadership machines: the 3-D Fast Fourier Transform (3-D FFT) and Asynchronous Multi-Tasking (AMT). The machine explored here is a DataVortex system which consists of conventional processors but with programmable logic incorporated in the memory architecture. The programmable logic controls the network and is incorporated both in the network interface cards and the network switches and implements a contention avoiding network routing. Both the 3-D FFT and AMT kernels show compelling performance for deployment to FFT driven applications in both molecular dynamics and density functional theory.

Keywords: FFT · FPGA · Heterogeneous systems
Asynchronous multitasking · High radix networks
Contention avoiding routing

1 Introduction

Future Exascale architectures will likely make extensive use of computing accelerators such as Field Programmable Gate Arrays (FPGAs) given that these accelerators are very power efficient. Oftentimes, these FPGAs are located at the network interface card (NIC) such as in the NetFPGA project [16] which has generated a large body of research on ways this configuration can improve networks. Programmable logic at the NIC not only offloads computation from the CPU to the NIC, but also enables more complicated routing schemes and topologies that can reduce contention at the scales Exascale researchers attempt to address. This work explores just such a heterogeneous FPGA architecture in

D. B. Heras and L. Bougé (Eds.): Euro-Par 2017 Workshops, LNCS 10659, pp. 653–663, 2018.
https://doi.org/10.1007/978-3-319-75178-8_52

the context of two kernels that are driving applications in leadership machines: the 3-D Fast Fourier Transform (3-D FFT) and Asynchronous Multi-Tasking (AMT).

The 3-D FFT kernel is a well known high performance computing (HPC) benchmark and is a key kernel in a wide range of HPC applications including molecular dynamics and density functional theory. AMT kernels, on the other hand, come from those emerging runtime models which combine multi-threading with some form of message-driven computation. These runtime models, sometimes referred to as "Asynchronous Multi-Tasking" or "AMT", feature the ability to express and perform fine grain thread parallelism in the context of distributed computation while also supporting the coarse grained parallelism of conventional parallel programming practice. Some examples of experimental AMT implementations include OCR [7], Legion [6], the Habanero family of languages [23,25,29,33], the Grappa framework for distributed shared memory [30], HPX [3,4], Qthreads [8], X10 [10], and Charm++ [1]. An emerging challenge for AMT implementations is that they generate a large number of small messages when operating in the modality of fine grain computation. While this may present a problem for a conventional system, a heterogeneous FPGA architecture is better equipped to handle this modality of operation.

The machine explored in this work is a DataVortex 200 series [2] which consists of conventional processors but with programmable logic incorporated in the memory architecture. The programmable logic controls the network and is incorporated both in the network interface cards and the network switches and implements a contention avoiding network routing. In June 2016 a DataVortex 200 series ranked 20th in the Green Graph 500 list [22] achieving 8.39 MTEPS per Watt.

For the 3-D FFT kernel, the expanded memory hierarchy in the network serves to significantly accelerate global memory rotations resulting in a significant speedup in FFT performance. For the AMT kernel, the incorporation of programmable logic directly controlling the network enables high memory bandwidth for small message sizes. These traits may prove crucial for applications in an Exascale setting.

This work is structured as follows. Related work is given in Sect. 2, followed by a detailed description of the prototype system and qualitative analysis of the potential of this type of architecture for Exascale in Sect. 3. Section 4 explores AMT runtime system requirements for dynamic applications and presents microbenchmark results empirically exploring small message behavior on the machine. Section 5 introduces the 3-D FFT kernel and explores the performance of this kernel in both a conventional and FPGA accelerated modality. The conclusions and directions for future work are given in Sect. 6.

2 Related Work

The incorporation of programmable logic into network interface cards has become extremely popular. The open source NetFPGA project [16] has been

cited in hundreds of academic works and is an open source field programmable gate array (FPGA) PCI Express board with Gigabit or Ten Gigabit Ethernet networking ports. This project has enabled hundreds of groups to experiment with programmable logic at the network interface card level with SRAM for data rearrangement and buffering. There are also many vendors selling PCI Express boards with programmable logic for building systems like the prototype system explored in this work including Bittware [12] and Alpha Data [11]. The widespread adoption of this technology for networks itself suggests the importance of studying a prototype system for potential Exascale use. The use of FPGAs for low-latency contention avoiding networking designs has already been adopted by high frequency traders in the financial industry [26,28].

Concurrent with the massive interest in programmable logic in network cards is the large body of topology work aimed at improving communication bandwidth through the elimination of link contention. Some examples of adaptive routing schemes to avoid contention include those of Reed [32], Deniziak and Tomaszewski [19] and Zhao et al. [36]. Topologies matter for performance and this is especially well illustrated in the Dragonfly work [24]. A significant strength of the prototype system explored in this work is the contention-avoiding topology.

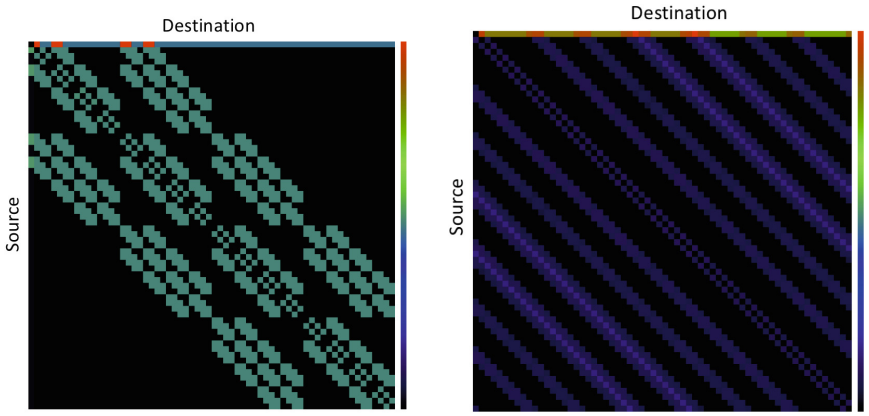
3 Experimental Setup

The prototype system for exploring the FFT and AMT kernels is the DataVortex 200 series system described here. The system consists of 8 nodes with one Intel Xeon E5-1630v3 operating at 3.7 GHz per node and 8 FPGA-based network interface cards. Each of these cards is an Altera Stratix 5 A7 FPGA and has 32 MB of SRAM. These network interface cards are connected to a switchboard consisting of four Altera Stratix 5 B6 FPGAs. The cards operate at a throughput of 550 million packets/sec and are connected across the PCIe 3.0 controller using eight lanes with an aggregate packet bandwidth of 35.2 GB/s in each direction for the entire 8 nodes. For network comparison studies, the prototype system also contains Mellanox Infiniband cards (Connect-X 3 VPI) to provide a redundant network against which to compare performance. The *stream* benchmark [9] on a single core of the prototype system indicates a sustained memory bandwidth of 14.5 GB/s.

All results in this work originate from the prototype system including both the AMT and FFT control cases which do not use programmable logic with SRAM and the DataVortex FPGA architecture cases which do. Due to the small size of the prototype system, there is no expected performance impact from the contention avoiding routing at the prototype system scale. The contention avoiding routing algorithm implemented is that of Reed [32]. In both the AMT and FFT kernels, the SRAM of the FPGAs is heavily utilized in order to accelerate strided memory accesses in conjunction with network operations. The small message behavior of the FPGA driven network is key for just the AMT kernel. Small messages are a key component of AMT runtime systems and have shown significant potential for improving the scalability and performance of scaling constrained applications [18,34].

4 Asynchronous Multi-tasking Kernel

Asynchonrous Multi-Tasking runtime systems frequently target medium to fine grain thread parallelism rather than the coarse-grained process parallelism employed in conventional parallel programming practices. This approach can significantly improve efficiency in algorithms with irregular and time-varying execution properties and show promise for Exascale usage. However, dynamic task and resource management execution for fine grain thread parallelism also results in a large number of small messages rather than the relatively small number of large messages that frequently appears when using conventional parallel programming practice. An example of this is illustrated in Fig. 1. Figure 1 shows a visualization of the sparsity pattern of network communication and size of messages for two different execution modalities of the Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics (LULESH) mini-application. LULESH [5] is a proxy application representing a commonly used kernel in scientific computation intended to better measure and reflect realizable performance on high performance computing architectures than benchmarks such as High Performance Linpack (HPL) [20] while also serving as a performance measure for potential Exascale architectures and to optimize for power, energy, and performance [27]. The two different execution modalities explored are coarse grain parallelism as typified using conventional parallel programming practice and asynchronous multi-tasking for fine grain parallelism and the modalities were explored using the SST/macro simulator [21] where MPI was used for the



(a) Conventional Practice (b) AMT runtime

Fig. 1. A visualization of the sparsity pattern of network communication and size of messages for the LULESH mini-application for both coarse grained conventional practice and a fine grained AMT approach. The color indicates the size of the messages with red being the largest and black being zero size. The conventional practice shows fewer but larger mesages while the AMT approach shows many more messages of much smaller size. (Color figure online)

conventional approach and HPX was used for the AMT approach. LULESH performance and network behavior were simulated on 64 nodes of a Cray XE6 for both modalities with significant overdecomposition in the AMT modality. The AMT approach generates significantly many more smaller messages than the conventional approach. Most networks, however, show their best efficiency with fewer, larger messages. In order to explore typical AMT behavior, the AMT kernel explored in this section is alltoall communication limited to 8 byte messages.

The prototype system programmable logic network shows behavior substantially different from conventional networks and favors large numbers of small messages such as what is seen in AMT runtime executions like that of Fig. 1. Alltoall communication bandwidth for 8 byte messages comparing infiniband and the programmable logic network of the prototype system is shown in Fig. 2. For many small messages the programmable logic network significantly increases communication bandwidth for typical AMT execution modalities. This characteristic may become an important feature for future Exascale architectures and is a natural consequence of the programmable logic network created for the prototype system here.

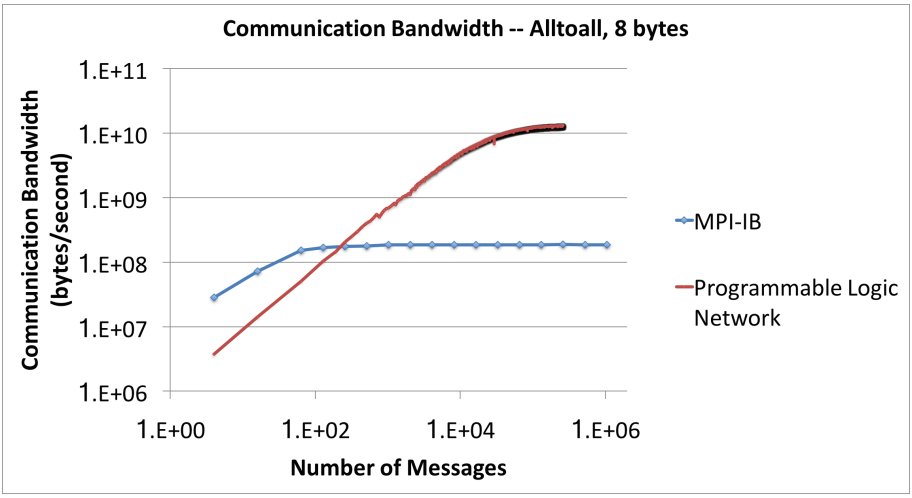


Fig. 2. Alltoall communication bandwidth for 8 byte messages comparing infiniband and the programmable logic network of the prototype system. For many small messages such as that in the AMT modality of Fig. 1, the programmable logic network has the benefit of not only employing contention avoiding routing but also increasing communication bandwidth for AMT execution modalities.

5 3-D Fast Fourier Transform

The 3-D Fast Fourier Transform (FFT) is a key scientific computing kernel used in many widely used software frameworks and toolkits. Some of these include

widely used molecular dynamics toolkits such as NAMD [31] and Gromacs [14] and Density Functional Theory toolkits such as VASP [17]. In NAMD, the smooth Particle-Mesh Ewald method [35] is critically dependent on the distributed 3-D FFT implementation for both performance and scalability. The 3-D FFT is also an important kernel for computational fluid dynamics simulations. Any potential Exascale architecture will need to compute the 3-D FFT extremely efficiently as well as strong scale without generating a lot of network contention.

Among the many ways to implement a 3-D FFT, several global memory rotations are usually implemented so that a 1-D FFT is applied along 1-D lengths of data that are stored consecutively in memory for fast access. No strided memory accesses occur this way and such 3-D FFT implementations are very fast. However, such global memory rotations are expensive requiring both a large alltoall operation and some data reordering. The incorporation of SRAM in the programmable logic network enables the network to also perform such a memory rotation when taking each of the x, y, and z FFTs and thereby compute the FFTs using the fastest memory layout possible. These rotations are illustrated in Fig. 3. The 3-D data is decomposed across the distributed memory system in just one dimension giving each CPU access to the entire fast dimension domain memory each time an FFT is computed. The initial memory layout has fastest access in the z direction and so the z FFT is computed first. The first rotation then places fastest memory access in the x direction for computing the x FFT. The second rotation places the fastest memory access in the y direction for computing the y FFT. The last rotation returns the memory to the original layout. Rotations in memory and FFT computations are entirely overlapped due to the expanded memory hierarchy in the network.

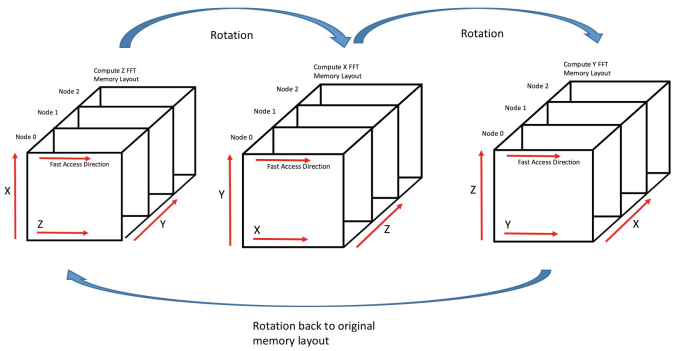


Fig. 3. The programmable logic network and associated SRAM are used to perform quick memory rotations and network communication for optimal memory layout of FFT computations. For a memory layout that begins with fastest access in the z direction, the z FFT is computed first. The first rotation then places fastest memory access in the x direction for computing the x FFT. The second rotation places the fastest memory access in the y direction for computing the y FFT. The last rotation returns the memory to the original layout.

For comparison purposes, the FFT kernel on the programmable logic network is compared against performance from the widely used MPI-FFTW library [13] and the FFT NAS Parallel Benchmark [15]. All of the comparison cases using MPI-FFTW or the NAS Parallel Benchmark were conducted on the prototype system but used the infiniband network. Figure 4 gives the time to solution for a fixed problem size, 1024^3 3-D FFT run from 1 to 8 nodes on the prototype system for complex double precision. In this figure, the lower the time to solution, the better the result. The programmable logic network version significantly outperforms the MPI-FFTW comparison in each case by around a factor of 4 or 5. The performance of the programmable logic network FFT is also better even on a single node, reflecting the usage of the fast programmable logic SRAM for data rearrangement and optimal FFT computation even while not in a distributed modality. Figure 5 gives the strong scaling speedup for the complex double 3-D FFT calculations. The programmable logic network FFT scales linearly with the number of nodes in both cases in addition to giving the significant performance advantage illustrated in Fig. 4.

The NAS Parallel Benchmark for FFT enables a comparison in terms of GFlops with the programmable logic network FFT. This comparison is shown in Fig. 6. In these results, several different problem sizes were explored consistent with NAS PB classes A–D while the programmable logic network FFT was performed at cubic sizes. In each case, the entire 8 node system was used. The performance improvement when using the programmable logic network versus

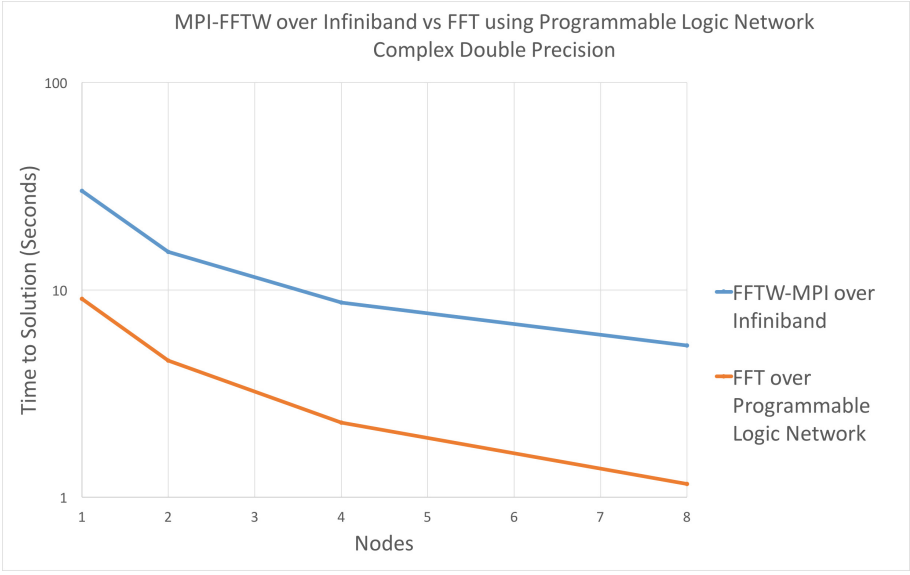


Fig. 4. Strong scaling result showing the time to solution between a 1024^3 complex double 3-D FFT using either MPI-FFTW over infiniband or the FFT with the programmable logic network. In this plot, the lower the line, the faster the time to solution and the better the result. All simulations used the prototype system. The scaling comparison for this data is found in Fig. 5.

the NAS PB over infiniband is between a factor of 3 and 4. The peak sustained performance for the programmable logic network version of the 3-D FFT was 163.1 GFlops over 8 nodes.

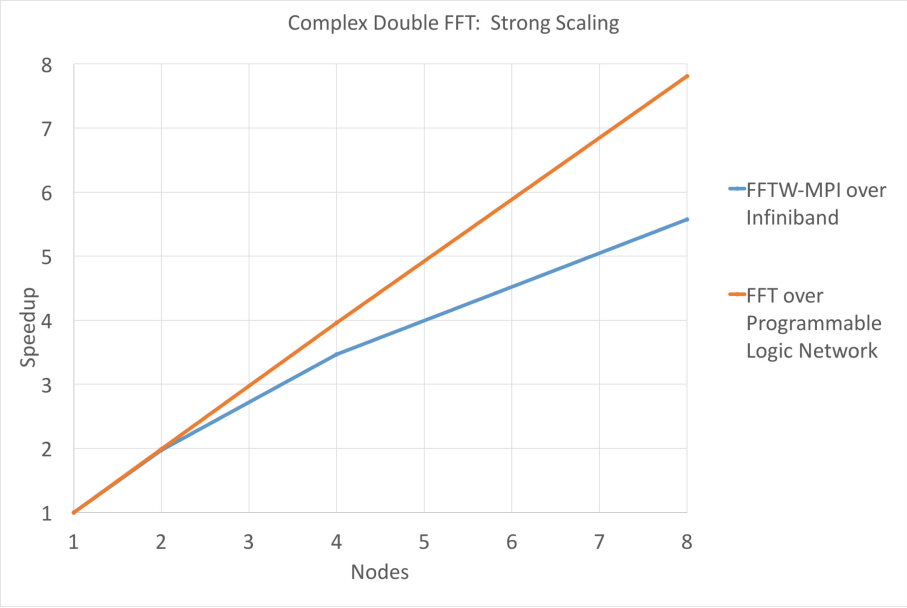


Fig. 5. Strong scaling result showing the speedup between a 1024^3 complex double 3-D FFT using either MPI-FFTW over infiniband or the FFT with the programmable logic network. In this plot, the higher the line, the better the scalability and the result. All simulations used the prototype system. The time to solution comparison for this data is found in Fig. 4.

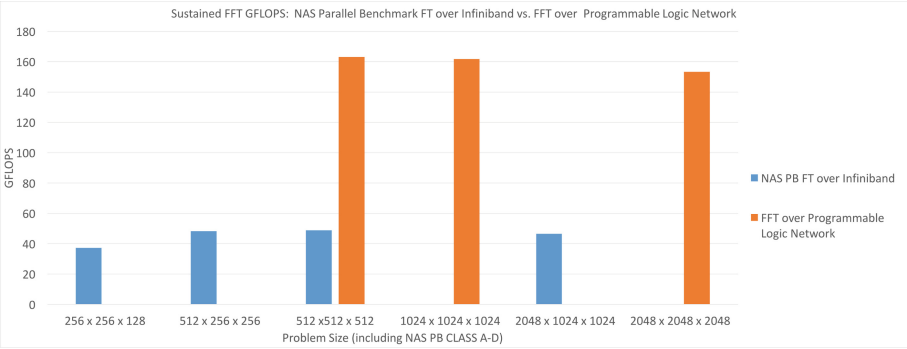


Fig. 6. A comparison of sustained GFlops for the FFT operation comparing the NAS Parallel FT Benchmark over Infiniband with the FFT over the programmable logic network. All results used the prototype system and use the the full system (8 nodes). Multiple 3-D problem sizes are explored. The NAS parallel FT benchmark peaks at 49 GFlops while the FFT over the programmable logic network peaks at 163 GFlops.

6 Conclusions

Because it is expected that FPGAs will likely play a significant role in reducing power consumption in emerging and future supercomputers, this work has explored a heterogeneous FPGA machine which incorporates programmable logic that both expands the memory architecture and controls the network. Two application motivated scientific computing kernels were explored on a small 8 node prototype system: an AMT kernel consisting of alltoall with 8 byte messages and the 3-D fast Fourier transform. While the AMT kernel tested the small message communication bandwidth capability of the system, the 3-D FFT kernel tested the global memory rotation capability of the system in order to accelerate performance over conventional practice.

High communication bandwidth for small messages was explored due to its importance for asynchronous multi-tasking runtime systems which target medium to fine grain thread parallelism and generate large numbers of small messages. This was demonstrated explicitly in this work using the SST/macro simulator. AMT runtime systems may become key components of the Exascale software stack both to improve efficiency and programmability. The FPGA machine was able to significantly outperform the equivalent infiniband 8 byte message alltoall. While a conventional application running on a conventional machine addresses this performance issue through message coalescence, AMT applications will often opt to avoid coalescence for greater overlap of computational phases. In this modality, the FPGA machine shows promise.

Fast rotations in conjunction with the programmable logic network enable a very fast algorithmic approach to 3-D FFT's so that the memory layout is arranged for optimal access at the same time the network exchanges necessary data between nodes. This results in performance improvements of as much as a factor of 5 over conventional practice in computing 3-D FFT's for the small prototype system in this work. Future work will directly explore the performance impact of this architecture on molecular dynamics toolkits like NAMD and density functional theory toolkits like VASP.

References

1. Charm++. http://charm.cs.illinois.edu/research/charm/
2. Datavortex. http://www.datavortex.com/
3. HPX. http://stellar.cct.lsu.edu/tag/hpx/
4. HPX-5. http://hpx.crest.iu.edu
5. Hydrodynamics Challenge Problem. Technical report LLNL-TR-490254, Lawrence Livermore National Laboratory
6. Legion programming system. http://legion.stanford.edu/
7. Open Community Runtime. https://01.org/open-community-runtime
8. Qthreads. http://www.cs.sandia.gov/qthreads/
9. Stream benchmark. https://www.cs.virginia.edu/stream/
10. X10. http://x10-lang.org/
11. Alpha data (2016). www.alpha-data.com

12. Bittware (2016). www.bittware.com
13. FFTW (2016). www.fftw.org
14. GROMACS (2016). www.gromacs.org
15. NAS parallel benchmarks (2016). https://www.nas.nasa.gov/publications/npb.html
16. NetFPGA project (2016). netfpga.org
17. VASP (2017). www.vasp.at
18. Anderson, M., Brodowicz, M., Kulkarni, A., Sterling, T.: Performance modeling of gyrokinetic toroidal simulations for a many-tasking runtime system. In: Jarvis, S.A., Wright, S.A., Hammond, S.D. (eds.) PMBS 2013. LNCS, vol. 8551, pp. 136–157. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10214-6_7
19. Deniziak, S., Tomaszewski, R.: Contention-avoiding custom topology generation for network-on-chip. In: Proceedings of the 2009 12th International Symposium on Design and Diagnostics of Electronic Circuits and Systems, DDECS 2009, pp. 234–237. IEEE Computer Society, Washington, DC, USA (2009). https://doi.org/10.1109/DDECS.2009.5012136
20. Dongarra, J.: Performance of various computers using standard linear equations software. Technical report CS-89-85, University of Tennesse Computer Science (2014). http://www.netlib.org/benchmark/performance.pdf
21. Hendry, G., Rodrigues, A.: SST: a simulator for exascale co-design. In: Proceedings of the ASCR/ASC Exascale Research Conference (2012)
22. Hoefler, T.: Seventh green graph 500 list (2016). http://green.graph500.org/
23. Imam, S., Sarkar, V.: Habanero-Java library: a Java 8 framework for multicore programming. In: 11th International Conference on the Principles and Practice of Programming on the Java Platform: Virtual Machines, Languages, and Tools (PPPJ 2014), September 2014
24. Kim, J., Dally, W.J., Scott, S., Abts, D.: Technology-driven, highly-scalable Dragonfly topology. In: Proceedings of the 35th International Symposium on Compute Architecture, ISCA 2008. IEEE (2008)
25. Kumar, V., Zheng, Y., Cave, V., Budimlic, Z., Sarkar, V.: HabaneroUPC++: a compiler-free PGAS library. In: 8th International Conference on Partitioned Global Address Space Programming Models (PGAS14), October 2014
26. Leber, C., Geib, B., Litz, H.: High frequency trading acceleration using FPGAs. In: 2011 21st International Conference on Field Programmable Logic and Applications, pp. 317–322, September 2011
27. Leon, E., Karlin, I., Grant, R.: Optimizing explicit hydrodynamics for power, energy, and performance. In: 2015 IEEE International Conference on Cluster Computing (CLUSTER), pp. 11–21, September 2015
28. Lockwood, J., Gupte, A., Mehta, N., Vissers, K.A.: A low-latency library in FPGA hardware for high-frequency trading. In: IEEE 20th Annual Symposium on High-Performance Interconnects, pp. 9–16, August 2012
29. Majeti, D., Sarkar, V.: Heterogeneous Habanero-C (H2C): a portable programming model for heterogeneous processors. In: Programming Models, Languages and Compilers for Manycore and Heterogeneous Architectures (PLC), May 2015
30. Nelson, J., Holt, B., Myers, B., Briggs, P., Ceze, L., Kahan, S., Oskin, M.: Grappa: a latency-tolerant runtime for large-scale irregular applications. In: International Workshop on Rack-Scale Computing (WRSC w/EuroSys), April 2014
31. Phillips, J.C., Braun, R., Wang, W., Gumbart, J., Tajkhorshid, E., Villa, E., Chipot, C., Skeel, R.D., Kale, L., Schulten, K.: Scalable molecular dynamics with NAMD. J. Comput. Chem. **26**, 1781–1802 (2005)

32. Reed, C.: Means and apparatus for a scaleable congestion free switching system with intelligent control III, US Patent 7835278, November 2010
33. Sarkar, V.: Habanero-Scala: Async-finish programming in Scala. In: The Third Scala Workshop (Scala Days 2012), April 2012
34. Treichler, S., Bauer, M., Aiken, A.: Realm: an event-based low-level runtime for distributed memory architectures. In: Proceedings of the 23rd International Conference on Parallel Architectures and Compilation, PACT 2014, pp. 263–276. ACM, New York, NY, USA (2014). https://doi.org/10.1145/2628071.2628084
35. Essmann, U., Perera, L., Berkowitz, M.L., Darden, T., Lee, H., Pedersen, L.G.: A smooth particle mesh Ewald method. J. Chem. Phys. **103**, 8577–8593 (1995)
36. Zhao, J., Zhou, Q., Cai, Y.: Fast congestion-aware timing-driven placement for island FPGA. In: Proceedings of the 2009 12th International Symposium on Design and Diagnostics of Electronic Circuits and Systems, DDECS 2009, pp. 24–27. IEEE Computer Society, Washington, DC, USA (2009). https://doi.org/10.1109/DDECS.2009.5012092