2014 IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines

3D FFTs on a Single FPGA

Benjamin Humphries Hansen Zhang Jiayi Sheng Raphael Landaverde Martin C. Herbordt
Department of Electrical and Computer Engineering
Boston University, Boston, MA

***Abstract*—The 3D FFT is critical in many physical simulations and image processing applications. On FPGAs, however, the 3D FFT was thought to be inefficient relative to other methods such as convolution-based implementations of multigrid. We find the opposite: a simple design, operating at a conservative frequency, takes 4μs for 16^3, 21μs for 32^3, and 215μs for 64^3 single precision data points. The first two of these compare favorably with the 25μs and 29μs obtained running on a current Nvidia GPU. Some broader significance is that this is a critical piece in implementing a large scale FPGA-based MD engine: even a single FPGA is capable of keeping the FFT off of the critical path for a large fraction of possible MD simulations.**

***Keywords*-High Performance Reconfigurable Computing; FFT;**

I. Introduction

The FFT is one of the most important applications implemented on FPGAs with the 1D and 2D versions finding uses especially in signal and image processing, respectively. A small sample of the massive amount of previous work includes [1]–[3]; IP for many variations of the 1D FFT is available from Altera and Xilinx [4], [5]. The 3D FFT is also critical: it is often the heart of electrostatics computations such as those used when computing the long-range force in Molecular Dynamics simulations (MD). But although MD on FPGAs has been widely studied, there have been few reports about the 3D FFT on FPGAs [3], [6], [7].

This prior work, and also that for large 2D FFTs (e.g., [8]), assumes the data set is too large to fit on chip. It therefore concentrates on efficient orchestration of memory access and data placement to instantiate communication, especially the transpose between phases. With current technology, however, most useful 3D FFTs for electrostatics can be run holding all data on chip. As is common when the traversal of a packaging boundary is removed, this leads to a "game-changing" difference in performance. The primary contribution here is demonstrating this difference and evaluating its consequence with respect to other compute technologies.

Our motivation is as follows: While in previous work it has been shown that the MD range-limited force can be effectively implemented on FPGAs [9], no comparable implementation exists for the long-range force. In fact, noting the difficulties with the 3D FFT at the time, we previously used different approach implementing it with multigrid [10], [11]. Although multigrid appears to be a good fit, it nonetheless proved to have neither sufficiently high performance nor accuracy; we therefore revisit the 3D FFT on FPGAs.

We use the following approach. First, we constrain the problem size and precision to those likely to be encountered on the critical path of electrostatics calculations. These are for problems sizes where strong scaling is problematic, primarily those with less than a few hundred thousand particles. These generally translate into FFTs with 32^3 and 64^3 grid points of single precision floating point [12]. Second, we take advantage of existing IP, in this case by Altera and Xilinx, to supply the 1D FFTs that are the basis of the design. Our rationale is that not only do the primary vendors integrate the existing algorithmic state-of-the-art, they also take advantage of device-specific features. Finally, we use a conservative design with simple timing and control.

We find that even with only logic-level optimizations, the 3D FFT takes 21μs for 32^3 single precision data points, a number somewhat better than that obtained from a current GPU. The significance is that this is sufficient to keep the FFT off of the critical path for a large fraction of possible MD simulations.

II. Approach and Implementation

Approach. Higher dimensional FFTs are decomposable into lower dimensional. Therefore the N^3 point 3D FFT can be computed by executing three sets of N^2 N-point 1D FFTs consecutively in the three dimensions. We assume a number of 1D FFT IP blocks similar to the number of points in a single dimension. With a current high-end FPGA this translates into a maximum of 32 IPs for 16^3 FFTs and 64 IPs for 64^3 FFTs. The number of "RAMs" is equal to the number of IPs. When necessary multiple BRAMs are ganged together to form a virtual RAM using standard EDA methods.

There are various ways to map data onto the RAMs. Figure 1 shows perhaps the most obvious: 2D slices (or slabs) of the cube are mapped onto each RAM. Each IP then calculates the $2N^2$ N-point 1D FFTs for dimensions

This work was supported in part by the NSF through award #CNS-1205593 and the NIH through award #R41-GM101907-01A1. Email: (bhump78|hszhang|jysheng|soptnrs|herbordt)@bu.edu

978-1-4799-5111-6/14 $31.00 © 2014 IEEE
DOI 10.1109/FCCM.2014.28

D1 and D2 using data only from a single RAM. In Figure 1, each IP/RAM combination does this for four 2D slices. Computing the FFTs for D3 requires traversing multiple RAMs, or transposing the data. We have decided to do the former by routing data with a crossbar.

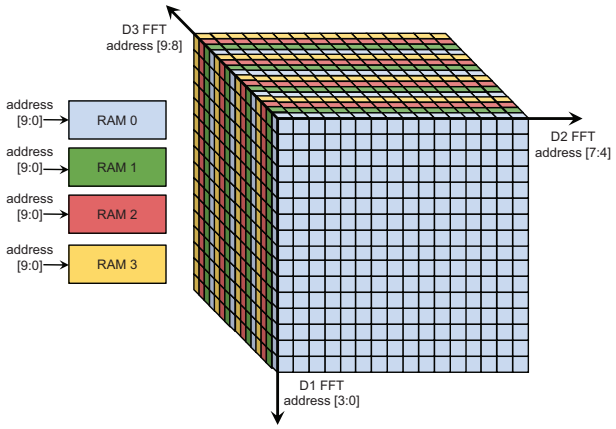

Figure 1. A possible mapping of points from a 16^3 FFT onto four RAMs.

Design Overview. As shown in Figure 2 the design has four main parts: RAMs, Crossbars, FFT Pipelines (the 1D IP), and Controller. The RAM's primary purpose is simply to store the data throughout the computation. The Crossbars work in conjunction with the RAMs to select the flow of data so as to effect transpose and untranspose as needed. The Controller is a large state machine that drives all of the inputs to the RAMs, Crossbars, and FFT Pipelines. For the Xilinx FFT pipelines we have used the Xilinx LogiCORE FFT v8.0 IP generator, in particular, Float32 with natural order output, pipelined streaming I/O, non-configurable transaction lengths, and real-time throttling. We have used the analogous IP for the Altera FPGAs.

This crossbar-based design is somewhat more general than strictly needed, but justified for two reasons. The first is that, while not scalable, the 32x32 and 64x64 crossbars require only a small fraction of the overall chip resources and so are a small price to pay for uniformity. The second is that the crossbars instantiate a communication mechanism sufficiently general for integration into FPGA-centric clusters. This is mentioned very briefly in the Discussion.

Dataflow. The 1D FFT blocks selected behave similar to FIFO delay elements and are used by inputting and outputting one word per clock. A full FFT is calculated by clocking in all words, waiting a fixed number of cycles, and then clocking all words out. The IPs selected allow for words from subsequent FFT frames to be input as it is calculating and outputting prior frames.

We now very briefly describe the dataflow. Overall, given that a particular RAM index and RAM address is always the home of any given data point, the controls to route data out of the FFT Pipelines are *delayed mirrors* of the controls to

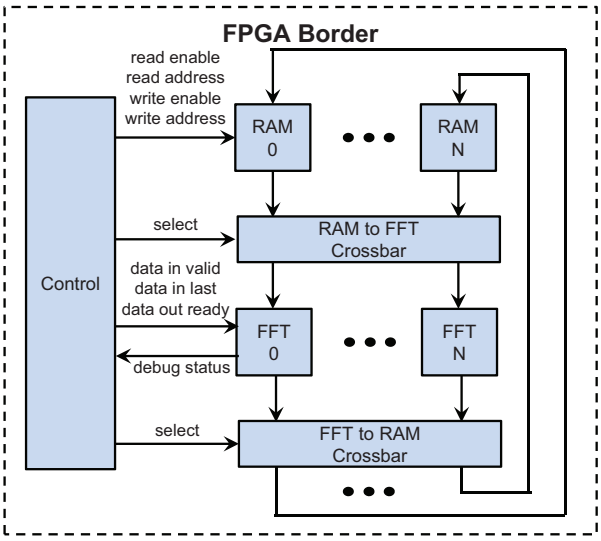

Figure 2. Block diagram for 3D FFT design.

route data into the FFT Pipelines. This greatly simplifies the modeling of the dataflow to the point that only the input flow has to be modeled and the output flow will simply be the input flow delayed by the latency of the FFT Pipeline. The one caveat is that the input routing flow must ensure that the data points from the prior FFT dimension have been written back to RAM before they are read out for the current FFT dimension. This data dependency is what limits the number of FFT Pipelines in the current design and hence the overall latency of 3D FFT calculation as a whole.

The D1 and D2 phases are straightforward, but the D3 phase imposes an additional timing requirement on the prior two phases. The reason is that the third phase operates on data that spans multiple RAMs and each FFT requires data from the same RAM on the same clock cycle. The solution is to skew the data driven to each FFT Pipeline so that only a single point of data is required from any particular RAM in any given cycle. When the skewing is propagated to the prior phases, it does not change the data flow control but merely skews it by the same amount as what it is in the third phase. The penalty for skewing the data is equal to the number of IPs and therefore minor; it only adds cycles for the data to fill up and drain out, which is negligible over the entire calculation. Otherwise all of the FFT Pipelines stay completely saturated.

III. Results

Design method. We have created a 3D FFT generator that allows us to parameterize designs by problem size and by number of 1D FFT IPs (and RAMs). Varying the number of IPs per problems size allows us to examine the trade off between total cycles and cycle time, the latter becoming a consideration as the chip is filled. The design has gone through one iteration of optimization with registers being

inserted in the critical path (controller). The most complex part of the generator is for the controller microcode (see [13] for details). We have synthesized a number of instances for both Xilinx Virtex and Altera Stratix product lines, some of which are described here.

Target hardware. We target two FPGA platforms for detailed study. The first is a Gidel PROCStar-III 260E-4AP development board with four Altera Stratix-III EP3ES260-F1152C2 FPGAs of which one is used. This implementation is used to demonstrate a working version, to fully validate the design, and to demonstrate a performance trend both across device vendors and generations of process technology. The second is the Xilinx Virtex-7 xc7v2000t-lflg1925. This is a large, new device built with a 28nm process. We use the Virtex-7 to demonstrate performance on current technology. Results for the Virtex-7 are from simulation and post place-and-route. We have also synthesized designs for a number of other FPGAs–in particular, the Stratix-V from Altera and Virtex-6 from Xilinx–and obtained results in line with those presented here.

Tools. For the Xilinx parts we used the Xilinx ISE design suite for simulation, synthesis, and mapping. This contains all of the Xilinx FPGA synthesis and targeting tools as well as the ISIM mixed language simulator and the LogiCORE IP core generator [5]. For Altera we used Quartus II design software for synthesis and mapping and Modelsim SE for simulation. Quartus II contains all of the Altera FPGA synthesis and P&R tools as well as the MegaCore IP generator [14]. For the GIDEL board the design was compiled with Quartus II tool chain and the bit file downloaded onto the board through Gidel's ProcWizard tool [15].

Validation. For the Gidel/Altera version we compared the results from the FPGA board with Matlab. The maximum relative difference was less than 0.008%. For the Virtex-7, running a full structural simulation is impractical. Instead we validated the overall designs using cycle accurate behavioral versions of the 1D IPs. These in turn were validated with respect to the structural versions which themselves were validated with respect to Matlab.

Results. Results are shown in Tables I and II. For the Virtex-7 each FFT size was implemented using various numbers of 1D FFT IPs. Designs with more IPs were also generated but either did not fit on chip or had very poor cycle times. Basic optimization was performed by inserting registers into critical paths. For the 32^3 FFT with 32 IPs this reduced the cycle time from 7.5ns to the 5.6ns shown. A similar optimization had little effect on the 64^3 64 IP design, probably because with high resource utilization there are multiple critical paths. Overall, since the IP blocks on their own run at 300MHz there should be substantial room for improvement with floor planning. We also generated results for fixed point FFTs. These showed little improvement over the floating point versions.

Comparison. We compare the results from the FPGA FFTs with those of sample CPUs, GPUs, and ASICs and present them Table III. We compare two cohorts of compute devices corresponding roughly to 2008- and 2012-era, respectively. For CPU and GPU we ran vendor library functions (from MKL [16] and CUFFT [17]) on the platforms shown. While there is a substantial literature on optimizing FFTs for CPUs and GPUs, we believe that these packages give (at least) close to the best available performance single devices. For MKL we note that performance is close to the theoretical peak. For CUFFT we note that for the 64^3 FFT the relative performance with respect to the analogous MKL FFT is in line with previously published ratios [18]. For CUFFT we note that other reported implementations (e.g., [19]) are not publicly available and that CUFFT has been substantially updated since the last published comparisons. ASIC results are from Anton [12] a 512-node ASIC-based dedicated MD compute engine. These results are not representative of the best possible on a single ASIC but rather are shown because of the high profile of that project and its similar goals.

IV. Discussion and Future Work

With the continued increases in device density ever larger problems fit on chip. In this study we observe that a class of 3D FFTs that dominates an important domain now fits entirely on a high-end FPGA. As expected this results in an order-of-magnitude improvement in performance over previous FPGA implementations. We also note that for 16^3 FFTs FPGAs yield substantially better performance than CPUs and GPUs and that this trend has carried across multiple process generations. For 32^3 FFTs FPGAs remain competitive with GPUs.

This work is part of a project that is exploring FPGA-centric clusters with direct connections among FPGAs through the multi-gigabit tranceivers. The work by DE Shaw has shown how effective low-latency communication can be to achieve strong scaling, particularly in MD. The significance of the current work is that it demonstrates two things: (i) a design that can scale to take additional inputs/outputs directly from the MGTs and (ii) performance that indicates that the long range force will not be on the critical path for MD on such systems.

References

[1] P. D'Alberto, P. Milder, A. Sandryhaila, F. Franchetti, J. Hoe, J. Moura, M. Pueschel, and J. Johnson, "Generating FPGA-Accelerated DFT Libraries," in *Proc. IEEE Symp. on Field Programmable Custom Computing Machines*, 2007.

[2] C. Dick, "Computing Multidimensional DFTs Using Xilinx FPGAs," in *8th Int. Conf. Signal Processing Applications and Technology*, 1998.

[3] C.-L. Yu, K. Irick, C. Charkrabarti, and V. Narayanan, "Multidimensional DFT IP Generator for FPGA Platforms," *IEEE Trans. Circuits and System I*, vol. 58, no. 4, 2011.

Table I
RESULTS FOR 16^3 FFT FOR THE ALTERA STRATIX-III EP3ES260 RUN ON A GIDEL PROCSTAR III BOARD. LARGER FFTS DO NOT FIT.

FFT Size	FFT IPs	% reg	% LUTs	% BRAMs	DSPs	Cycles	Cycle Time	Latency
16^3	16	65.4%	61.2%	1.9%	33.3%	995	4.46n	4.5us

Table II
RESULTS FOR 16^3, 32^3, AND 64^3 FFTS FOR THE XILINX VIRTEX-7 XC7V2000T-1FLG1925 THROUGH PP&R.

FFT Size	FFT IPs	% reg	% LUTs	% BRAMs	DSPs	Cycles	Cycle Time	Latency
16^3	8	1.1%	1.7%	2.1%	5.9%	1916	4.0ns	7.7us
16^3	16	2.2%	3.7%	4.4%	11.8%	1149	4.6ns	5.3us
16^3	32	4.4%	9.4%	4.9%	23.7%	773	4.7ns	3.6us
32^3	8	1.3%	2.1%	14.9%	9.6%	12907	5.4ns	69.8us
32^3	16	2.6%	4.4%	15.5%	19.2%	6765	5.5ns	37.5us
32^3	32	5.2%	10.4%	14.3%	38.5%	3694	5.6ns	20.7us
64^3	16	3.0%	4.6%	89.2%	22.2%	50112	9.8ns	492.9us
64^3	32	6.0%	11.1%	89.2%	44.4%	25538	11.0ns	281.8us
64^3	64	12.0%	27.4%	84.2%	88.9%	13251	16.3ns	215.9us

Table III
RESULTS FOR VARIOUS TECHNOLOGIES AND PROBLEM SIZES. ANTON IS FIXED POINT, OTHERWISE RESULTS ARE FOR SINGLE PRECISION FLOATING POINT. ALL TIMES ARE IN MICROSECONDS. RELEASE DATE IS FROM CORPORATE ANNOUNCEMENTS OF AVAILABILITY IN QUANTITY. VIRTEX-7 TIMES ARE FROM PP&R. ANTON RESULTS ARE FROM [12]. ALL OTHERS ARE FROM RUNS BY THE AUTHORS.

Implementation Technology									Performance in μs		
Tech.	Make	Model	Parallelism	Part #	Proc.	Freq.	Rel. Date	Code	16^3	32^3	64^3
2008 era technology											
CPU	Intel	Nehalem	4 cores	E5530	45nm	1.6GHz	2009/Q1	MKL	38	116	983
GPU	NVIDIA	Tesla	240 SPs	C1060	55nm	1.3GHz	2008/Q3	CUFFT	54	66	257
FPGA	Altera	Stratix-III	16 1D FFTs	EP3ES260	65nm	0.22GHz	2008/Q2	here	4.5	DNFit	DNFit
ASIC	DE Shaw	Anton	512 PEs	——-	90nm	0.8GHz	2008/Q3	report	Not Av.	4	13
2012 era technology											
CPU	Intel	Sandy Bridge	8 cores	E5-2680	32nm	2.7GHz	2012/Q1	MKL	22	55	288
GPU	NVIDIA	Kepler	2688 SPXs	Tesla K20c	28nm	0.73GHz	2012/Q4	CUFFT	25	29	92
FPGA	Xilinx	Virtex-7	various	XC7v2000	28nm	various	2012/Q2	here	3.6	21	216

[4] Altera, "FFT MegaCore Function: User Guide," http://www.altera.com/ literature/ug/ ug_fft.pdf accessed 1/18/2014, 2014.

[5] Xilinx, "LogiCORE IP Fast Fourier Transform v9.0: Product Guide for Vivado Design Suite," http://www.xilinx.com/ support/documentation/ip_documentation/ xfft/v9_0//pg109-xfft.pdf accessed 10/26/2013, 2014.

[6] S. Lee, "An FPGA Implementation of the Smooth Particle Mesh Ewald Reciprocal Sum Compute Engine (RSCE)," Master's thesis, University of Toronto, 2005.

[7] T. Sasaki, K. Betsuyaku, T. Higuchi, and U. Nagashima, "Reconfigurable 3D-FFT Processor for the Car-Parrinello Method," *Journal of Computer Chemistry, Japan*, vol. 4, no. 4, pp. 147–154, 2005.

[8] B. Akin, P. Milder, F. Franchetti, and J. Hoe, "Memory Bandwidth Efficient Two-Dimensional Fast Fourier Transform Algorith and Implementation for Large Problem Sizes," in *Proc. IEEE Symp. on Field Programmable Custom Computing Machines*, 2012.

[9] M. Chiu and M. Herbordt, "Molecular dynamics simulations on high performance reconfigurable computing systems," *ACM Trans. on Reconfigurable Technology and Systems*, vol. 3, no. 4, pp. 1–37, 2010.

[10] Y. Gu and M. Herbordt, "FPGA-based multigrid computations for molecular dynamics simulations," in *Proc. IEEE Symp. on Field Programmable Custom Computing Machines*, 2007, pp. 117–126.

[11] ——, "Amenability of multigrid computations to FPGA-based acceleration," in *Proc. High Performance Embedded Computing Workshop*, 2007.

[12] C. Young, J. Bank, R. Dror, J. Grossman, J. Salmon, and D. Shaw, "A 32x32x32, spatially distributed 3D FFT in four microseconds on Anton," in *SC '09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, 2009, pp. 1–11.

[13] B. Humpries, "Using Offline Routing to Implement a Low Latency 3D FFT in a Multinode FPGA System," Master's thesis, Department of Electrical and Computer Engineering, Boston University, 2013.

[14] Altera, "Quartus-II Handbook," http://www.altera.com/ literature/hb/qts/ quartusii_handbook.pdf, 2014.

[15] *ProcWizard Product Brief*, Gidel Reconfigurable Computing, http://www.gidel.com/ PROCwizard.htm, 2014.

[16] *Intel Math Kernel Library*, Intel Corporation, software.intel.com/en-us/intel-mkl, Accessed 4/2014.

[17] NVIDIA, "CUDA Toolkit Documentation: CUFFT," http://docs.nvidia.com/ cuda/cufft accessed 1/18/2014, 2014.

[18] V. Lee, C. Kim, J. Chhugani, M. Deisher, D. Kim, A. Nguyen, N. Satish, M. Smelyanskiy, S. Chennupaty, P. Hammarlund, R. Singhal, and P. Dubey, "Dubunking the 100x GPU vs. CPU myth: An evaluation of throughput computing on CPU and GPU," in *Proc. Int. Symp. on Computer Architecture*, 2010.

[19] A. Nukada and S. Matsuoka, "Auto-tuning 3D FFT library for CUDA GPUs," in *Proc. ACM/IEEE Int. Conf. for High Performance Computing, Networking, Storage and Analysis – Supercomputing*, 2009.