3D FFT for FPGAs *

Ben Humphries Martin C. Herbordt

Department of Electrical and Computer Engineering
Boston University; Boston, MA 02215

Abstract: The 3D FFT is critical in electrostatics computations such as those used in Molecular Dynamics simulations. 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 using less than half the chip resources, and operating at a very conservative frequency, takes less than 50us for 32^3 and 200us for 64^3 single precision data points, numbers similar to the best published for GPUs. The 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.

1. 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 [2, 4]; IP for many variations of the 1D FFT is available from Altera and Xilinx.

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). Somewhat surprisingly, although MD on FPGAs has been widely studied, we are only aware of one implementation of the 3D FFT on FPGAs: a Masters Thesis from the University of Toronto [5]. This work is now dated both in technology and assumptions – they found the bottleneck to be off-chip memory access. With current technology, most useful 3D FFTs can hold all data on chip.

Our motivation is as follows: While in previous work we have shown that the MD range-limited force can be effectively implemented on FPGAs [1], no comparable implementation exists for the long-range force. In fact, noting the difficulties with the 3D FFT at the time, we used a different approach, implementing it with multigrid [3]. Although multigrid appeared 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: 32^3 and 64^3 data points of single precision floating point [6]. Second, we take advantage of existing IP, in this case by 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 using less than half the chip resources and operating at a conservative frequency, the 3D FFT takes less than 50us for 32^3 and 200us for 64^3 single precision data points, numbers similar to the best published for GPUs. The significance is that this is sufficient to keep the FFT off of the critical path for a large fraction of possible MD simulations.

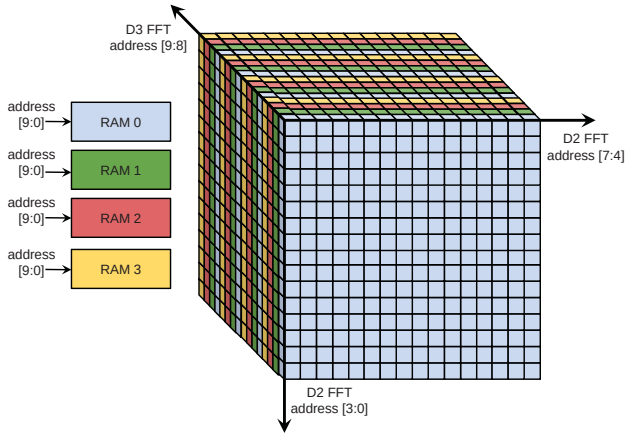

Figure 1: A possible mapping of points from a 16^3 FFT onto four Block RAMs.

2. APPROACH AND IMPLEMENTATION

Approach. Higher dimensional FFTs are decomposable into lower dimensional. Therefore the N^3 point 3D FFT can be computed by executing 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 dimension. 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. Shown in Figure 1 is 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 D1 and D2 using data only from a single RAM. Computing the FFTs for D3 requires traversing multiple RAMs, or transposing the data. We have decided to do the former routing data with a crossbar.

Design Overview. As shown in Figure 2 the design is composed of four main parts: RAMs, Crossbars, FFT Pipelines, 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 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.

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 tiny 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-

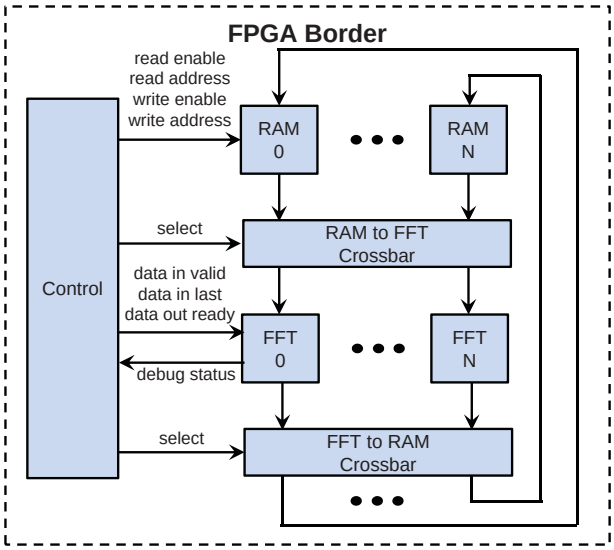

Figure 2: Block diagram for 3D FFT design.

centric clusters. This is mentioned very briefly in the discussion.

Dataflow. The 1D FFT blocks 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 IP 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 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 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.

3. RESULTS AND DISCUSSION

Methods. We used the Xilinx ISE design suite for simulation, synthesis, and mapping. The ISE suite contains all of the Xilinx FPGA synthesis and targeting tools as well as the ISIM mixed language simulator and the LogiCORE IP core generator. We target the Xilinx Virtex-7 xc7v2000t-lflg1925. This is a large, new device built with a 28nm process. But since even our largest designs use only a small fraction of the registers and LUTs we should get similar results with high-end Virtex-6 parts. Results are currently from simulation and post place-and-route. We have validated the design with a standard soft FFT.

Table 1: Results for 32^3 and 64^3 FFTs. Shown are fraction of resources used (registers, LUTs, and BRAMs), operating frequency, and overall latency. Designs have 32 and 64 1D FFT IPs, respectively.

Size	% reg	% LUTs	% BRAMs	MHz	Latency
32^3	4%	6%	19%	80	42us
64^3	10%	18%	50%	70	183us

Results. Results are shown in Table 1. Note that only a small fraction of chip resources are used. No optimization has been done on the overall design. Since the IP blocks on their own run at 300MHz there should be substantial room for improvement.

Status and work in progress. This extended abstract describes work in progress. We are currently porting this application to the Convey HC-2ex. We are also exploring a version that maximizes the IP that can fit on the FPGA. This will need more complex and control and require more care to make timing, but should result in at least somewhat better performance.

Significance. 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 tranceivers in an FPGA-centric cluster and (ii) performance that indicates that the long range force will not be on the critical path for MD on such systems.

4. REFERENCES

[1] Chiu, M., and Herbordt, M. Molecular dynamics simulations on high performance reconfigurable computing systems. *ACM Trans. on Reconfigurable Technology and Systems 3*, 4 (2010), 1–37.

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

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

[4] Jackson, P., Chan, C., Rader, C., Scalera, J., and Vai, M. A Systolic FFT Architecture for Real Time FPGA Systems. In *High Performance Embedded Computing Workshop* (2004).

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

[6] Young, C., Bank, J., Dror, R., Grossman, J., Salmon, J., and Shaw, D. 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.