International Journal on Advanced Science Engineering Information Technology

Vol.9 (2019) No. 2
ISSN: 2088-5334

Design and Evaluation of a Scalable Engine for 3D-FFT Computation in an FPGA Cluster

Roberto Ammendola[#1], Pierpaolo Loreti[*2]

[#1]NFN Roma Tor Vergata, Via della Ricerca Scientifica,1 - 00133 Roma – Italy
E-mail: roberto.ammendola@roma2.infn.it

[*2]Department of Electronic Engineering, University of Rome Tor Vergata, Via del Politecnico 1, 00133 Rome, Italy
E-mail: pierpaolo.loreti@uniroma2.it

Abstract— **The Three Dimensional Fast Fourier Transform (3D-FFT) is commonly used to solve the partial differential equations describing the system evolution in several physical phenomena, such as the motion of viscous fluids described by the Navier–Stokes equations. Simulation of such problems requires the use of a parallel High-Performance Computing architecture since the size of the problem grows with the cube of the FFT size, and the representation of the single point comprises several double precision floating-point complex numbers. Modern High-Performance Computing (HPC) systems are considering the inclusion of FPGAs as components of this computing architecture because they can combine effective hardware acceleration capabilities and dedicated communication facilities. Furthermore, the network topology can be optimized for the specific calculation that the cluster must perform, especially in the case of algorithms limited by the data exchange delay between the processors. In this paper, we explore an HPC design that uses FPGA accelerators to compute the 3DFFT. We devise a scalable FFT engine based on a custom radix-2 double-precision core that is used to implement the Decimation in Frequency version of the Cooley–Tukey FFT algorithm. The FFT engine can be adapted to different technology constraints and networking topologies by adjusting the number of cores and configuration parameters in order to minimize the overall calculation time. We compare the various possible configurations with the technological limits of available hardware. Finally, we evaluate the bandwidth required for continuous FFT execution in the APEnet toroidal mesh network.**

Keywords— **3D-FFT; FPGA; high-performance computing; cluster.**

I. INTRODUCTION

Continuous demand for efficient computing power is pushing designers into integrating dedicated hardware components in High-Performance Computing (HPC) architectures to improve computational efficiency. General-purpose CPUs can delegate specific tasks to the hardware accelerator decreasing the latency of computationally demanding task such as the training of neural networks [1]-[4], video or audio processing [5]-[8], environmental forecasting [9]-[11], security algorithms [12], automotive applications [13], etc. In other scenarios, hardware accelerators are used to reduce system power consumption [14]-[15].

Modern HPC systems are evaluating the inclusion of FPGAs as components of their system architectures because they can combine effective hardware acceleration capabilities and dedicated communication facilities in a single device. The resulting design is suitable to execute distributed tasks in computer clusters effectively. An actual example is Microsoft Catapult [16], a data center able to act as an HPC system thanks to the introduction of FPGA accelerators. In this context, FPGAs also allows optimizing the data exchange among the hardware acceleration modules thanks to the direct connections supported by the network controllers which are integrated into the programmable hardware.

A. *FFT for simulations*

A widely used algorithm in the simulations of physical phenomena is the Multidimensional FFT and in particular the 3D FFT [17] that are employed in solving the partial differential equations of physical models, such as the Navier–Stokes equations that describe the motion of viscous fluids [18] or Newtonian mechanics equations of Molecular Dynamics [19]. Simulation of such problems requires the use of HPC since the size of the problem grows rapidly and this is mainly due to three reasons:

- The dimension of the problem increases with the cube of the FFT size;
- The representation of the single data point value typically requires a double precision floating point complex number (i.e., 32, 64 or 128 bits);
- The single point usually comprises different components (for example, if the point represents the velocity it has three components per point).

Fig. 1 reports the required RAM vs. the FFT size *N* for different computer cluster configurations, starting from the single unit, up to 1024 nodes (a 32x32 bi-dimensional cluster). The required RAM per node is compared with the RAM sizes of the FPGAs development kits currently available on the market (512 MB, 2GB and 8GB). The figure shows that the single unit needs handling about 0.25 GB for N = 256 or up to 1024 GB for N = 4096.

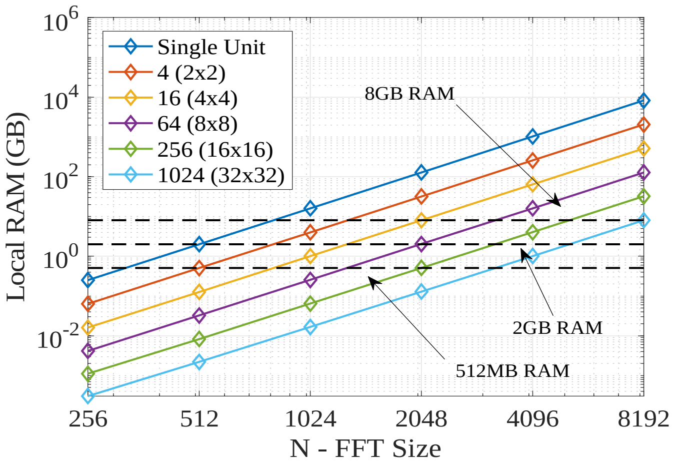

Fig. 1 Required local RAM per node for various cluster sizes increasing the FFT size.

This implies that typical physical problems have to be tackled on HPC computer clusters where multiple compute units work in parallel to distribute the data domain. For example, considering an FPGA with 8GB of local RAM such as the Xilinx UltraScale+ VU37P, we can execute an N = 8192 points 3D FFT only using 1024 FPGAs, i.e. a 32x32 cluster.

To better quantify the impact of the FFT in physics simulations, we have performed a computation of the Navier–Stokes equations for a 1024x1024x1024 volume on an increasing number of processors. The computation was performed by the NewTurb cluster [20]. In Fig. 2 we report the percentage of execution time dedicated to the FFT and inverse FFT computation and the percentage of the other computing activities. As can be noted, the percentage of time dedicated to the calculation decreases with the number of processors while the percentage of FFT computing time increases.

B. *Related Work*

Over recent years, FPGAs have been employed for several different applications such as the Internet of Things [21]-[23], wireless communications [24]-[25], network protocols [26]-[27], etc. The employment of FPGAs in HPC is relatively recent and still the subject of research and optimization. FPGAs have been initially conceived to implement custom networking infrastructure among the nodes of the cluster [28]. Only recently, the proposal of FPGAs as a hardware accelerator has appeared in some works. In [29] the authors evaluate the effectiveness of an OpenCL FPGA implementation of several algorithms and point out that FPGAs can be more energy-efficient than GPUs and sometimes faster.

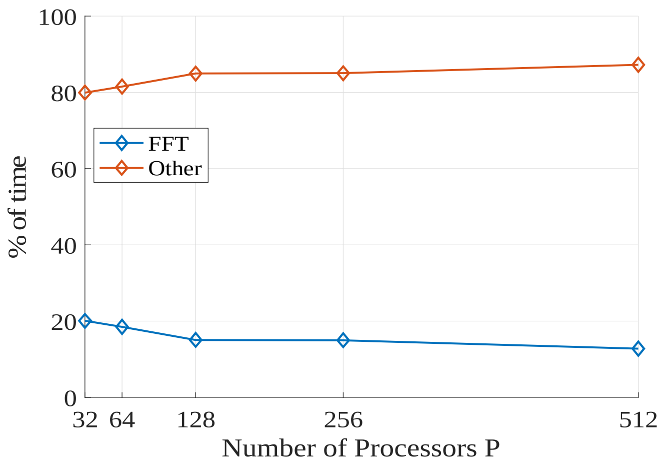

Fig. 2 Percentage of time for the FFT and the other computations for Navier-Stokes step increasing the number of processors.

The integration of FPGAs in HPC clusters in the field of 3D FFT has been discussed in [31], showing a distributed parallel implementation of a problem sized of up to 128^3 points along with a 3D-torus direct network of 8^3 FPGA-based nodes. In terms of solution time, results clearly state the viability of FPGA clusters in terms of strong scaling. Moreover, in [32] it is demonstrated how the inter-accelerator dedicated network (in this case each FPGA has a single 40Gbps link) gives a sensible increase of overall performances on the 3D FFT, even compared to a GPU cluster running cuFFT.

C. *Main goals*

In this paper, we explore a possible evolution of the HPC APEnet system. The APEnet network connections are designed to support the distributed processing effectively and are implemented by FPGA cards. Considering that modern FPGAs have the considerable processing power, the paper investigates the possibility of using the networking FPGA as a hardware accelerator for computing of the 3D FFT. For this reason, a Radix-2 module will be designed for complex double-precision numbers. Also, we design a scalable architecture for the mono-dimensional FFT computation that integrates the Radix-2 module and can be adapted to the various possible network topologies. Finally, the required bandwidth in the APEnet toroidal mesh network will be calculated to support real computation.

II. Material and Method

A. *APEnet*

The APEnet project dates back to 2004 [28] and aims at developing a network fabric, which allows assembling an HPC, cluster with direct network topologies with off-the-shelf components. Over the years several card families have

$$X[k_1,k_2,\dots,k_D] = \sum_{n_1=0}^{N-1}\left(W_N^{k_1 n_1}\sum_{n_2=0}^{N-1}\left(W_N^{k_2 n_2}\dots\sum_{n_D=0}^{N-1}\left(W_N^{k_D n_D}x[n_1,n_2,\dots,n_D]\right)\right)\right) \qquad (1)$$

Eq. (1) Multidimesional FFT definition

been developed [32]-[33], integrating on an FPGA-based NIC multiple bidirectional off-board high performance serial links, a PCIe-based interface towards the host PC and peculiar offloading features such as a Remote Direct Memory Access (RDMA) protocol for peer-to-peer (P2P) connections among Fermi-class NVIDIA GPUs that allows real zero-copy, low latency GPU-to-GPU transfers.

Nowadays FPGAs have evolved into more complex platforms with several specialized hardware capabilities, such as multicore ARM embedded processors, DSP units with floating point capabilities, large amount of high speed serial links (in the range of 100 Gbit/s) and, at the latest addition, in-package High Bandwidth Memory (HBM) which is becoming available on Xilinx Ultrascale+ devices. This memory technology allows integrating large sizes of memories (i.e. 8GB) with enormous available peak bandwidth (460GB/s) distributed along 32 AXI ports directly connected into the FPGA fabric. Such a technology may be disruptive in applications like the 3DFFT, where intermediate buffering of large data sets is critical.

B. 3D FFT

The Multidimensional FFT is an optimized implementation of the Multidimensional Discrete Fourier Transform (DFT) that is defined in Eq. (1), where $W_N = e^{-i2\pi/N}$, D is the number of dimensions, the output indices run from $k_i = 0,1,\dots,N-1$ and all the dimensions have a size of *N*. The inverse expression has a complementary expression that can be found in [17].

Analyzing Eq. (1) it can be noted that the 3D FFT can be computed as a nested sequence of monodimensional FFTs. The order in which the three dimensions are scanned is not relevant. For example, the FFT can be computed along the X direction then the transformed data can be worked on again along the Y direction and finally along the Z direction. Every transformation requires N^2 monodimensional FFTs. After the 3D transformation, data is used to perform the required processing and then antitransformed again. Thus the entire compute step involves 6xN^2 monodimensional FFTs.

C. Implementation of the monodimensional FFT

The Cooley–Tukey FFT algorithm is the most commonly used to calculate the DFT and the inverse DFT. It is a divide and conquers algorithm that recursively breaks down a DFT into many smaller DFTs. The algorithm can be applied in two versions: the decimation in time (DIT) or the decimation in frequency (DIF). In the following, we are going to use the DIF version.

The mono-dimensional FFT can be divided as follows:

$$X[k] = \sum_{n=0}^{N-1} W_N^{kn}x[n] = \sum_{n=0}^{N/2-1} W_N^{kn}x[n] + \sum_{n=N/2}^{N-1} W_N^{kn}x[n] \qquad (2)$$

$$X[k] = \sum_{n=0}^{N/2-1} W_N^{kn}x[n] + \sum_{n=0}^{N/2-1} W_N^{k(n+\frac{N}{2})}x[n+N/2] \qquad (3)$$

$$X[k] = \sum_{n=0}^{N/2-1}(x[n] + (-1)^k x[n+N/2])W_N^{kn} \qquad (4)$$

where in the last step we used $W_N^{\left(\frac{N}{2}\right)k} = (-1)^k$. If we consider the even and the odd $X[k]$ separately, we obtain:

$$X[2r] = \sum_{n=0}^{\frac{N}{2}-1}\left(x[n] + x\left[n+\frac{N}{2}\right]\right)W_{\frac{N}{2}}^{rn} \qquad (5)$$

$$X[2r+1] = \sum_{n=0}^{N/2-1}(x[n] - x[n+N/2])W_N^n W_{N/2}^{rn} \qquad (6)$$

where $r = 0,1,\dots,N/2-1$.

As can be noted, the proposed formulation allows dividing the even and the odd part of the FFT thus reducing its size of the FFT by a factor 2 every step. The simple kernel that can be defined from eq. (5) and eq. (6) is depicted in Fig. 3 and is usually called “butterfly”.

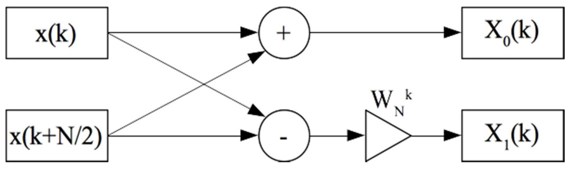

Fig. 3 Radix-2 Butterfly kernel

The butterfly kernel combines the sample *k* and the sample *k+N/2* to produce two outputs, which can be processed separately in the next step. The complete FFT can be built from this basic butterfly kernel, building a structure of N/2 butterflies at each stage for a total of $log_2(N)$ stages. In Fig. 4 we show an example of an 8-point FFT. Each stage includes $N/2 = 4$ butterfly units and the total number of stages is $log_2(N) = 3$. Each butterfly unit operates on a pair of results from the previous stage. The distance between the two samples selected decreases by a factor two every stage. Finally, the output values have to be reordered.

III. Results and Discussion

A. FPGA Radix-2 Implementation

In this section, we detail the design of the Radix-2 module depicted in Fig. 3 that will be used as the main building block for the full monodimensional FFT implementation. Writing explicitly the real and imaginary parts of the equations representing the inner butterfly FFT operations, we have:

$$\mathfrak{Re}(X_i) = \mathfrak{Re}(x_i) + \mathfrak{Re}(x_j) \tag{7}$$
$$\mathfrak{Im}(X_i) = \mathfrak{Im}(x_i) + \mathfrak{Im}(x_j) \tag{8}$$
$$\mathfrak{Re}(X_j) = \mathfrak{Re}(\mathrm{W})\left(\mathfrak{Re}(x_i) - \mathfrak{Re}(x_j)\right) - \mathfrak{Im}(\mathrm{W})\left(\mathfrak{Im}(x_i) - \mathfrak{Im}(x_j)\right) \tag{9}$$
$$\mathfrak{Im}(X_j) = \mathfrak{Im}(\mathrm{W})\left(\mathfrak{Re}(x_i) - \mathfrak{Re}(x_j)\right) + \mathfrak{Re}(\mathrm{W})\left(\mathfrak{Im}(x_i) - \mathfrak{Im}(x_j)\right) \tag{10}$$

where $x_i$and x_j represent the butterfly inputs, X_i and X_j the outputs and W is the twiddle factor. A direct hardware implementation of these equations for FPGA is reported in Fig. 5, where the complex operations are broken up in double precision adders and multipliers operating on the real and the imaginary components.

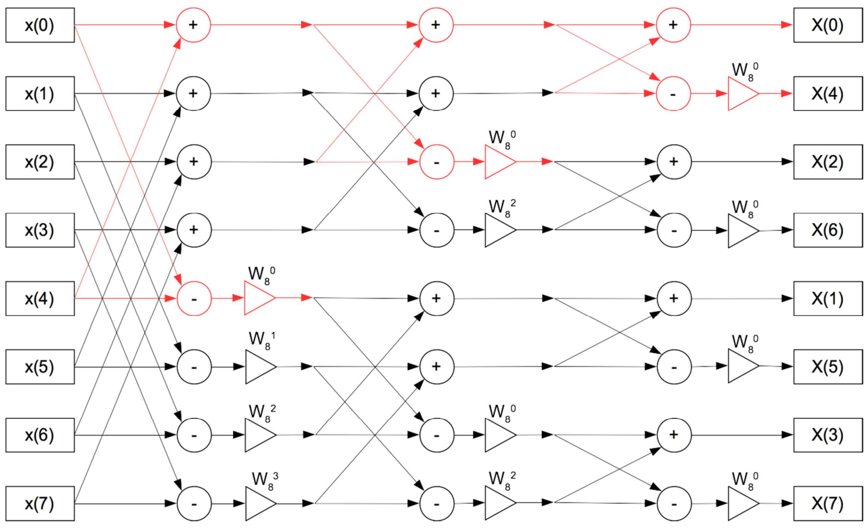

Fig. 4 Example of the 8 point FFT with DIF algorithm

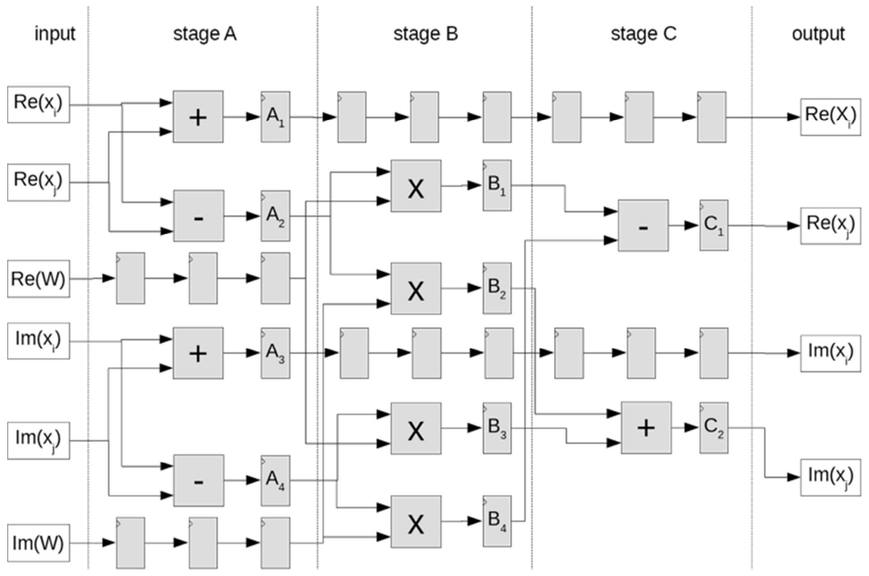

Fig. 5 FPGA implementation of the Radix-2 Butterfly kernel using three separated computational stages with parametric latency.

To maximize data throughput, the FPGA uses concurrent operations, for a total of six adders and four multipliers.

Other approaches are possible, as reviewed in [34], targeting a specific optimization, such as low power consumption [35], low area usage [36], or real-valued only signals [37]. In the proposed implementation, each operator has its characteristic (and parametric) internal latency and a streaming interface (in input and output ports), and they are typically available from IP libraries of the FPGA vendors, even in a double precision format with full IEEE-754 standard compliance. Thus, a balanced chain of operators perfectly fits into a pipelined architecture, where after the initial latency FFT outputs are available at each clock cycle.

Computing is organized in three stages. In Stage A, the following operations are performed:

$$\mathrm{A}_1 = \mathfrak{Re}(x_i) + \mathfrak{Re}(x_j), \tag{11}$$

$$A_2 = \mathfrak{Re}(x_i) - \mathfrak{Re}(x_j), \quad (12)$$
$$A_3 = \mathfrak{Im}(x_i) + \mathfrak{Im}(x_j), \quad (13)$$
$$A_4 = \mathfrak{Im}(x_i) - \mathfrak{Im}(x_j). \quad (14)$$

$\mathfrak{Re}(W)$ and $\mathfrak{Im}(W)$ are delivered to stage B by a register chain with the same length of the latency of the adders.

In Stage B, the following operations are performed:

$$B_1 = A_2\mathfrak{Re}(W),$$
$$B_2 = A_4\mathfrak{Im}(W),$$
$$B_3 = A_2\mathfrak{Im}(W),$$
$$B_4 = A_4\mathfrak{Re}(W),$$

A_1 and A_3 are delayed to the output stage on register chains. Finally, in Stage C, the following operations are performed:

$$C_1 = B_1 - B_4$$
$$C_2 = B_2 + B_3,$$

and the results are delivered to the output.

Among these stages, data is registered to ease resource placement and achieve a higher frequency. Adders have a programmable latency among 0 and 14 clock cycles and multipliers among 0 and 12, and they are all configured to allow operands to be applied on every clock cycle (one cycle per operation). The computational complexity of this radix-2 butterfly implementation is 10 Floating Point operations per cycle. Defining the latency of the three stages l_A, l_B and l_C, then the total latency of the butterfly operation block is:

$$l_{tot} = l_A + l_B + l_C + 4 \quad (15)$$

where the constant value 4 is the number of registration cycles among the 3 stages. Varying l_{tot} has an impact on working clock period T_c, thus making it an adaptable parameter.

B. Scalable FFT Implementation

The full hardware of the monodimensional FFT includes a matrix of $N/2 \cdot log_2(N)$ Radix-2 modules and $log_2(N)$ swap modules as shown in Fig. 6. The full implementation is clearly the most efficient from the computational point of view, because it can theoretically allow the computing of one monodimensional FFT for clock tick T_c.

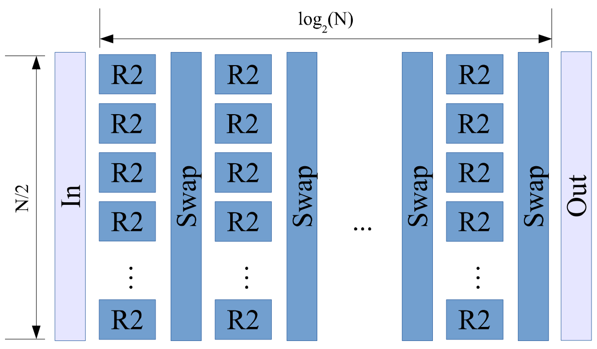

Fig. 6 Full monodimensional FFT implementation

However, the hardware components required for a full implementation rapidly exceed the available resources in the current available FPGAs, and thus alternative designs have to be considered. Moreover, since data is spread over the whole cluster to parallelize the computation, therefore, nodes periodically need to exchange data, the required bandwidth among them turns out to be too high for the available communication technologies so that the processors would spend most of the time idling.

To decrease the total number of radix-2 computing units required for the single FFT, we consider two strategies: i) to reduce C, the number of columns with $C \leq log_2(N)$ or ii) to reduce R, the number of rows in the computation matrix with $R \leq N/2$. To explain the basic design required by the two solutions, in Fig. 7 and Fig. 8 we depict the two extreme cases: $C = 1$ and $R = 1$ respectively.

The single column solution uses $N/2$ radix-2 units reused $log_2(N)$ times (see Fig. 7). Thus, the total FFT execution time results in$T_c log_2(N)$. More generally, if the number of columns of the FFT is C, the FFT execution time is$T_c log_2(N)/C$. Considering the single row solution in Fig. 8, we have that every FFT step needs repeating N/2 times. Moreover, to feed the next step we need to wait for all the $N/2$ computations buffering the intermediates values. The total resulting execution time in case of R rows results in $T_c(N/2)/R$.

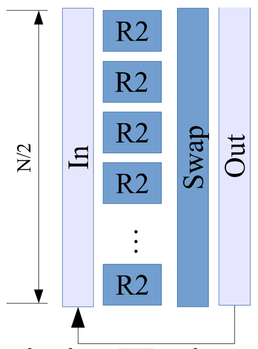

Fig. 7 Single column FFT implementation

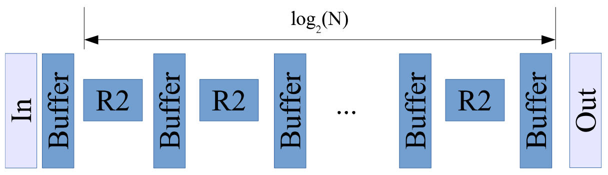

Fig. 8 Single row FFT implementation

It is possible to implement a general solution with R rows and C columns. In this case, the total FFT execution time is:

$$T_{FFT} = T_c \left(\frac{N}{2}\right) log_2(N)/(RC) \quad (16)$$

C. Radix2 Performance

The proposed double precision FFT engine in the case of $R = 1$ and $C = log_2(N)$ has been implemented on a Xilinx Virtex Ultrascale+ VU37P device for full synthesis and simulation with Vivado 2018.3 version. In Table I, we compare hardware resource usage for $N = 1024$ and

$N = 4096$ and setting the latency of the floating point operators to 3 cycles.

TABLE I
RESOURCE USAGE AND HARDWARE CHARACTERIZATION OF A SINGLE FFT ENGINE

	N=1024	N=4096
% LUTs	4.03	4.92
% REGs	1.66	2.07
% BRAM	4.86	20.83
% DSP	5.10	6.12
Cycles	5260	24744
Frequency (MHz)	259	243
Exec. time (μs)	20.31	101.83
GFLOPS	25.9	29.16
Power (W)	5.606	5.167

We can observe good scalability in terms of Slice LUTs, Slice Registers, and DSP blocks. Block RAM usage is relatively high. It can be further optimized or to reduce the memory impact when scaling to larger N, a possible solution is to use different types of memory resources, such as distributed memories or UltraRAM blocks.

In terms of energy efficiency, we estimated the power consumption using Xilinx Power Analyzer tool: the single FFT engine implemented is capable of 4.62 GFLOPS/Watt for the $N = 1024$ case and 5.64 GFLOPS/Watt for the $N = 4096$ case. This preliminary result indicates that the FPGA can be considered a viable solution also for highly power efficient computing.

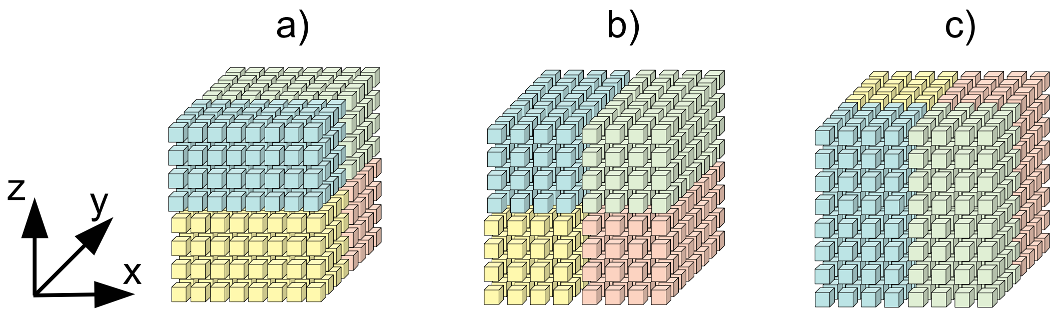

Fig. 9 Example of 2D decomposition with P processing nodes a) in the X direction, b) in the Y direction and c) in the Z direction.

In Table 2 we report the results of several synthesis attempts with different floating-point operators' latency. We note that increasing the overall butterfly latency; we can achieve a higher working frequency of the engine. In our scenario, the total execution time is more affected by the working frequency rather than engine latency.

TABLE II
BUTTERFLY LATENCY VERSUS SYNTHESIZABLE FREQUENCY FOR AN FFT OF N=1024

operator latency	l_{tot}	frequency (MHz)
1	4	101
3	13	259
6	22	359
9	31	375
12 (14)	44	380

D. Network required capacity

To perform a distributed 3D FFT, data has to be divided among the available processors that compute the monodimensional FFTs. The most efficient decomposition strategies are the 1D and the 2D [38]. The cluster processing efficiency depends on the number of processors available and the size of the problem: if the nodes are relatively few it is better to use the 1D strategy; on the contrary, if the number is high the 2D strategy becomes convenient. In both decompositions, the nodes process the local data and then need to exchange a part of to compute the FFTs in along other dimensions.

Since we are targeting large clusters, in the following we assume to be using the 2D partition strategy that is depicted in Fig. 9. Each node has to compute a pencil of the full cube in the three directions X, Y, and Z. The processing nodes are typically arranged in a 2D network, and we assume a switch-less torus topology typical of parallel computer systems [33].

In the 2D decomposition, the data is only exchanged among the nodes of the same row in the XY transposition and the nodes of the same column in the YZ transposition [38]. If we use a total of P processing units in the 2D cluster, each node executes $N^2/\sqrt{P}$ FFTs and needs exchanging $16\,N/\sqrt{P}$ bytes with every other node on the row/column for every FFT (where 16 bytes = 2*64 bits is the representation of the single point). Since we have $\sqrt{P}$ processors per row/column, the total amount of bytes that we need to exchange in a single link in the torus network is $16\,N\sqrt{P}/2$ for every FFT.

In Fig. 10 we report the network throughput of the single torus link with increasing P, the number of processing elements for various characteristics of the FFT engine. We also report two link bandwidth capabilities available on the target FPGA device.

We note that increasing the engine dimension, in terms of R and C, we quickly ramp up the required bandwidth to values not compatible with current link technologies so that communication delay dominates the total computation time.

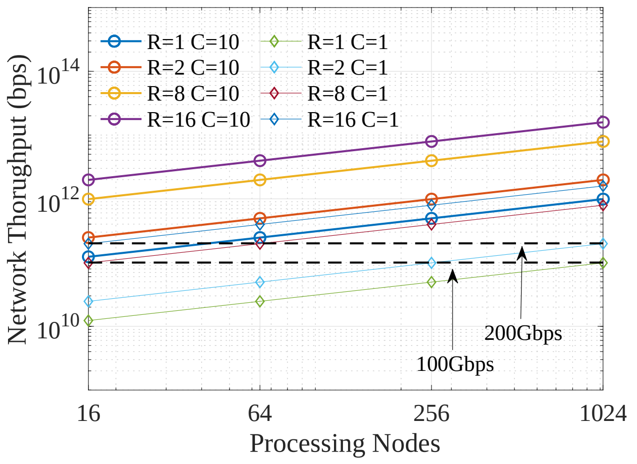

Fig. 10 Required network throughput vs. the number of processing nodes

IV. Conclusions

We presented a scalable FFT engine suitable for HPC cluster architectures that employ FPGAs as hardware accelerators and network communication. The FFT engine is based on a radix-2 double precision floating point module, specifically designed for the FFT computation. A single row version of the core has been implemented and synthesized in a Xilinx Virtex Ultrascale+ VU37P to evaluate the achievable performances. The results have been used to estimate the link bandwidth capacity required for a 2D torus network.

Acknowledgment

We would like to thank the group of Luca Biferale for the use of the NewTurb cluster.

References

[1] Giardino, D., Matta, M., Silvestri, F., Spanò, S., & Trobiani, V. “FPGA implementation of hand-written number recognition based on CNN.” 2019 *International Journal on Advanced Science, Engineering and Information Technology*, 9(1).

[2] Ismail, A. R., & Zarir, A. A., “Convolutional neural networks and deep belief networks for analyzing the imbalanced class issue in handwritten dataset” *International Journal on Advanced Science, Engineering and Information Technology*, vol. 7(6), 2302-2307, 2017.

[3] Cardarilli, G. C., Cristini, A., Di Nunzio, L., Re, M., Salerno, M., & Susi, G. “Spiking neural networks based on LIF with latency: Simulation and synchronization effects,” 2013 IEEE Asilomar Conference on Signals, Systems and Computers, 1838-1842

[4] G. C. Cardarilli, L. Di Nunzio, R. Fazzolari, D. Giardino, M. Matta, M. Patetta, M. Re, S. Spanò “Approximated computing for low power Neural Networks” 2019 Telkomnika (Telecommunication Computing Electronics and Control), 17 (3), ARTICLE IN PRESS

[5] Esposito, A., Lomuscio, A., Cardarilli, G. C., Di Nunzio, L., Fazzolari, R., Nannarelli, A., & Re, M. “Dynamically-loaded hardware libraries (HLL) technology for audio applications,” 2017 Conference Record - Asilomar Conference on Signals, Systems, and Computers, 882-886.

[6] Abhishek, S., Veni, S., & Narayanankutty, K. A. “Splines in Compressed Sensing.” *International Journal on Advanced Science, Engineering and Information Technology*, 6(4), 469-476, 2016.

[7] Tan, S. Y., Arshad, H., & Abdullah, A.. An efficient and robust mobile augmented reality application. *International Journal on Advanced Science, Engineering and Information Technology*, 8(4-2), 1672-1678, 2018.

[8] Castro, F. L., De Luca, M., & Iarossi, S. “Simulation of an Ultrasonic Flow Meter for Liquids,” *Sensors* (pp. 397-402). Springer, Cham, 2015.

[9] Waheeb, W., & Ghazali, R., “Chaotic time series forecasting using higher-order neural networks,” International Journal on Advanced Science, *Engineering and Information Technology*, 6(5), 624-629, 2016.

[10] Mustaffa, Z., Sulaiman, M. H., Rohidin, D., Ernawan, F., & Kasim, S. Time Series Predictive Analysis based on Hybridization of Meta-heuristic Algorithms. *International Journal on Advanced Science, Engineering and Information Technology*, 8(5), 1919-1925, 2018.

[11] Bostanbekov, K., Nurseitov, D., & Kim, D. Risk Assessment Model of Technogenic Pollution of the Environment from Oil Spill in the Northern Caspian Sea. *International Journal on Advanced Science, Engineering and Information Technology*, 8(1), 37-43, 2018.

[12] Lim, S. Y., Fotsing, P. T., Almasri, A., Musa, O., Kiah, M. L. M., Ang, T. F., & Ismail, R. Blockchain Technology the Identity Management and Authentication Service Disruptor: A Survey. *International Journal on Advanced Science, Engineering and Information Technology*, 8(4-2), 1735-1745, 2018.

[13] Benedetti, I., Giuliano, R., Lodovisi, C., & Mazzenga, F. “5G wireless dense access network for automotive applications: Opportunities and costs.”, In 2017 IEEE International Conference of Electrical and Electronic Technologies for Automotive (pp. 1-6).

[14] G. C. Cardarilli, L. Di Nunzio, R. Fazzolari, M. Re and R. B. Lee, "Integration of butterfly and inverse butterfly nets in embedded processors: Effects on power saving," 2012 Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR), Pacific Grove, CA, 2012, pp. 1457-1459.

[15] Cardarilli, G.C., Di Nunzio, L., Fazzolari, R., Re, M., Silvestri, F. and Spanò, S. 2018, “Energy consumption saving in embedded microprocessors using hardware accelerators”, Telkomnika (Telecommunication Computing Electronics and Control), vol. 16, no. 3, pp. 1019-1026.

[16] Caulfield, A. M., Chung, E. S., Putnam, A., Angepat, H., Fowers, J., Haselman, M. “A cloud-scale acceleration architecture.” In 2016 IEEE/ACM International Symposium on Microarchitecture (p. 7)

[17] https://en.wikipedia.org/wiki/Fast_Fourier_transform

[18] Vu, D. T., & Linh, N. M. “Solving Navier-Stokes Equation Using FPGA Cellular Neural Network Chi.” In 2016 International Conference on Advances in Information and Communication Technology (pp. 562-571). Springer, Cham,

[19] Chin, M., Herbordt, M. C., & Langhammer, M. “Performance potential of molecular dynamics simulations on the high-performance reconfigurable computing system.” In 2008 IEEE Second International Workshop on High-Performance Reconfigurable Computing Technology and Applications (pp. 1-10).

[20] Lanotte, A. S., Benzi, R., Malapaka, S. K., Toschi, F., & Biferale, L. Turbulence on a fractal Fourier set. *Physical review letters*, 115(26), 2015.

[21] Giuliano, R., Mazzenga, F., Neri, A., & Vegni, A. M. “Security access protocols in IoT capillary networks,” *IEEE Internet of Things Journal*, 4(3), 645-657, 2017

[22] Giuliano, R., Mazzenga, F., Neri, A., & Vegni, A. M., “Security access protocols in IoT networks with heterogenous non-IP terminals. Paper presented at the Proceedings”, IEEE International Conference on Distributed Computing in Sensor Systems, DCOSS 2014, 257-262.

[23] Bracciale, L., Loreti, P., Detti, A., Paolillo, R., & Melazzi, N. B.. “Lightweight Named Object: an ICN-based Abstraction for IoT Device Programming and Management” *IEEE Internet of Things Journal*, 2019

[24] Benedetti, I., Giuliano, R., Lodovisi, C., & Mazzenga, F., “5G wireless dense access network for automotive applications: Opportunities and costs.” In 2017 IEEE International Conference of Electrical and Electronic Technologies for Automotive (pp. 1-6).

[25] Mazzenga, F., Giuliano, R., & Vatalaro, F, “FttC-based fronthaul for 5G dense/ultra-dense access network: Performance and costs in realistic scenarios”. *Future Internet*, 9(4) 2017

[26] Detti, A., Bracciale, L., Loreti, P., Rossi, G., & Melazzi, N. B. “A cluster-based scalable router for information-centric networks,” *Computer Networks*, 142, 24-32, 2018.

[27] Detti, A., Orru, M., Paolillo, R., Rossi, G., Loreti, P., Bracciale, L., & Melazzi, N. B. “Application of information centric networking to nosql databases: the spatio-temporal use case” In 2017 IEEE International Symposium on Local and Metropolitan Area Networks LANMAN (pp. 1-6).

[28] Amendola, R., et al. "APENet: a high speed, low latency 3D interconnect network." 2004 IEEE International Conference on Cluster Computing.

[29] Muslim, F. B., Ma, L., Roozmeh, M., & Lavagno, L. “Efficient FPGA implementation of OpenCL high-performance computing applications via high-level synthesis” *IEEE Access*, 5, 2747-2762, 2017.

[30] Sheng, Jiayi, et al. "Design of 3D FFTs with FPGA clusters." *2014* IEEE High-Performance Extreme Computing Conference *(HPEC)*.

[31] Sheng, Jiayi, et al. "HPC on FPGA clouds: 3D FFTs and implications for molecular dynamics." 2017 27th International Conference on Field Programmable Logic and Applications (FPL).

[32] Amendola, Roberto, et al. "APEnet+: a 3D Torus network optimized for GPU-based HPC Systems." *Journal of Physics: Conference Series*. Vol. 396. No. 4. IOP Publishing, 2012.

[33] Amendola, Roberto, et al. "Latest generation interconnect technologies in APEnet+ networking infrastructure." *Journal of Physics: Conference Series*. Vol. 898. No. 8. IOP Publishing, 2017.

[34] Joshi, Shubhangi M. "FFT architectures: a review." *International Journal of Computer Applications* 116.7 (2015).

[35] Ayinala, Manohar, Michael Brown, and Keshab K. Parhi. "Pipelined parallel FFT architectures via folding transformation." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 20.6 (2012): 1068-1081.

[36] Garrido, Mario, Keshab K. Parhi, and Jesús Grajal. "A pipelined FFT architecture for real-valued signals." *IEEE Transactions on Circuits and Systems I: Regular Papers* 56.12 (2009): 2634-2643.

[37] Garrido, Mario, et al. "Pipelined radix-2^k feedforward FFT architectures." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 21.1 (2013): 23-32.

[38] Pekurovsky, D. (2012). P3DFFT: A framework for parallel computations of Fourier transforms in three dimensions. *SIAM Journal on Scientific Computing*, *34*(4), C192-C209.