Accelerating Discrete Fourier Transforms with Dot-product Engine

Miao Hu and John Paul Strachan

Hewlett Packard Laboratories, Palo Alto, CA 94304, USA. Email: {miao.hu, john-paul.strachan}@hpe.com

***Abstract*—Discrete Fourier Transforms (DFT) are extremely useful in signal processing. Usually they are computed with the Fast Fourier Transform (FFT) method as it reduces the computing complexity from O(N^2) to O($Nlog(N)$). However, FFT is still not powerful enough for many real-time tasks which have stringent requirements on throughput, energy efficiency and cost, such as Internet of Things (IoT). In this paper, we present a solution of computing DFT using the dot-product engine (DPE) – a one transistor one memristor (1T1M) crossbar array with hybrid peripheral circuit support. With this solution, the computing complexity is further reduced to a constant O(λ) independent of the input data size, where λ is the timing ratio of one DPE operation comparing to one real multiplication operation in digital systems.**

I. Introduction

Discrete Fourier Transform (DFT) converts the sampled signal or function from its original domain (order of time or position) to the frequency domain. It is regarded as the most important discrete transform and used to perform Fourier analysis in many practical applications including mathematics, digital signal processing and image processing [1]. At early stage people has issues to apply DFT for long signals since the direct computing complexity of DFT is $O(N^2)$, where N is the signal size. The inventions of Fast Fourier Transform (FFT) algorithms dramatically alleviate the issue as it computes the same result as DFT but does it more quickly with $O(N\log N)$ [2], [3]. This improvement is so critical for practical Fourier transform applications that in 1994 Gilbert Strang described the FFT as " the most important numerical algorithm of our lifetime"[1] and it was included in Top 10 algorithms of 20th Century by the IEEE journal Computing in Science & Engineering [4].

Although FFT is much more efficient than DFT, it is still not fast enough for many applications. For example, in fast convolution algorithms for real-time image processing, the FFT and IFFT steps are the bottlenecks of the throughput [5], [6]. Another example is Discrete Cosine Transform (DCT) for loss image compression, where FFT still consumes the most computation power comparing to quantization and other steps [7].Although tremendous effort has been put in the algorithm improvement [8], [9] and hardware implementation [10], [11] to accelerate FFT, so far no algorithms with lower computation complexity are known, and digital hardware implementations based on existing FFT algorithms are still limited by the complexity – $O(N\log N)$.

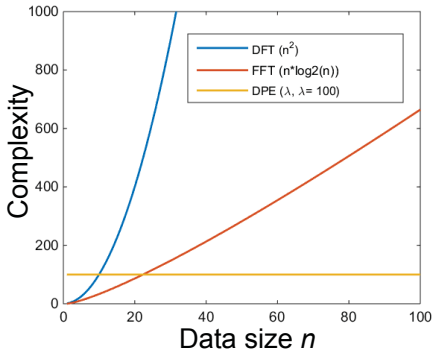

Fig. 1. Computing complexity of DFT, FFT and DPE implementation.

Here we introduce a completely new hardware implementation for DFT and other general fixed matrix multiplications in the analog domain, which can break the $O(N\log N)$ limitation and further reduce the computation time to a constant time complexity $O(\lambda)$ independent of input data size, as shown in Fig. 1. λ is the timing ratio of one Dot-product engine (DPE) operation to one real multiplication operation in digital systems [12]. Currently, DPE works at 10MHz [12], or say 100 ns per operation, from which we can make a fair assumption that $\lambda = 100$. Larger array size can contribute more RC time delay in the crossbar and limits the minimum value of λ, but it is still significantly smaller than the setup time of DAC inputs (if original input data is digital) and the reading delay of ADC, which together limits the DPE from going faster than 100 MHz ($\lambda = 10$). In short, the array size is not the bottleneck of λ, while the peripheral circuit is.

This implementation uses DPE as its fundamental building block, and a circuit structure – DPE cluster to organize multiple memristor crossbars for more complex computation is also proposed. The major speed-up comes from using the physical structure of memristor crossbars to approximate matrix-vector multiplications: by applying a vector of voltage signals to the rows of a memristor crossbar, multiplication by each memristor element's conductance is carried out by the KCL rule and the current is summed across each column. DPE automatically converts and programs the matrix values appropriately to the memristor element's conductance in order to compensate parasitics in the circuit, and by translating the input vector to input voltage signals by Digital Analog Converters (DAC), we can directly get our dot-product result by sensing the output current with Analog Digital Converters (ADC).

978-1-5090-1370-8/16/$31.00 ©2016 IEEE

To evaluate the performance, we employ one dimensional DFT as the benchmark. A real-life scenario is also simulated for practical application interest, where a remote surveillance camera compresses its color image with only 6 memristor crossbars. The result shows our new hardware implementation with DPE can have great potential in low-power signal processing, filtering and Internet of things(IoT) applications.

II. Preliminary

A. Discrete Cosine Transforms

Discrete cosine transform(DCT) is a Fourier-related transform similar to DFT, but using only real numbers [13]. Comparing to DFT, DCT has two strong advantages: first, it is much easier to compute, second and more important, it has nice energy compaction. Energy compaction is the ability to pack the energy of the spatial sequence into as few frequency coefficients as possible, and this is very important for image compression – if compaction is high, we only need to transmit a few coefficients instead of all the pixels. In this work we use DCT to calculate DFT result since DCT can be regarded as DFT with double the length for real values. For simplicity, here we only demo real value computations since complex value computations can also be realized by real value computations.

B. Dot-Product Engine

Dot-product engine (DPE) is an accelerator for approximated vector-matrix multiplications with one transistor one memristor(1T1M) memristor crossbar arrays. The reason of using 1T1M instead of 1M or one selector one memristor(1S1M) is because access transistor at each cross-point can enable precise control of the memristor states, and transistor has better linear ON states and less variabilities than other existing selector solutions.

One key difference of DPE from previous crossbar-based computing engines is the conversion algorithm that finds the correct mapping between mathematical calculations and circuit operations to overcome the known circuit issues and other limitations to maximize computing accuracy. The mapping is currently run on an external computer but can be embedded on-chip for higher efficiency. With conservative assumptions, a 1T1M crossbar simulation platform has been built and calibrated to experimental data. The performance and efficiency of the DPE is compared to a state-of-the-art ASIC. The result shows the DPE can have 1,000 to 10,000 better speed-energy efficiency product than the ASIC using 512×512 crossbars. In previous work, DPE's application in neuromorphic computing [12] and scientific computations [14] has been discussed, and here we extend its usage to more general matrix computing applications like DFT.

III. Dot-Product Engine Cluster

A. Structure of DPE cluster

The original DPE module focuses on its primary function of doing matrix vector multiplication with one multi-channel DAC/ADC and one memristor crossbar. Although complex matrix-based computations can be realized with different combinations of the primal DPE module, more compact and efficient implementations can be realized with more sophisticated peripheral circuit design. Fig. 2 shows the diagram of DPE cluster, which organize multiple memristor crossbars instead of one for better efficiency and area. Since DAC/ADC are the most area and power hungry components in the design, it only includes one set of multi-channel DAC/ADC as drive/read circuit for necessary device programming and off-chip communication. A small MCU is embedded in the chip for crossbar selection, buffer selection and matrix conversion (if needed). Crossbar selection & buffer controls are latches that enable/disable crossbars and buffers. Buffers consist of analog amplifiers that can provide direct signal propagation from one crossbar to other enabled crossbars. Last but not least, a multi-channel MUX is also necessary to program bipolar memristor devices with access transistors. Although drive circuit can provide bipolar voltages, NMOS access transistors usually could not satisfy the RESET current requirement in the negative voltage region. With the help of MUX, some useful functions can also be realized in a neat way.

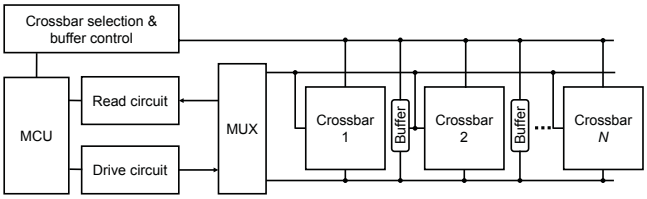

Fig. 2. Dot-product engine cluster

B. Matrix-based computations on DPE cluster

With the help of DPE cluster, we can realize and not limited to the following matrix-based computations much more efficiently than using DPE modules.

- **Sequential matrix multiplication** needs the input vector to multiply with multiple matrices sequentially. To realize this, we can enable the buffers and crossbars in the correct order to form the required data flow. This method can reduce the need of DAC/ADC to only the begin and the end stages of the computation. However, it should be noticed that noise will accumulate in analog signal propagations and intermediate DAC/ADC steps are required for noise cancellation.
- **Parallel matrix multiplication** needs the vector to multiply with multiple matrices in parallel. To realize this, selected crossbars should be enabled and final result can be sensed at one time step instead of selecting and accumulating output result from different crossbars.
- **Iterative matrix multiplication** needs the vector to iteratively multiply with the same matrix until certain threshold or condition in the result is triggered. This computation is extensively used in many iterative neural networks, like the Brain-state-in-a-box (BSB) neural network for pattern association and recognition [15]. By just enabling the buffer adjacent to the selected crossbar, we can easily realize the computation in DPE cluster.

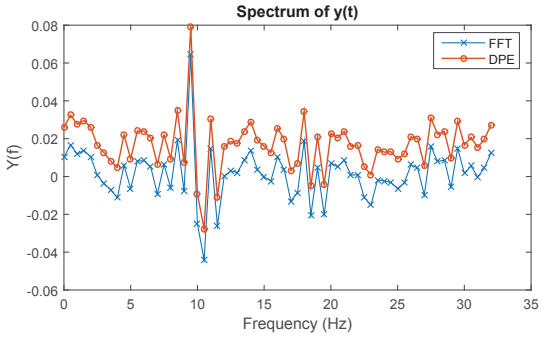

Fig. 3. DFT result with signal size = 128 using FFT method or DPE cluster

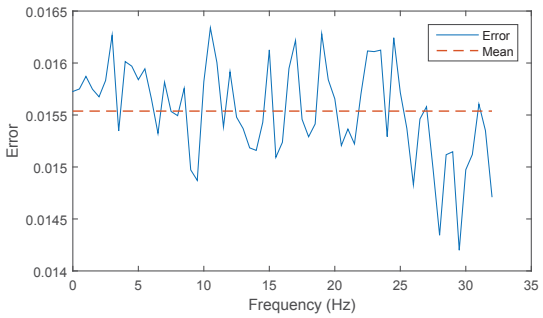

Fig. 4. Error of DFT result with signal size = 128 using DPE cluster

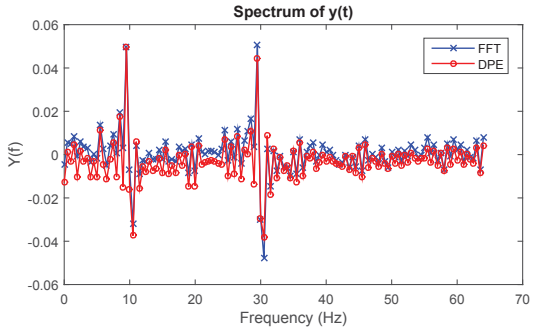

Fig. 5. DFT result with signal size = 256 using FFT method or DPE cluster

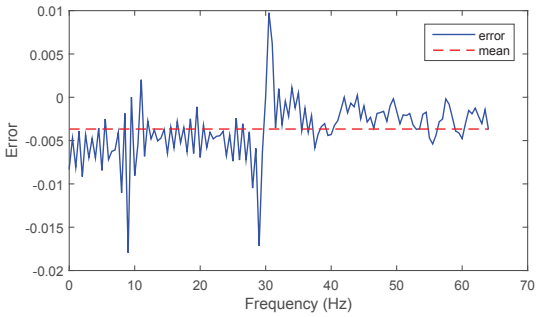

Fig. 6. Error of DFT result with signal size = 256 using DPE cluster

Meanwhile, the saturation/idle status of amplifiers in the buffer can also be configured as the threshold condition.

- **Transport matrix multiplication** needs the vector to multiply with the transport of the matrix. Generally it needs the signal to travel from columns to rows instead of rows to columns in crossbar arrays. It can be realized either with the MUX or the buffer, depends on the origination as well as the purpose of the signal(on-chip or off-chip), and the accuracy requirement. It is important because for symmetric matrix like DCT matrix, multiplying the transport equals multiplying the inverse of the matrix. It means that a DPE cluster for DCT computation can be easily used for IDCT computation by just changing of the direction of signal flow in crossbar arrays.

IV. DFT IMPLEMENTATION

In simulation we use experiment calibrated component models and Table I summaries the simplified simulation configurations [12]. For example, the transistor model is calibrated with the 0.18um transistor used in crossbar arrays, and its typical ON and OFF state resistance are measured and summarized in the table. We have noticed that the transistor ON state resistance slightly varies depending on the V_{DS} signal, and we have considered its impact in simulation. The conductance of memritor also varies because of device nonlinearity and here we use 0.2V as the read voltage amplitude for all devices to generate the final conductance matrix. The 10 Ω wire resistance in crossbar is the wire resistance between two adjacent devices. Rin and Rout in crossbar represent the input and output resistance, respectively.

TABLE I
SIMPLIFIED SIMULATION CONFIGURATIONS

Component	Simplified properties:			
Memristor	Ron(Ω) 150k	Roff(Ω) 3M	Temp.(k) 300	Vread (V) 0.2
Crossbar	wire (Ω) 10	Rin(Ω) 100	Rout(Ω) 100	Size 128/256
Transistor	Ron (Ω) ∼500	Roff(Ω) 1G		
DAC	Bit-accuracy 8	V_out (V) 0∼0.125		
ADC	Bit-accuracy 8	I_in (mA) 0∼1mA		

To implement one dimensional DFT, we actually use the conversion algorithm to convert the DCT matrix instead of DFT matrix to device conductance values. Since DCTs are equivalent to DFT of roughly twice the length, operating on real data with even symmetry, we only need to extend the data size by copying existing data. The conductance values are then programmed to one of the memristor crossbar array in the DPE cluster and keeps other crossbars un-selected. We first test the DFT performance on the 128×128 crossbar array. the data size is set to 64 and extended to 128. Then we use $0.7 \cdot \sin(2 \cdot \pi \cdot 10 \cdot t)$ as the input signal. A Gaussian noise with $\mu = 0.5, \sigma = 1$ is added to the signal. Fig. 3 shows the DFT result with FFT method and the DPE cluster implementation. Interestingly, DPE result shows fairly high consistence with the correct result (by FFT method) only with a near constant positive mean shift around 0.015, as shown in Fig. 4. This mean shift is due to usual offset issue in analog signal processing.

A similar test is also done for 256×256 crossbar array to prove the scalability. For data size = 128 and extended to 256, we use $0.7 \cdot \sin(2 \cdot \pi \cdot 10 \cdot t) + \sin(2 \cdot \pi \cdot 30 \cdot t)$ as

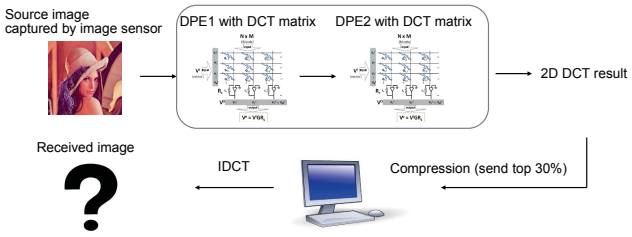

Fig. 7. Data flow in the surveillance camera scenario

the input signal this time. The same noise is also applied to the input signal. Fig. 5 shows the result. The DPE result is still very close to the correct value while the mean shift is negative in this case, as shown in Fig. 6. This may due to the impact of device nonlinearity on conversion algorithm's optimization process. Since in large arrays, device nonlinearity has a larger and more complex impact on the output result, while the conversion algorithm is always trying to minimize the overall average computing error, the mean-shift in final computing can also be changed. However, in both case, the mean-shift are consistent among all data points and can be easily removed or ignored if applications only interest at the peaks.

V. Image Compression Application

A. Wireless surveillance camera scenario

DPE-based DFT result may not be as accurate as FFT result in digital implementations, but it can still be very useful for many real-life applications which emphasize more on energy efficiency, speed and fabrication cost. An typical example is the wireless surveillance camera that captures and transfers the image data to remote computers. The image data must be processed and compressed before sending out. In this scenario 2D-DCT is used for image compression. Generally, 2D-DCT is just doing column-wise DCT on each image vector and then do a row-wise DCT on the previous DCT result. To realize this in DPE cluster, we need and only need six memristor crossbars for the three R,G,B pixel data of a color image. Then one quarter of the final DCT result, the top-left corner of the 2D-DCT result, is sent out to the computer and finally decompressed there with IDCT by FFT method.

DPE cluster has three attractive advantages in this scenario: first, since it's analog at core, it can directly process real-world analog signals captured by sensors, omitting the high power and high delay DAC step in traditional digital signal processing units; second, since the implementation only uses six crossbars, it can be extremely power efficient and high throughput; last but not least, these crossbars are still re-programmable, so more advanced functions, like image filters or even neural network for pattern recognitions can be implemented onto the DPE cluster to make the camera more smart and more powerful.

B. Result

All of the advantages mentioned above require a simple prerequisite: DPE cluster can realize 2D-DCT with enough

Fig. 8. Result of restored image. Image compression on left is completed by FFT, and on right is by DPE.

Fig. 9. Four pairs of restored images for comparison. Similar to Fig. 8, in each pair, the image compression for the right figure is completed by DPE

accuracy to generate recognizable restored figures in the end. Here the restored image from DCT and DPE are shown in Fig. 8. More examples are shown in Fig. 9. It is obvious that the images are still easy recognizable, with some loss of detail at the edges and corners. Some color shifting happens but the shape of images are well-preserved for further pattern recognition. In addition, these result are generated by a system with no optimization for image compression: quantizer and entropy encoder are not included in the design yet.

C. Error analysis

We check the data between two DCT steps, as shown in Fig. 10. At the first column-wise DCT step, image signal is the input and the output result well matches the ideal result. However, since DCT has high energy compaction and compress energy to low frequency signals, after column-wise DCT large values are concentrated to the first few rows and remaining rows are holding very small values. When a row vector of very small signals enters the memristor crossbar for row-wise DCT, its output could not be very accurate since the DPE is not calibrated to such small signals but to whole-range signals. The similar issue happens to the first few rows as well since their values are all large, but the impact is less severe since large-value rows are fewer than small value rows. An simple fix would be using two memristor crossbars at the row-wise DCT step, one is calibrated to process small signals and another is calibrated to process large signals.

Many IoT applications require fast and low power signal pre-processing on the sensor data, it needs to real-time process all the analog data provided by sensors, and just needs a modest accuracy that is enough to trigger the alarm for more advanced and more expensive data process request. Here in the wireless surveillance camera scenario, DPE cluster is good enough as the pre-processing unit to enable high throughput and high efficiency real-time image processing and compression.

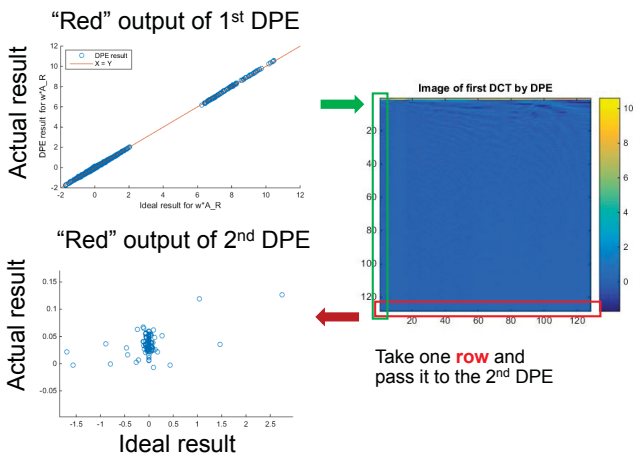

Fig. 10. Error analysis for 2D-DCT with DPE.

VI. Conclusion

In this work, we present the DPE cluster for DFT applications. It uses DPE as the fundamental building block and extend its functionality for more complex matrix computations. Comparing to existing DFT and FFT methods, it breaks the complexity barrier of O($Nlog(N)$) and reaches a constant time complexity independent of data size. We also demonstrate its performance in DFT and image compression. The result proves the potential of DPE cluster for many IoT applications.

VII. Acknowledgments

This research is based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via contract number 14080800008. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

References

[1] G. Strang, "Wavelets," *American Scientist*, vol. 82, no. 3, pp. 250–255, 1994. [Online]. Available: http://www.jstor.org/stable/29775194

[2] C. Van Loan, *Computational Frameworks for the Fast Fourier Transform*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1992.

[3] M. Heideman, D. Johnson, and C. Burrus, "Gauss and the history of the fast fourier transform," *IEEE ASSP Magazine*, vol. 1, no. 4, pp. 14–21, October 1984.

[4] J. Dongarra and F. Sullivan, "Guest editors introduction to the top 10 algorithms," *Computing in Science Engineering*, vol. 2, no. 1, pp. 22–23, Jan 2000.

[5] K. Pavel and S. David, *Algorithms for efficient computation of convolution*. INTECH Open Access Publisher, 2013.

[6] O. Fialka and M. Čadik, "Fft and convolution performance in image filtering on gpu," in *Information Visualization, 2006. IV 2006. Tenth International Conference on*. IEEE, 2006, pp. 609–614.

[7] A. B. Watson, "Image compression using the discrete cosine transform," *Mathematica journal*, vol. 4, no. 1, p. 81, 1994.

[8] H. Pang, D.-x. Li, Y.-x. Zu, and Z.-j. Wang, "An improved algorithm for harmonic analysis of power system using fft technique [j]," *Proceedings of the Csee*, vol. 6, no. 009, 2003.

[9] D. P. Kolba and T. W. Parks, "A prime factor fft algorithm using high-speed convolution," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 25, no. 4, pp. 281–294, 1977.

[10] A. Pedram, J. McCalpin, and A. Gerstlauer, "Transforming a linear algebra core to an fft accelerator," in *Application-Specific Systems, Architectures and Processors (ASAP), 2013 IEEE 24th International Conference on*. IEEE, 2013, pp. 175–184.

[11] E. Sujatha, C. Subhas, M. G. Prasad, and N. Padmaja, "A review on optimized fft/ifft architectures for ofdm systems," *i-Manager's Journal on Wireless Communication Networks*, vol. 4, no. 3, p. 33, 2015.

[12] M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, R. S. Williams, and J. Yang, "Dot-product engine for neuromorphic computing: programming 1t1m crossbar to accelerate matrix-vector multiplication," 2016.

[13] K. R. Rao and P. Yip, *Discrete cosine transform: algorithms, advantages, applications*. Academic press, 2014.

[14] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *Proc. ISCA*, 2016.

[15] M. Hu, H. Li, Q. Wu, and G. S. Rose, "Hardware realization of bsb recall function using memristor crossbar arrays," in *DAC*. ACM, 2012, pp. 498–503.