EFFICIENT SIGNAL RECONSTRUCTION VIA DISTRIBUTED LEAST SQUARE OPTIMIZATION ON A SYSTOLIC FPGA ARCHITECTURE

Muya Chang, Samantak Gangopadhyay, Tomer Hamam, Justin Romberg, and Arijit Raychowdhury

Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA

ABSTRACT

Optimization problems form the basis of a wide gamut of computationally challenging tasks in signal processing, machine learning, resource planning and so on. Out of these, convex optimization, and in particular least square optimization, covers a vast majority; and recent advances in iterative algorithms to solve such problems of large dimensions have gained traction. Multi-core designs with systolic or semi-systolic architectures can be a key enabler for implementing discrete dynamical systems and realize massively scalable architectures to solve such optimization algorithms. In this paper, we present a platform architecture implemented in programmable FPGA hardware to solve a template problem in distributed optimization, namely signal reconstruction from non-uniform sampling. This is a quintessential problem with wide-spread applications in signal processing, computational imaging etc. We expect such an architectural exploration to open up promising opportunities to solve distributed optimizations that are becoming increasingly important in real-world applications. The complete system design, mapping and optimization into an FPGA architecture as well as analysis of convergence and scalability have been presented.

Index Terms— Optimization, Distributed, Least Square, Hardware

1. INTRODUCTION

In the era of big data and machine learning over large data sets, the role of solving optimization problems is becoming ever important [1]. This is coupled with the realization that the conventional Von-Neumann machine is a poor choice for solving iterative algorithms, where large amount of data need to be periodically read and written from an external memory. In structure, a vast majority of optimization problems are iterative and involve (1) local updates of state variables with (2) near neighbor interactions to pass information until convergence is achieved. Such a computing paradigm is inspired by nature including the human brain, where local computation is coupled with near-neighbor communication to solve computationally 'hard' problems [2].

Precedence for such a computing paradigm can be found in systolic arrays of parallel computing units where each processing element is connected only to its adjacent neighbors [3]. Systolic designs take advantage of globally asynchronous and locally synchronous (GALS) architectures to reduce clocking complexity and improve design scalability [4]. This architecture finds application in wide range of problems due to modularity, design simplicity and reduced communication cost [5] [6]. In recent work, FPGA-based implementations of systolic array have been used [7–9]. However, most of the designs have been employed to solve signal processing applications such as FFTs. In this paper, we demonstrate the capability of such machines to emulate iterative dynamics with applications for solving optimization problems. In particular, we demonstrate a solution for signal reconstruction from non-uniform samples via least-square minimization.

2. OVERVIEW OF NON-UNIFORM SAMPLING AND SIGNAL RECONSTRUCTION

There are numerous applications in discrete signal processing where the process of sampling is non-uniform. It occurs when the samples cannot be collected uniformly or if samples lie in a non-uniform space [10] [11]. Following the non-uniform sampling process, it is required to reconstruct the original signal. Computerized tomography (CT) and magnetic resonance imaging (MRI) [12] are two such examples where reconstruction from non-uniform sampling is a fundamental step. In the

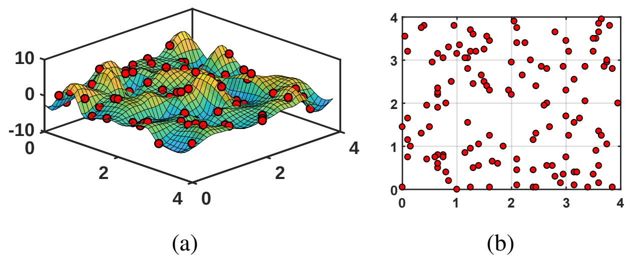


Fig. 1. (a) 2D continuous function *f(u,v)* with non-uniform samples. (b) Spatial location of the non-uniform samples.

This work was supported by a wholly-owned subsidiary of the Semiconductor Research Corporation (SRC), through Extremely Energy Efficient Collective Electronics (EXCEL), an SRC-NRI Nanoelectronics Research Initiative under Research Task ID 2698.002

following subsections, the algorithmic structure of an iterative approach to solve reconstruction from non-uniform samples is discussed.

To describe the algorithm, the following notations are used: (1) u and v are the horizontal and the vertical arguments of a continuous signal. (2) x and y are the discrete coordinate indexes. (3) ω_x and ω_y are horizontal and vertical spatial frequencies. Let $f(u, v)$ be a band-limited signal with finite energy in two dimensional real space, $\mathbb{R}^2$. The signal is non-uniformly sampled and are stored in vector $\boldsymbol{b}$, which are referred to as $f(x, y)$. The objective is to use the non-uniform samples to obtain complete reconstruction of $f(u, v)$. Fig. 1 shows an example of $f(u, v)$ and the results of non-uniform sampling.

2.1. Signal Model

To reconstruct the signal accurately, the choice of basis functions is critical. In this work, we use 2D lapped orthogonal transform (LOT) cosine-IV harmonics as the basis functions. For each frame, a set of shifted cosine-IV basis is associated. This is how the LOT cosine-IV basis is formed. Further more, a smoothing function $g(u, v)$ is applied to all the basis functions not only to avoid distortions, but also to limit the effect of the basis on distant neighbors. Equation (1) shows a general LOT cosine-IV basis function. Here, $f(u, v)$ is split into K_x by K_y frames and $[k_x, k_y]$ represent a specific frame.

$$\psi_{k_x,\omega_x,k_y,\omega_y}(u, v) = \sqrt{2} \cdot g(u - k_x, v - k_y)\cdot$$
$$cos((\omega_x + \frac{1}{2})\pi(u - k_x))cos((\omega_y + \frac{1}{2})\pi(v - k_y)) \quad (1)$$

Since $f(u, v)$ lies in a $N_x \cdot N_y$ dimensional subspace, it can be expressed as linear combination of the basis functions as (2), where α refers to the coefficients associated with the basis functions.

$$f(u, v) = \sum_{\omega_x=1}^{N_x}\sum_{\omega_y=1}^{N_y}\sum_{k_x=1}^{K_x}\sum_{k_y=1}^{K_y}\alpha(k_x, \omega_x, k_y, \omega_y)\psi_{k_x,\omega_x,k_y,\omega_y}(u, v) \quad (2)$$

2.2. Gram matrix of the basis

According to (2), we can write an equation for each sample and the coefficients can be found by solving the inverse-linear problem of

$$\boldsymbol{Az} = \boldsymbol{b} \quad (3)$$

Here $\boldsymbol{b}_{\{m,1\}}$ is the sample vector, $\boldsymbol{z}_{\{KN,1\}}$ is the coefficient vector obtained by stacking the coefficients $\alpha(k_x, \omega_x, k_y, \omega_y)$, and $\boldsymbol{A}_{\{m,KN\}}$ is referred to as the Grammian matrix.

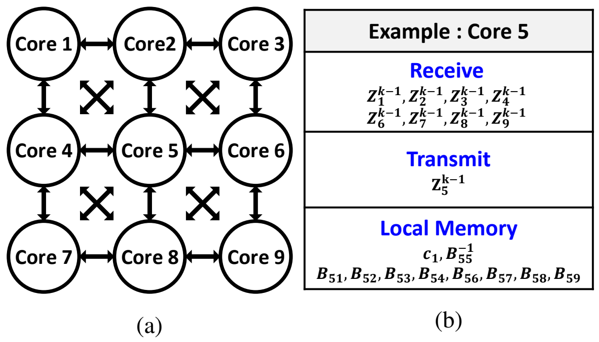
2.3. Approach

There are many ways to solve $\boldsymbol{z}$, one of the well known and the straight forward approaches would be using the pseudo-inverse of A. However, this method incurs extreme penalties when the size of $\boldsymbol{A}_{\{m,KN\}}$ matrix is large. Therefore, alternatively we follow an iterative approach, the Jacobi method. A general update of $\boldsymbol{z}$ in j^{th} component at the k_{th} iteration is given as (4), where $\boldsymbol{B} = \boldsymbol{A}^T\boldsymbol{A}$ and $\boldsymbol{c} = \boldsymbol{A}^T\boldsymbol{b}$.

$$\boldsymbol{z}_j^k = \boldsymbol{B}_{jj}^{-1}(\boldsymbol{c}_j - \sum_{i\neq j}\boldsymbol{B}_{ji}\boldsymbol{z}_i^{k-1}) \quad (4)$$

Here the inverse of matrix B is required. Since it is used locally in the computation, we pre-compute and load the values to the block ram as part of the initial values. Per-core B matrices are small and the computation of the inverse is not computationally challenging.

Some observations are worth emphasizing: (1) To update $\boldsymbol{z}_j^k$, only the values, which are $\boldsymbol{z}_j^{k-1}$, from the previous iteration are need. (2) Columns of $\boldsymbol{A}$ are coupled only with neighboring frames, which leads to simpler computation of $\boldsymbol{B}_{ji}$.



Core 1
Core2
Core 3
Core 4
Core 5
Core 6
Core 7
Core 8
Core 9
(a)
Example : Core 5
Receive
$z_1^{k-1}, z_2^{k-1}, z_3^{k-1}, z_4^{k-1}$
$z_6^{k-1}, z_7^{k-1}, z_8^{k-1}, z_9^{k-1}$
Transmit
z_5^{k-1}
Local Memory
c_1, B_{55}^{-1}
$B_{51}, B_{52}, B_{53}, B_{54}, B_{56}, B_{57}, B_{58}, B_{59}$
(b)

Fig. 2. (a) Network topology in 2D structure. (b) The received, transmitted, and local data of Core 5.

2.4. Network topology

The network topology for the 2D problem has been shown in Fig. 2. Here, each core solves a part of the coefficients $\boldsymbol{z}_j$. At k_{th} iteration each core solves a series of linear matrix equations and communicates its updated values to its neighbors, which will be further discussed in the next sections.

3. ARCHITECTURE DESIGN

The design has been synthesized and implemented on a Xilinx Virtex-7 FPGA board. Table 1 lists the specifications of the Virtex-7 FPGA board. Also an emulator has been developed in software to help us program and debug the hardware, simulate large-scaled design and perform functional verification. Further energy efficiency can be achieved by designing an ASIC [13]

Model	Logic Cells	DSP Slices	Memory(Kb)	I/O Pins
XC7VX485T2	326,400	1,120	27,000	700

Table 1. Specifications of the Virtex-7 FPGA board

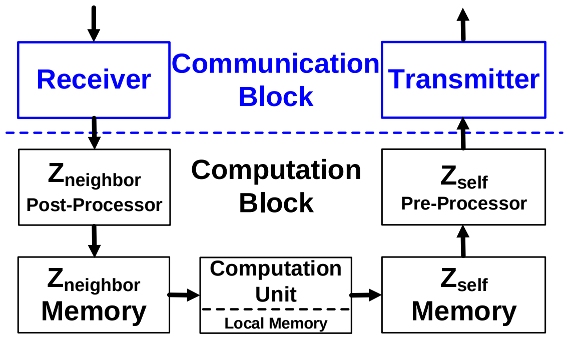


Fig. 3. Core architecture.

3.1. Top level architecture

At the top level, the system can be easily reconfigured from a 2D to a 1D architecture by turning off communication channels in both the vertical and diagonal communication routers. To avoid complex clock tree synthesis and enable scaling without incurring the cost of routing global signals, the system does not have a global clock; instead, each individual core has its own local clock following a globally asynchronous and locally synchronous (GALS) topology. The individual core operates in two phases: computation phase and communication phase. To program the system before the start of an iteration, all the constants are scanned-in and written into the local memory of each core.

3.2. Core unit design

Each core is locally synchronous and communicates with other cores through asynchronous mechanism. The structure of a core is shown in Fig. 3. To take a start from the receiver, where a core receives the updated data from the neighbors and saved into $\mathrm{Z}_{\mathrm{neighbor}}$ memory, then processed the data in the computation unit, and at the end send to the neighbors through the transmitter.

3.3. Pre/Post processing blocks

In order to reduce the communication cost, each coefficient in $\boldsymbol{z}$ is encoded before the transmission. The encoding scheme includes two steps (1) evaluate the change of coefficients in successive cycles, $\Delta \boldsymbol{z}_j$. (2) identify the location of the most significant bit (MSB). Thus, the data wires required between cores becomes $\log_2(bitprecision) + 1$. While this method will be referred to as ΔZ_{MSB} in the following discussion, the original communication method will be referred to as Z_{All}, where we send the complete $\boldsymbol{z}$.

3.4. Communication unit design

During each iteration, once the computation is done, the core enters the communication phase. In this work, 4-phase handshake protocol has been used because of the reduced logical complexity and competitive power/area efficiency. Fig. 4a shows the block-diagram of the communication unit. The implemented design uses Muller-C element to generate *Req* and *Ack* signal. Fig. 4b demonstrates the associated waveform diagram for the 4-phase handshake protocol.

4. MEASURED RESULTS

Various key aspects of the design are studied and discussed in the following subsections.

4.1. Accuracy Analysis

4.1.1. Data Resolution

The number of bit used to represent the variables is important not only because it affects the estimation, but also because it affects the hardware architecture. Here we use Q-format to represent numbers. As can be seen from Fig. 5a, the normalized error for signal reconstruction decreases dramatically after 16 bits and saturates when the bit precision reaches 32 bits.

4.1.2. Subspace dimension

Fig. 5b shows the accuracy when the number of subspace dimensions varies. Here, the original signal is synthesized under the subspace dimension of 5x5, and an interesting trend is observed. In the beginning, as subspace dimensions increase, the error reduces. However, we note a reversal of this trend in the error when the subspace dimension is larger than 5x5. This can be attributed to the over-fitting noise.

4.1.3. Convergence rate

For majority of the data-sets extremely fast convergence is observed ($<$ 10 iterations), while the time required for each

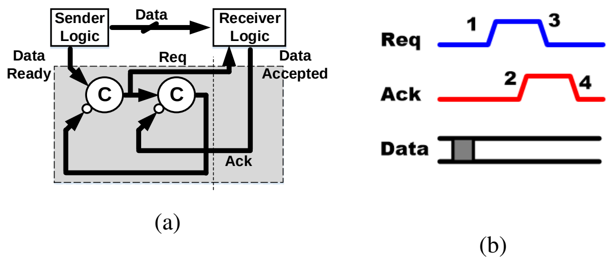


Fig. 4. 4-phase hand shaking mechanism. (a) schematic. (b) waveform.

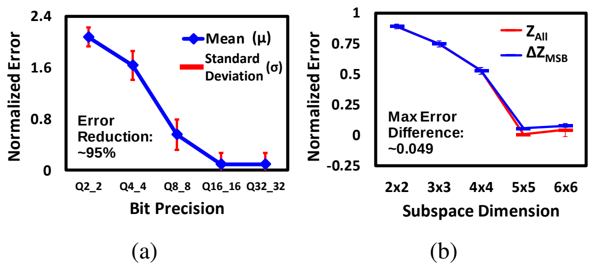


Fig. 5. (a) Measured static error versus different value of bit precision. (b) Measured static error versus different size of subspace dimensions.

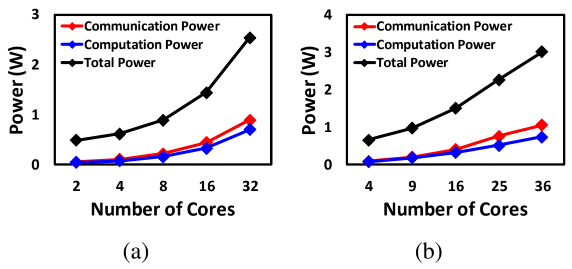


Fig. 6. Measured power consumption of system on FPGA. (a) 1D case. (b) 2D case.

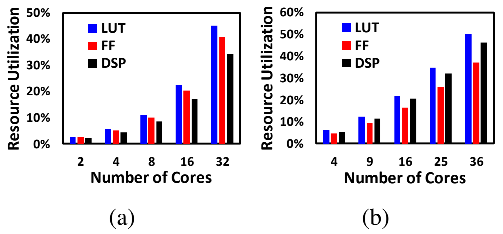


Fig. 7. Post-synthesis resource utilization on the FPGA platform. (a) 1D case. (b) 2D case.

iteration depends on the architecture choice and the clock frequency.

4.2. Design Scalability

4.2.1. Critical Delay

The effect of increasing the number of cores on performance is studied. More number of cores enable solution of larger data-sets. The distributed and systolic GALS architecture easily supports an increasing number of computational cores without affecting critical path delay and hence the maximum operating frequency. We measure for both 1D and 2D problems, the critical path increases by 7.82% when the number of cores increases by 10x.

4.2.2. Power Consumption

Fig. 6a and Fig. 6b show a linear increase in power with scaling for both 1-D and 2-D architectures. This shows an excellent scaling path for larger data-sizes. Also, as mentioned before the ΔZ_{MSB} encoding method reduces the size of the interconnects and hence the routing overhead. From measurements on the FPGA platform we note an average of 44% reduction in power consumption for both 1D and 2D archicitecures.

4.2.3. Resource Utilization

The amount of resources which the system requires also increases almost linearly as shown in Fig. 7. Here, it is important to point out that the major bottleneck for scaling is the interconnect routing inside the FPGA.

The measured results show that the modular and scalable design with the GALS architecture introduces high degree of parallelism and allows large problems with large data-sets to be mapped into the architecture. This makes co-design of iterative algorithms and systolic-architectures a powerful paradigm for solving distributed optimizations over large data-sets.

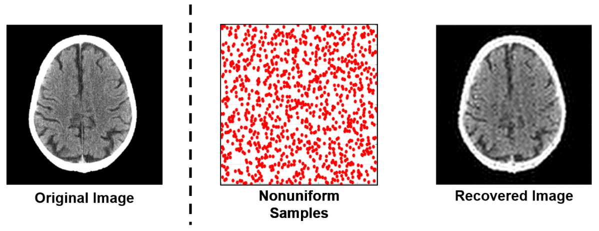


Fig. 8. 2D Example: Brain Computed Topography Recovery.

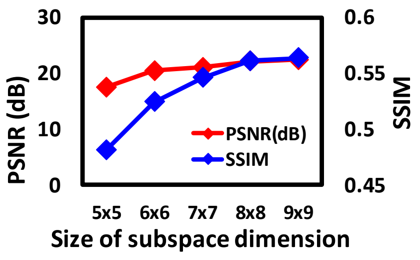


Fig. 9. Peak signal-to-noise ratio (PSNR) & Structural similarity (SSIM) of Recovery of a sampled image from the CT scan of a brain.

5. APPLICATIONS

Although this is a generic computing architecture, we present one application for the 2D architecture. In CT scans and MRIs, the acquired signal is non-uniformly sampled. As a demonstration, a 800x800 pixels CT scan image was non-uniformly sampled, divided into 8x8 blocks, and then recovered through our system, as shown in Fig. 8. The reconstruction was experimented with the architecture of 8x8 cores, and swept the subspace dimension from 5x5, 6x6, to 9x9. PSNR and SSIM for the reconstructed scan are shown in Fig. 9. Here we observe that as the size of subspace dimension increases, quality of the recovered image increases.

6. CONCLUSIONS

Iterative dynamical systems form computational models for solving distributed optimization problems. Due to the semi-separable nature, they can often be parallelized, albeit with architectural support. We propose such a distributed architecture for solving leat-square minimization, which employs a globally asynchronous design by exploiting a 4-phase handshaking mechanism between cores to eliminate unnecessary latency, bandwidth, and energy overheads associated with a global clock. The design is verified on a Xilinx FPGA and measurements reveal that the proposed architecture is scalable to large data-sets.

7. REFERENCES

[1] Z. Gui-Xia, Z. Cheng-Jing, and W. Xiao-Yan, "Research of distributed data optimization storage and statistical method in the environment of big data," in *2017 International Conference on Smart Grid and Electrical Automation (ICSGEA)*, pp. 612–617, May 2017.

[2] A. Parihar, N. Shukla, M. Jerry, S. Datta, and A. Raychowdhury, "Vertex coloring of graphs via phase dynamics of coupled oscillatory networks," *Scientific Reports*, vol. 7, 2017.

[3] A. Ibrahim, T. F. Al-Somani, and F. Gebali, "New systolic array architecture for finite field inversion," *Canadian Journal of Electrical and Computer Engineering*, vol. 40, pp. 23–30, winter 2017.

[4] M. Krstic, E. Grass, F. K. Grkaynak, and P. Vivet, "Globally asynchronous, locally synchronous circuits: Overview and outlook," *IEEE Design Test of Computers*, vol. 24, pp. 430–441, Sept 2007.

[5] H.-T. Kung, "Why systolic architectures?," *IEEE computer*, vol. 15, no. 1, pp. 37–46, 1982.

[6] S. Borkar, H. Kung, *et al.*, "Supporting systolic and memory communication in iwarp," in *Computer Architecture, 1990. Proceedings., 17th Annual International Symposium on*, pp. 70–81, IEEE, 1990.

[7] J. G. Nash, "High-throughput programmable systolic array fft architecture and fpga implementations," in *Computing, Networking and Communications (ICNC), 2014 International Conference on*, pp. 878–884, IEEE, 2014.

[8] S. D. Muñoz and J. Hormigo, "High-throughput fpga implementation of qr decomposition," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 9, pp. 861–865, 2015.

[9] J. McWhirter, "Recursive least-squares minimization using a systolic array," *Real-Time Processing VI, Aug. 1983*, pp. 105–112, 1983.

[10] P. J. S. Ferreira, "The stability of a procedure for the recovery of lost samples in band-limited signals," *Signal Processing*, vol. 40, no. 2-3, pp. 195–205, 1994.

[11] R. Marks, "Restoring lost samples from an oversampled band-limited signal," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 31, no. 3, pp. 752–755, 1983.

[12] H. Stark, "Polar, spiral, and generalized sampling and interpolation," in *Advanced Topics in Shannon Sampling and Interpolation Theory*, pp. 185–218, Springer, 1993.

[13] M. Chang, L.-H. Lin, J. Romberg, and A. Raychowdhury, "Optimo: A 65nm 270mhz 143.2mw programmable spatial-array-processor with a hierarchical multi-cast on-chip network for solving distributed optimizations," *IEEE Custom Integrated Circuits Conference*, 2019.