Improved Parallel ZF-VBLAST Detector for MIMO System

Oruba Alfawaz[1], Maha Alaa Eddin[2], Khawla A. Alnajjar[1]and Ali A. El-Moursy[2]
[1] Department of Electrical Engineering, [2] Department of Computer Engineering,
University of Sharjah, Sharjah, United Arab Emirates,
u17105772@sharjah.ac.ae, malaaeddin@sharjah.ac.ae, kalnajjar@sharjah.ac.ae and aelmoursy@sharjah.ac.ae

***Abstract*—The rapid growth of data traffic in the last decade is expected to continue in the next generation (5G) system. In order to service the high demand of data rates in 5G, complex communication architectures are needed such as multiple input multiple output (MIMO) technology. The main idea of MIMO is to increase the number of antennas in both sides (transmitter Tx and receiver Rx) that can achieve high throughput and energy efficiency. Receivers complexity and speed are some of the important requirements in MIMO systems. We focus on Vertical Bell Laboratories Layered Space Time (VBLAST) receiver. Using large number of antennas with VBLAST technique can increase the complexity of the system and the required time to recover the original signal. However, parallel computing can be used to divide the tasks of complex algorithm and distribute them among different microprocessors to run in an acceptable execution time. In this paper, we perform deep performance and time analysis to decide the opportunities of parallelization. We achieved speed up of 11.4X in the computation time of ZF-VBLAST algorithm for different sizes of MIMO systems using multi-threaded implementation.**

***Index Terms*—Vertical Bell Labs Layered Space Time (V-BLAST), multiple input multiple output (MIMO), zero forcing (ZF), parallel computing, multithreading.**

I. Introduction

The demand of high data rate, lower latencies and quality of service are increasing with every new generation of wireless communication. The 5th generation mobile technology 5G achieves better communication between anybody, anything, anytime and anywhere by using anyhow devices, networks and technologies such as multiple input multiple output (MIMO) [1]. MIMO is a promising technique to increase the capabilities of the base stations and cope with increasing data demand. The idea behind MIMO is to get large number of antennas in both transmitter and receiver [2]. A receiver receives the transmitted signal and recovers the original signal or the desired signal while reducing the effect of noise and interference. There are two types of receivers or detectors: linear and non-linear. Linear detectors are using linear detection technique to filter the received signal, such as maximum ratio combiner (MRC), zero forcing (ZF) and minimum mean square error (MMSE). While non-linear detectors are using non-linear technique such as maximum likelihood (ML) and successive interference cancelling (SIC) [3]. Increasing the number of antennas in MIMO increases the receiver complexity, hardware implementation as well as the execution time. Using parallel computing allows complex algorithms to be implemented efficiently, reduce the computational complexity and may achieve a significant speed up of the execution time in the wireless communication systems [4]. Parallel computing is a computation method, which can handle many executions or calculations at the same time [5].

Conventionally the instructions are constructed in a serial form to run on a single central processing unit (CPU) and only one instruction can be executed at a time. On the other hand, the parallel form allows many instructions to be executed at the same time using different hardware and software techniques [6]. Parallelization breaks down the task or problem into sub-tasks that can be executed simultaneously. The main objective of using parallelization is to reduce the time required to execute the instructions to solve large and complex problems. The initial steps in parallelization technique are to understand the serial algorithm, identify the hot-spots and bottlenecks of the program to distinguish the compute intensive sections. Based on that, an appropriate parallelization technique is implemented [4]. Shared memory, distributed memory and hybrid are different models of parallel processing. In shared memory, tasks share common address space where different CPUs access the same random access memory (RAM). Alternatively, distributed memory model has numbers of processors, which gives scalability to distributed memory. Hybrid model combines two different parallel programming models [7].

In [8], flex-core graphical processing unit (GPU) implementation along sphere detector with large scale MIMO achieved better detection performance than SIC. In [9], massively parallel processor array with iterative algorithm was proposed to accelerate the detection process in large scale MIMO system. The performance of the proposed algorithm was enhanced compared with the conventional detectors as shown in the results. In [10], multi core and GPU were used for the equality number of transmitter and receiver MIMO system ZF SIC detector. CUDA failed to speed up sequential version for small number of signals due to low complexity and non-parallel pattern, while achieved speed up where large MIMO are considered. CUDA increases the execution time for optimized ZF SIC. Open MP cannot parallelize ZF SIC since determined component depends on the previous computations.

The aim of this paper is to reduce the execution time of ZF-VBLAST detector for MIMO system using the parallelization techniques. However, to achieve this goal a deep performance analysis and a task dependency investigation had been done to explore the parallelization opportunities.

The rest of this paper is arranged as follows. Section II describes the system model. The parallel ZF-VBLAST is illustrated in Section III. Section IV discusses the experimental setup and results, and Section V gives the conclusions.

II. System Model

Assuming number of receivers (N_r) is serving number of transmitters (N_t), the received signal $\boldsymbol{y}$ for this uplink model is calculated by

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n} = \sum\nolimits_{\boldsymbol{m}=1}^{\boldsymbol{N}_t} \boldsymbol{h}_m \boldsymbol{x}_m + \boldsymbol{n}, \qquad (1)$$

where $\boldsymbol{H}$= $[\boldsymbol{h}_1\ \boldsymbol{h}_2\ \ldots.\ \boldsymbol{h}_{\boldsymbol{N}t}]$ is the $N_r \times N_t$ channel matrix, $\boldsymbol{x} = [\boldsymbol{x}_1\boldsymbol{x}_2\ldots\boldsymbol{x}_{N_t}]^T$ is the $N_t \times 1$ transmitted vector, $\boldsymbol{y}$ = $[\boldsymbol{y}_1\boldsymbol{y}_2\ \ldots.\ \boldsymbol{y}_{N_r}]^T$ is the $N_r\times 1$ received vector and $\boldsymbol{n} = [\boldsymbol{n}_1\boldsymbol{n}_2$ $\ldots.\ \boldsymbol{n}_{N_r}]^T$ is $N_r \times 1$ noise vector (Gaussian noise) [11]. ZF combiner requires the number of transmitted antennas (N_t) to be less than or equal to the number of received antennas (N_r). This is due to the nulling operation to eliminate the interference. The ZF detector can be calculated by

$$\boldsymbol{W} = \boldsymbol{H}(\boldsymbol{H}^{\boldsymbol{H}}\boldsymbol{H})^{-1}. \qquad (2)$$

The estimated output is

$$\boldsymbol{b} = \boldsymbol{W}^{\boldsymbol{H}}\boldsymbol{y}. \qquad (3)$$

Vertical Bell Laboratories Layered Space Time (VBLAST) is a wireless algorithm that detects the transmitted signal according to the ordering. Then, it uses the traditional detectors such as linear detectors [11]. VBLAST increases the channel capacity and its complexity comes from the ordering of matrix inversion. VBLAST detector firstly detects the stream with the highest power, then it continues the detection using the traditional methods. The main steps of VBLAST are: ordering to know the index of the signal, nulling the unwanted signal, and cancellation to subtract the detected signal from the received signal [12]. Here, we focus on ZF-VBLAST receiver which uses the linear ZF for nulling.

III. The Parallel ZF-VBLAST Structure

Generally, the ZF-VBLAST algorithm consists of five tasks that are explained through the flowchart in Fig. 1. Note that the flowchart shows input output relationship by f(.). Task 1 generates the transmitted vector ($\boldsymbol{x}$), the noise vector ($\boldsymbol{n}$) and the channel matrix ($\boldsymbol{H}$). Task 2 calculates the received signal ($\boldsymbol{y}$) and the matrix ($\boldsymbol{W}$). In task 3, ordering nulling, slicing and canceling are performed. Task 4 updates the channel matrix. In task 5, the average of error is calculated [11]. We parallelize the channel realization loop which includes the tasks from 1 to 4 using multi-threading. In multithreading we use 2, 4, 8 and 16 cores to check the enhancement of speedup comparing with serial time (one core). When the threads are created, the master calls the slaves, distributes the number of loop iterations equally among the slaves to guarantee the load balancing and waits them to finish, once the master communicates with the slaves and knows that they have finished the job, then the master concludes the final results. The synchronization is performed through wait and signal functions and the atomic update predicates.

A. Complexity Analysis

Using VBLAST technique gives better performance for MIMO system but increases the computational complexity. Complexity analysis allows the evaluation of the time required to recover the original signal which defines as the number of operations in terms of floating point operations per second (FLOPS). The complexity of ZF can be calculated using number of flops for real and complex of operations; real addition, complex addition, real multiplication and complex multiplication. It is shown that the complexity of the linear ZF can be calculated in terms of flops by [13]

$$\begin{aligned}\boldsymbol{C}_{\mathrm{ZF}}^{(\mathrm{flops})} &= 7\boldsymbol{N}_t^3 + 7\boldsymbol{N}_t^2\boldsymbol{N}_r - 2\boldsymbol{N}_t + 4\boldsymbol{N}_t\boldsymbol{N}_r \\ &+ \frac{1}{2}\boldsymbol{N}_r\log_2(M), \qquad (4)\end{aligned}$$

where M signifies the type of modulation. For example, $M = 2$ when the modulation type is a binary phase shift keying (BPSK). The complexity of ZF-VBLAST in terms of flops can be calculated by [13]

$$\begin{aligned}\boldsymbol{C}_{\mathrm{ZF-VBLAST}}{}^{(\mathrm{flops})} &= \boldsymbol{N}_t{}^4 + \frac{5}{3}\boldsymbol{N}_t{}^3 + \frac{8}{3}\boldsymbol{N}_t{}^3\boldsymbol{N}_r \\ &+ \frac{3}{4}\boldsymbol{N}_t{}^2 + \frac{7}{2}\boldsymbol{N}_t{}^2\boldsymbol{N}_r + \frac{55}{6}\boldsymbol{N}_t\boldsymbol{N}_r \\ &- \frac{17}{2}\boldsymbol{N}_t + \frac{1}{2}\boldsymbol{N}_t\log_2(M). \qquad (5)\end{aligned}$$

With reference to the flowchart in Fig. 1, analysis applied to parallelize the number of experiments for different channel realizations loop that include the iterative process with tasks from 1 to 4. Dependency analysis is formed to verify the ability of parallelization. Moreover, execution time analysis is applied to verify the computationally intensive tasks in the algorithm. The call graph in Fig. 2 is generated using gprof, which is performance analysis tool for Linux to perform time analysis for ZF-VBLAST algorithm and display the percentage of time spent in each task.

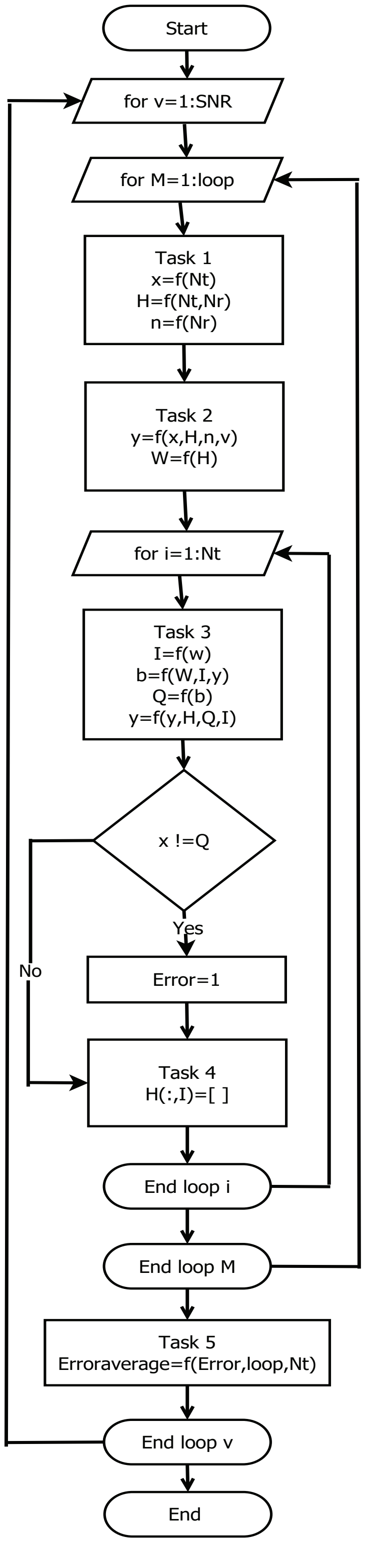


Fig. 1: Detailed Flow Chart for ZF-VBLAST Algorithm.

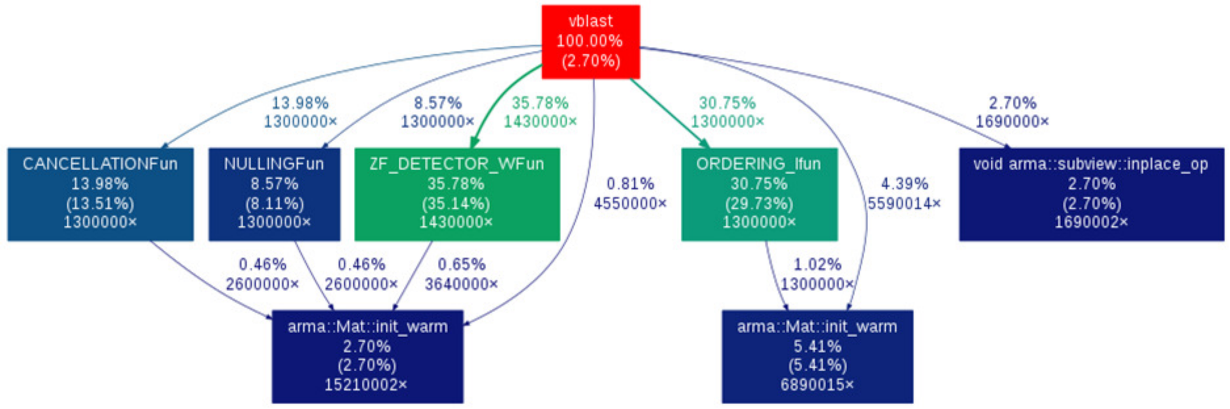
The call graph shows that the algorithm is fully dependent and the major computation time of the algorithm is spent in calculating the weight of the matrix $\boldsymbol{W}$, which is 35.78% as shown in Fig. 2. Therefore, some experiments are carried out to measure the serial time and the parallel time for the channel realization loop iterations for different MIMO sizes and different number of threads. Parallelization is affected by N_t, N_r and the number of channel realization.

IV. EXPERIMENTAL SETUP AND RESULTS

This section tests the ability to parallelize the ZF- VBLAST algorithm. In addition, we evaluate the achieved speed up for ZF-VBLAST algorithm using multithreaded implementation. The multithreaded implementation is performed on a server with up to 48 cores employed for the experiments. Our server is Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60G and 128GB of RAM. We implement ZF and ZF-VBLAST uplink multi users MIMO system in MATLAB. The Experimental results show the performance of MIMO system based on symbol error rate (SER) versus signal to noise ratio (SNR) using BPSK modulation and two different techniques of detection the linear ZF detector and the nonlinear ZF-VBLAST detector. We use different MIMO systems sizes NtxNr which are (10×10) and (10×100), for 1000 and 10000 channel realization loop. It is clearly shown in Fig. 3 that ZF-VBLAST is better than the linear ZF. ZF-VBLAST algorithm it is converted to C++ Armadillo version 9.7 linear algebra library for the C++ language. The C++ (g++ 5.4) for Ubuntu is used to compile the code with -04 compiler optimization level. We measure the parallel time by running the experiments with different number of threads 2, 4, 8 and 16.

As mentioned in Section III, some experiments are implemented to evaluate the serial time and the parallel time of parallelizing the channel realization loop iterations for two different MIMO sizes (10×10) and (10×100) and different number of threads. The time analysis for (10×10) MIMO system is performed when N_t = 10 and N_r = 10 of channel realization 1000 and 10000, respectively. Then same experiments were implemented but when N_t = 10 and N_r = 100 of channel realization 1000 and 10000. Figures 4 and 5 show the effect of increasing the number of cores (X-axis) on the scaling of the speed up (Y-axis). We run the experiments with serial and parallel implementation and calculate the achieved speed up for parallelizing the channel realization loop iterations. The parallelization for 1000 channels achieves maximum speed up of 3.5X, where X is a multiple time, for (10×10) MIMO systems, and for (10×100) MIMO systems achieves maximum speed up 7X for 16 cores as shown in Fig. 4.

Then, the same experiments were carried out, but with 10000 channels as shown in Fig. 5. These experiments shows that parallelization for (10×10) MIMO system achieves maximum speed up of 5.8X for 16 cores, and for (10×100) MIMO system achieves maximum speed up 11.4X. An increase in the speed up has been noticed with 10000 channels compared with the 1000 channels. To explain that Fig. 6 shows that the speed up was affected while changing the number of channels. We have used a master-slave model [14] for the parallel algorithm in which the master node assigns tasks to the slave threads. Therefore, synchronization is required to run all tasks among the slave threads. That is why some analysis performed to check the synchronization time for parallelizing



vblast 100.00% (2.70%)
13.98% 1300000×
8.57% 1300000×
35.78% 1430000×
30.75% 1300000×
2.70% 1690000×
CANCELLATIONFun 13.98% (13.51%) 1300000×
NULLINGFun 8.57% (8.11%) 1300000×
ZF_DETECTOR_WFun 35.78% (35.14%) 1430000×
0.81% 4550000×
ORDERING_Ifun 30.75% (29.73%) 1300000×
4.39% 5590014×
void arma::subview::inplace_op 2.70% (2.70%) 1690002×
0.46% 2600000×
0.46% 2600000×
0.65% 3640000×
1.02% 1300000×
arma::Mat::init_warm 2.70% (2.70%) 15210002×
arma::Mat::init_warm 5.41% (5.41%) 6890015×

Fig. 2: Call Graph for ZF-VBLAST Algorithm

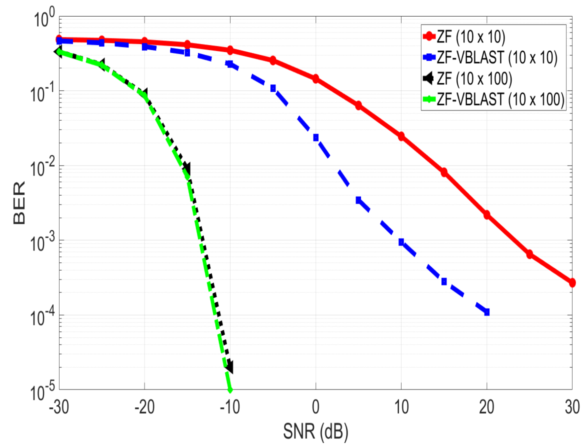


Fig. 3: The Performance of ZF and ZF-VBLAST.

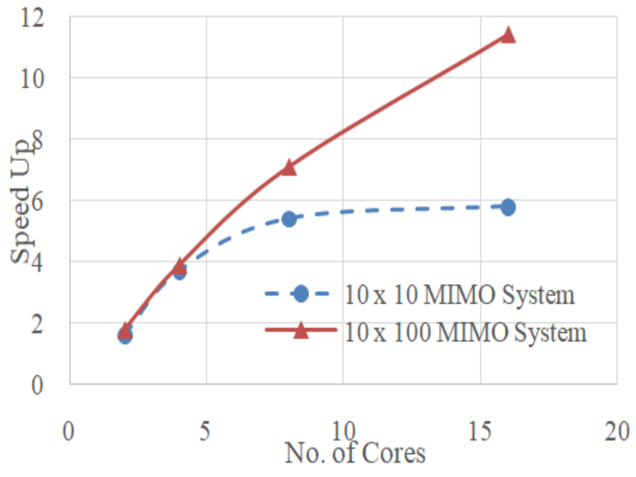


Fig. 5: Speed up versus Number of Channels for 10 × 10 and 10 × 100 MIMO Systems with 10000 Channels

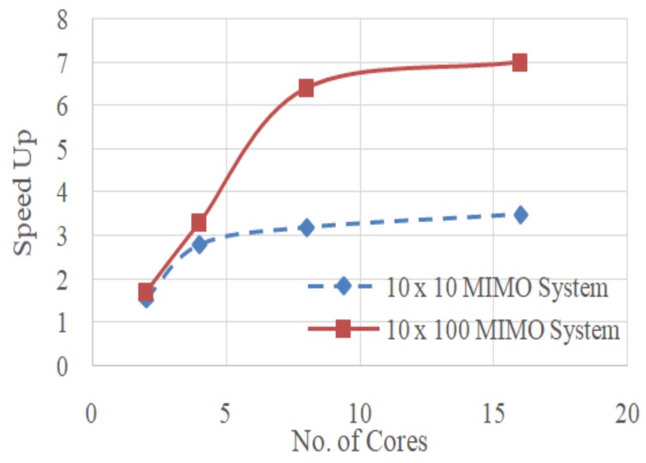


Fig. 4: Speed Up versus Number of Cores for 10 × 10 and 10 × 100 MIMO Systems with 1000 Channels

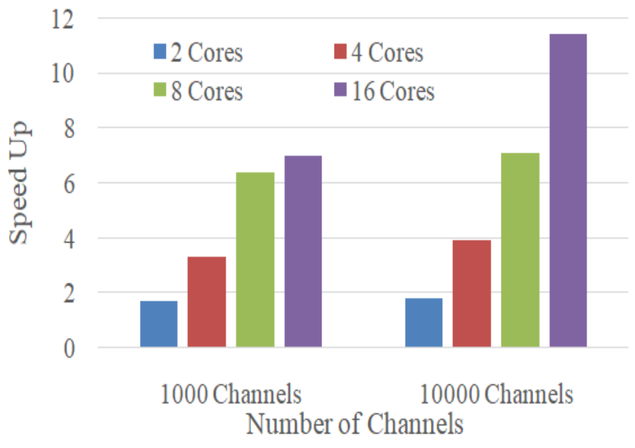


Fig. 6: The Effect of the Number of Channels over Speed up for 10 × 100 MIMO System

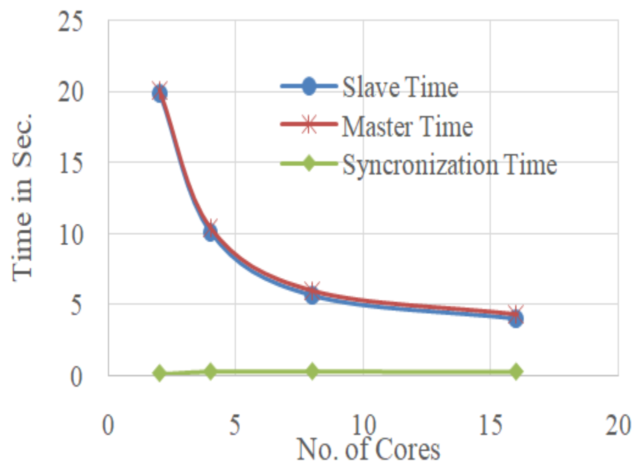


Fig. 7: Time versus Number of Cores for 10×100 MIMO System with 10000 Channels

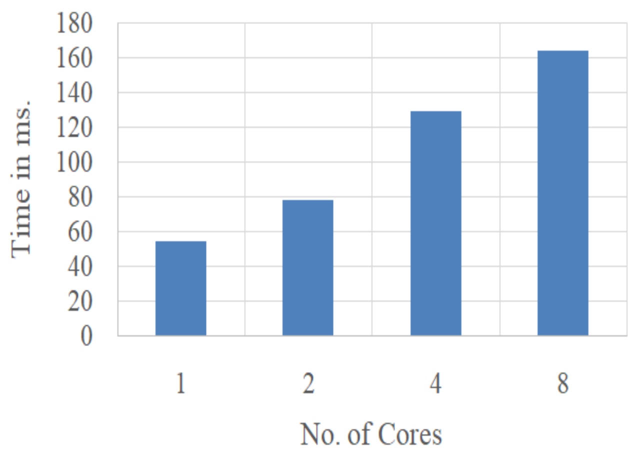


Fig. 8: Time versus Number of Cores for 10×100 MIMO System with 10000 Channels

the channel loop. This is done using (10×100) MIMO system with 10000 channels as shown in Fig. 7. Its is clear that the synchronization time is small and linear. As well, the master time to assign tasks and slave time to perform the assigns tasks by the master node decreases with increasing the number of cores.

All results of parallelizing the channel realization shown from Fig. 4 to Fig. 7 improves the efficiency of the ZF-VBLAST. As illustrated in Section III, we verified the ability to parallelize the VBLAST algorithm. Graph dependency using gprof indicates that the algorithm is fully dependent. We realized that the major computation time is spent in calculating the weight matrix W. Because of the limited time involved to calculate the weight matrix W, the synchronization time is greater than the serial time as shown in the experiment in Fig. 8.

V. CONCLUSIONS

We succeed in getting speed up of ZF-VBLAST algorithm for different sizes of MIMO system (10×10) and (10×100) using different numbers of threads (2, 4, 8 and 16). The obtained maximum speed up is 11.4X which can be achieved by increasing the system size and channel realization loop which leads improving the performance of the ZF-VBLAST algorithm. Despite of the high dependency in the VBLAST algorithm and the resultant high synchronization time compared with the serial time, we achieved a speed up from parallelizing the channel loop.

ACKNOWLEDGMENT

This research is supported in part by a University of Sharjah Targeted Project no. 1602040336-P and in part by the Distributed and Networked Systems Research Group Operating Grant number 150410, University of Sharjah.

REFERENCES

[1] C. Wang *et al.*, "Cellular architecture and key technologies for 5G wireless communication networks," *IEEE communications magazine*, no. 2, pp. 122–130, 2014.
[2] L. Lu *et al.*, "An overview of massive MIMO: Benefits and challenges," *IEEE journal of selected topics in signal processing*, no. 5, pp. 742–758, 2014.
[3] W. Zhongpeng, "Iterative detection and decoding with PIC algorithm for MIMO-OFDM systems," *Scientific Research Publishing, Int'l J. of Communications, Network and System Sciences*, 2009.
[4] A. I. Elnashar, "To parallelize or not to parallelize, speed up issue," *arXiv preprint arXiv:1103.5616*, 2011.
[5] V. Kumar *et al.*, *Introduction to parallel computing*, 1994.
[6] B. Barney *et al.*, "Introduction to parallel computing," *Lawrence Livermore National Laboratory*, no. 13, p. 10, 2010.
[7] H. Khaled *et al.*, "Parallel study of 3-D oil reservoir data visualization tool using hybrid distributed/shared-memory models," in *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*. IEEE, 2018, pp. 1016–1021.
[8] C. Husmann *et al.*, "Flexcore: massively parallel and flexible processing for large {MIMO} access points," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 197–211.
[9] G. Hegde *et al.*, "Parallel low-complexity M-PSK detector for large-scale MIMO systems," in *2016 IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM)*, 2016, pp. 1–5.
[10] C. Ramiro *et al.*, "MIMOPack: a high-performance computing library for MIMO communication systems," *Springer, The Journal of Supercomputing*, vol. 71, no. 2, pp. 751–760, 2015.
[11] K. Alnajjar *et al.*, "Low complexity V-BLAST for massive MIMO," in *IEEE 2014 Australian Communications Theory Workshop (AusCTW)*, 2014, pp. 22–26.
[12] J. Choi *et al.*, "Improved linear soft-input soft-output detection via soft feedback successive interference cancellation," *IEEE Transactions on Communications*, no. 3, pp. 986–996, 2010.
[13] A. M. Elshokry, "Complexity and performance evaluation of detection schemes for spatial multiplexing MIMO systems," *Islamic University Gaza, M. S. thesis*, 2010.
[14] S. Sahni and G. Vairaktarakis, "The master-slave paradigm in parallel computer and industrial settings," *Journal of Global Optimization*, vol. 9, no. 3-4, pp. 357–377, 1996.