Computational Complexity Analysis of PTS Technique under Graphics Processing Unit

Satyendra Singh Yadav, *Student Member, IEEE*[*], Prasanta Kumar Pradhan, *Student Member, IEEE* [†], Sarat Kumar Patra, *Senior Member, IEEE*[‡]
Department of Electronics and Communication Engineering
National Institute of Technology, Rourkela, India-769008
[*]yadav89satyendra@gmail.com, [†]dearprasanta@gmail.com, [‡]skpatra@nitrkl.ac.in

***Abstract*—Peak-to-Average Power Ratio (PAPR) reduction is a important signal processing aspect of Orthogonal Frequency Division Multiplexing (OFDM). Minimization of PAPR can be achieved efficiently using PTS Technique. The main drawback of PTS lies in its computational complexity due to large order of multiple IFFT and phase factor computation. Typically simulation to tune 4 sub-blocks with 4 phase factors in PTS is achievable using standard computing hardware. In this paper we present performance comparison of PTS technique using different number of sub-blocks (V=2, 4, 6 and 8) with different phase factors (W=2, 4 and 8) under Central Processing Unit (CPU) and Graphics Processing Unit (GPU) environments. The performance gain of GPU over CPU in terms of speedup is presented and the computational complexity involved is analyzed. The GPU is approx 3.7×, 7×, and 9× faster than CPU in case of 2, 4, and 8 phase factors respectively.**

Keywords: PTS, GPU, OFDM, PAPR, Phase factor and Streaming Multiprocessor.

I. Introduction

Orthogonal Frequency Division Multiplexing (OFDM), a multicarrier modulation technique, is a key technology for the future wireless communication systems. This technology converts frequency selective channel to several flat fading channels for elimination of fading effects [1]. Many International standards such as European Digital Video Broadcasting (DVB), Digital Audio Broadcasting (DAV), Wireless MAN (IEEE 802.16e), Wireless LAN (IEEE 802.11 a/g/n) and Long Term Evaluation (LTE) have been designed using OFDM technology [1]. Multi-carrier transmission system produce high peaks that causes inter-modulation among the sub-carriers and generates out-of-band radiation that is highly undesired. To operate the power amplifiers in linear region the ratio between peak power and average power of the transmitter should be close to unity. This paper analyzes the problem of PAPR reduction [2].

To overcome PAPR problem many techniques has been proposed. Some of these are selected mapping (SLM) [3], coding [4], active constellation extension (ACE) [5] , clipping and filtering [6], tone injection (TI) [7], tone reservation (TR) [8], partial transmit sequence (PTS) [2, 9, 10] and interleaving [12] . PTS technique divides the information data in to number of sub-blocks that has to be processed by separate IFFT blocks with constant number of sub-carriers. Each sub-stream of data is then multiplied by phase factor. As the number of sub-blocks increases, computational complexity of the PTS technique also increases.

General purpose Graphics Processing Unit (GPU) computing is an recent trend in wireless communication systems to increase computational performance of the system [13] . GPU is an array of low cost parallel processors that allow users to compute by employing different cores to execute a set of data in parallel.

This paper investigates comparative complexity analysis of PTS technique in terms of speedup for PAPR reduction by varying the number of sub-blocks with different possible phase factors under GPU and CPU environments. The research work is focused to provide the parallel support to the most computational complex part of PTS technique under GPU environment and compares the processing time with serial processing time under CPU environment, which are described in section IV.

Following this introduction, rest of paper is organized as follows. Section II discusses PAPR problem and PTS technique for PAPR reduction. Section III introduces the GPU architecture. Section IV presents the simulation framework and section V discusses on the results obtained. Finally section VI provides concluding remarks.

II. PAPR and PTS Technique

A. PAPR in Multi-carrier system

In OFDM system modulated serial data stream is converted in to parallel data stream consisting series of frames which are orthogonal to each other and modulated by a set of N number of sub-carrier $X = [X_0, X_1, ..., X_{N-1}]^T$. This is obtained by assuming $\triangle f = 1/(NT)$ where, T is the duration of the OFDM symbol and $\triangle f$ is the sub-carrier spacing. This process of modulation is achieved by use of IFFT and is represented as-

$$x(t) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X(k)\, e^{j2\pi t\Delta f k}, \qquad 0 \le t \le NT \quad (1)$$

Where, NT is a data block period.

This multi-carrier signal produces very high peaks causing nonlinearities in terms of inter-modulation among the sub-carriers, producing out-of-band radiation. The peak power of $x(t)$ can be written as

$$P_{peak} = \max_{0\leq n\leq NT} |x(t)|^2 \tag{2}$$

The average power of signal $x(t)$ is given by

$$P_{avg} = E\left\{|x(t)|^2\right\} \tag{3}$$

Where, $E\{.\}$ is the expected value of the signal $x(t)$.

PAPR of the OFDM signal is the ratio of its maximum instantaneous power to its average power [2]. Hence, the PAPR of signal $x(t)$ from (2) and (3) can be written as

$$PAPR = \frac{\max_{0\leq n\leq NT} |x(t)|^2}{E\left\{|x(t)|^2\right\}} = \frac{P_{peak}}{P_{avg}} \tag{4}$$

PAPR is usually presented using complementary cumulative distribution function (CCDF) of PAPR as its performance measure parameter. The CCDF of PAPR denotes the probability that PAPR of one data block exceed the certain given threshold [2]. The CCDF of the discrete-time PAPR is represented as-

$$\begin{aligned} CCDF(N, PAPR_0) &= \Pr\{PAPR > PAPR_0\} \\ &= 1-(1-e^{-PAPR_0})^N \end{aligned} \tag{5}$$

Where, N is the number of sub-carriers and $PAPR_0$ is the threshold value of PAPR.

B. PTS Technique

The block diagram of PTS for PAPR is presented in Fig. 1. Modulated data X is partitioned in to M disjoint sub-blocks, $X_m = [X_{m,0}, X_{m,1}.....X_{m,N-1}]$, $m = 1, 2.....M$. The sequences $[X_{m,0}, X_{m,1}, ...X_{m,N-1}]$ are called as PTS.

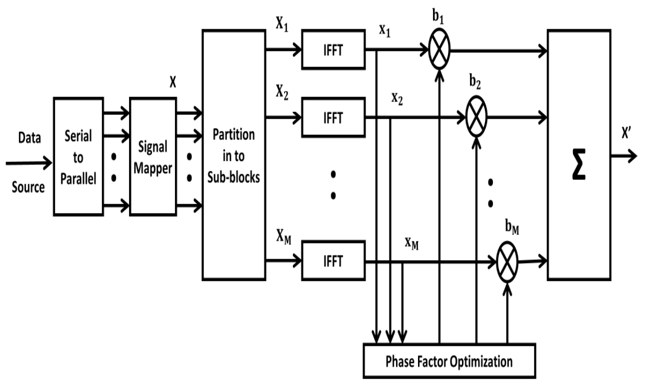

Fig. 1. Block Daigram of PTS Technique

Fig. 2 shows the adjacent sub-block partition method for PTS technique, where number of sub-carriers is 8 (i.e. N=8) and number of sub-block is 4 (M=4). Hence the data X can be represent as

$$X = \sum_{m=1}^{m=M} X_m \tag{6}$$

After sub-blocks partition, N point IFFT operation is performed on each disjoint sub-block separately therefore $x_m = IFFT(X_M)$. The IFFT is multiplied by the weighted phase factors $\mathbf{b} = [\mathbf{b_1}, \mathbf{b_2}, ..., \mathbf{b_M}]^\mathbf{T}$. The complex values of phase factors can be obtained by $b_m = e^{j\phi_m}$, where $m = 1, 2, ..., M$. This paper uses the phase factor (±1), $(\pm1, \pm j)$ and $(\pm1, \pm j, \pm\sqrt{2}-\sqrt{2}, \pm\sqrt{2}+\sqrt{2})$ for $W = (2, 4, 8)$ respectively where, W is the number of possible phase factors. The total number possible combinations of the phase factors can be get by W^{V-1}. Each sub-block is multiplied with these phase factor and combine result after multiplication can be represent as

$$X' = \sum_{m=1}^{m=M} b_m \cdot x_m \tag{7}$$

The values of the phase factors should be optimized to achieve the lowest peak power and high average power of X', that provides the PAPR of data X as given in (4).

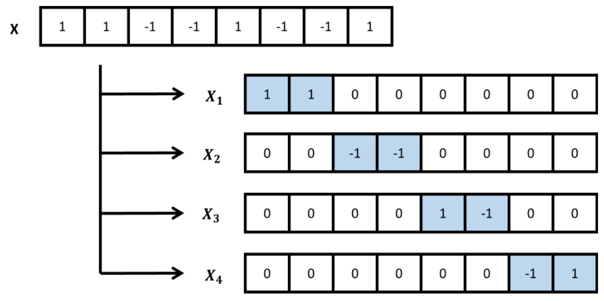

Fig. 2. Adjacent Sub-blocks Partition Scheme in PTS

III. GPU Architecture

The block diagram of a GPU is presented in Fig. 3. A GPU based High Performance Computing (HPC) system has multiple processors. It has an array of smaller processors with their shared cache and a shared memory. Currently, a single GPU system can have thousands of streaming multiprocessors (SMs). These systems have the capability of high processing throughput through parallelization. This architecture gives high throughput when the all streaming processors (SPs) are working in parallel and can handle large amount of data very efficiently. Highly parallel structure of GPU makes it more effective than CPU for parallel algorithms and general purpose computing [13–15].

IV. Simulation Framework

This simulation work presents comparative study of processing time of PTS technique for PAPR calculation under CPU and GPU environment. To analyse the performance of PTS technique for PAPR reduction using GPU the following simulation framework was employed-

The whole simulation is performed using MATLAB-2012 software installed in HPC (High Performance Computing) server using remote login. The simulating system hardware has a host PC which consists of $Intel$® $Xeon$® E5-2650 processor operating at 2.0 GHz frequency with Linux operating system. On the other side NVIDIA Tesla M-2090 GPU is

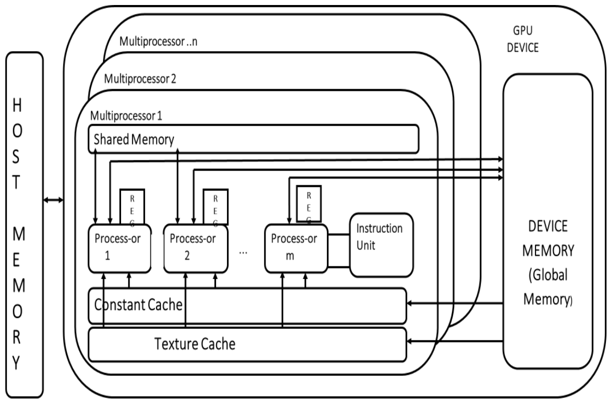

Fig. 3. NVIDIA GPU Architecture

used which have 512 streaming multiprocessor and operating at 1.3 GHz frequency. To compare and analyze performance of xeon processor (CPU) in reference to GPU environment, parallel computing toolbox of MATLAB was extensively used during simulation studies.

In this work to simulate PTS technique 16 QAM modulation is used with N=128 number of sub-carrier. Simulation is performed by varying the number of sub-blocks (i.e. V=2,4,6, 8) while keeping the number of sub-carrier constant for different number of possible phase factors (i.e. W=2,4,8). The performance of CPU and GPU systems are presented in terms of speedup, which is the ratio of CPU time to the GPU time.

V. Results and Discussion

Simulation of PTS technique for PAPR reduction has been performed under CPU and GPU environment considering above discussed parameters. Fig. 4 presents the CCDF plot of PAPR of PTS technique for CPU and GPU system for different number of sub-blocks (V=2,4,6,8). Form the Fig. 4, it can be observed that PAPR performance of both CPU and GPU are same (i.e GPU does not have any advantage regarding to the reduction in PAPR value of PTS technique). As the number of sub-blocks partition of PTS technique increases the PAPR decreases significantly at the cost of computational complexity of the PTS technique.

Hence by increasing the number of sub-blocks, the system becomes more computationally intensive, therefore multiplication of the phase factor with the each sub-block as denoted in (7) is quite time consuming process, hence this part of the system is simulated under GPU environment. The processing time to simulate (7) under CPU and GPU environment is represented in Fig. 5. The time to move the data to/from CPU to GPU and vice versa is not included in timing analysis.

It can be observed from Fig. 5, GPU consuming more time, when the number of sub-blocks is less therefore, if small amount of data has to be processed the performance of GPU system is inferior compared to the CPU because of its overhead. On the other hand as the number of sub-block is increases GPU processes all the sub-blocks in parallel, hence performance of the GPU system is several times better than CPU. The Fig. 5 represents speedup curve on secondary Y-axis (right side) for different sub-blocks. From the Fig. 5 it can be observed that GPU is approx 3.7 $\times$ faster than CPU for W=2 and V=8.

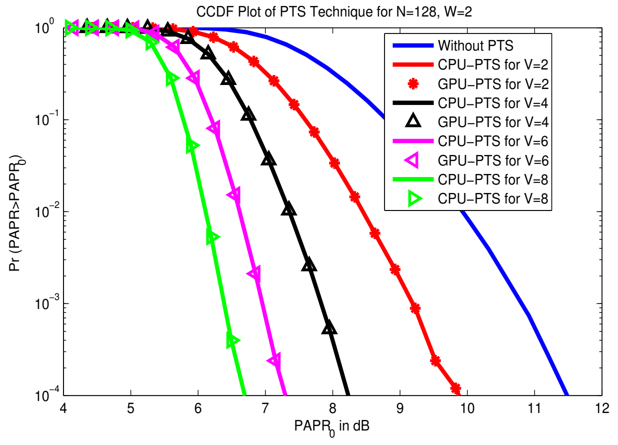

Fig. 4. CCDF of PAPR of an PTS technique with different sub-blocks (V=2,4,6,8) for 16-QAM modulation and W=2, N=128 under CPU and GPU environments.

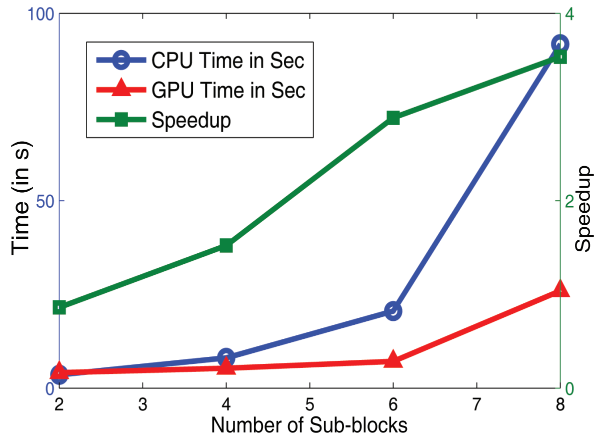

Fig. 5. Processing time under CPU and GPU environment and speedup for N=128, W=2

As the number of possible phase factors (W) increases the performance of PTS technique also improves. CCDF curve of PAPR for PTS technique with the same simulation parameters as discussed earlier except W=4 presented in Fig. 6. It shows that the PAPR of PTS technique reduced for both CPU and GPU in comparison to the previous case as shown in Fig. 4.

The PTS technique with W=4 has approx 1 dB lower PAPR value for each different number of V as compared to W=2, this improvement due to increasing the number of phase factors, cost in additional computational complexity in PTS technique.

Hence the computational complexity in terms of phase factor multiplication for W=4 is very high in comparison to W=2, that can be find by $B = W^{V-1}$ Where, B is the number of possible allowed phase factors. The processing time to simulate (7) under CPU and GPU environment for different

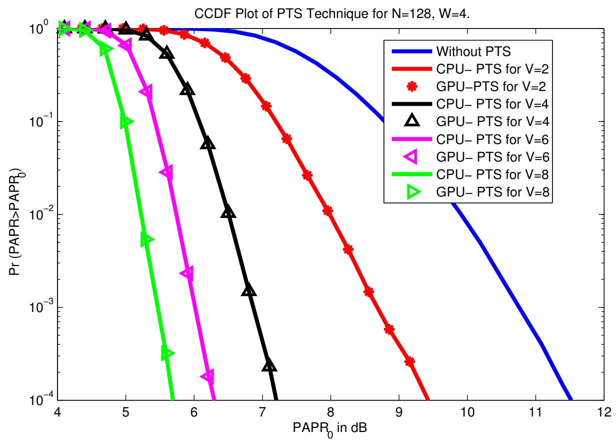

Fig. 6. CCDF of PAPR of an PTS technique with different sub-blocks (V=2,4,6,8) for 16-QAM modulation and W=4, N=128 under CPU and GPU environments.

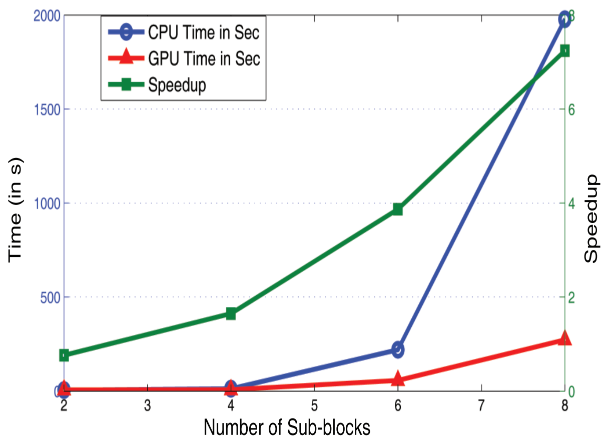

Fig. 7. Processing time under CPU and GPU environments and speedup for N=128, W=4

number of sub-blocks with number of possible phase factors W=4 is presented in Fig. 7. It can be observed that the computational time for W=4 and V=8 under GPU environment rises only 10 times in comparision to the previous case W=2. While it is approx 20 times higher under CPU environment. The speedup curve considering W=4 and having same simulation parameter as earlier is presented on secondary Y-axis in Fig. 7. In this case for V=8, the GPU is approximately 7× faster than the CPU.

The simulation of PTS technique for PAPR calculation with possible number of allowed phase factor W=8 under CPU and GPU environment is presented in Fig. 8. It can be observed that the PAPR of OFDM is reduced till great extend as the number of possible phase factors (W) increases. The PAPR with W=8 have approximately 1 dB and 2 dB improvement with respect to the previous cases for W=4 and W=2 respectively under both the environments. But this improvement has 128 times more computational complexity than W=4 in terms of choosing the optimal phase factor and multiplication of these with the sub-blocks, for example- if W=4 and V=8 then the total number of possible phase factor combinations is 16384 while for W=8 and V=8 it is 2097152. Hence for this part of the system parallel processing is required by executing the data on GPU.

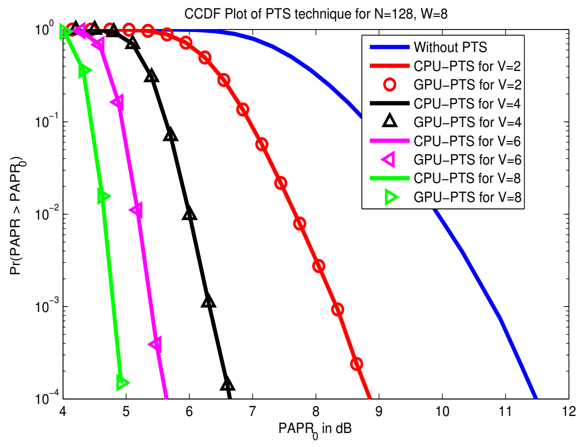

Fig. 8. CCDF of PAPR of an PTS technique with different sub-blocks (V=2,4,6,8) for 16-QAM modulation and W=8, N=128 under CPU and GPU environments.

The multiplication of phase factors with data of different sub-blocks are performed in parallel under GPU environment and time taken to simulate the data by CPU and GPU is presented in Fig. 9. It presents the processing time in seconds by CPU and GPU on first Y-axis (left side), number of sub-blocks on X-axis and speedup on the secondary Y-axis. From Fig. 9 It can be seen that GPU takes less time to execute more complex part of the system. The GPU has peak performance around 9× more faster than CPU for W=8 and V=8.

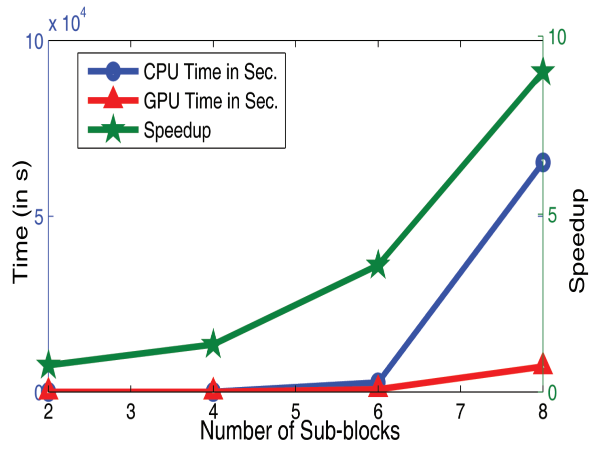

Fig. 9. Processing time under CPU and GPU environments and speedup for N=128, W=8

The complete timing analysis for 5 successive iterations under CPU and GPU with different sub-block for W=4 are presented by bar chart in Fig. 10 and complete timing analysis presented in Table I. The bar chart is not considered for W=8 because of the huge computational complexity related to it (i.e it is not possible to get the data for 5 iterations easily by CPU).

TABLE I
SPEEDUP COMPARISON BETWEEN CPU AND GPU FOR DIFFERENT VALUES OF V, WHERE W=4

	For V=2			For V=4			For V=6			For V=8		
Iteration No.	CPU Time	GPU Time	**Speedup**	CPU Time	GPU Time	**Speedup**	CPU Time	GPU Time	**Speedup**	CPU Time	GPU Time	**Speedup**
1	5.3	6.951	0.762	12.8	7.762	1.648	219.8	56.836	3.867	1977.2	273.040	7.241
2	4.6	5.989	0.768	15.4	9.253	1.664	313.3	60.237	5.201	2520.8	399.843	6.304
3	5	6.524	0.766	15.3	9.431	1.622	314.7	59.993	5.245	2528.9	399.718	6.326
4	5.6	7.124	0.786	15.5	10	1.55	310.4	58.068	5.345	2201.3	304.588	7.227
5	5.9	7.352	0.802	16.1	10.563	1.524	311.4	57.048	5.458	2208.4	304.504	7.252

Chart denotes the error bar with Confidence Interval (CI) for respective sub-blocks in both the environments. CI is the range of time in which the simulation time, for CPU and GPU for different respective number of sub-blocks is estimated to lie.

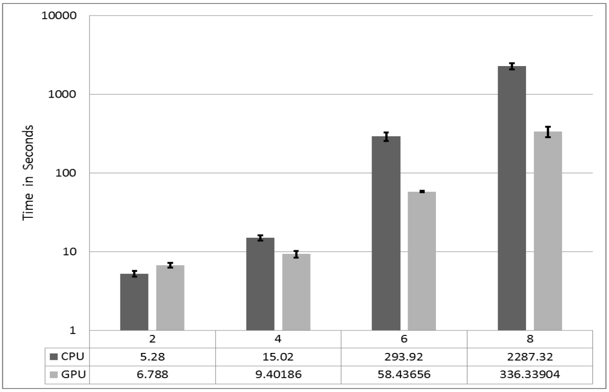

Fig. 10. Timing analysis of PTS technique under CPU and GPU environments

The vertical axis of the bar chart is plotted on logarithmic scale to accumulate all the points for different V because, the variation in time from V=2 to V=8 is from 5.28 seconds to 2287.32 seconds respectively, this difference is too large to accommodate on the linear scale. From the Fig. 10, it can be observed that for V=2 the CPU performs better than the GPU with the average time 5.28 sec and 6.788 sec respectively. But as the number of sub-blocks increases the complexity of PTS technique rises therefore performance of the GPU system also increases, for V=4 the average processing time for CPU is 15.2 sec while in case of GPU it is 9.402 sec. For V=8 GPU has the standard deviation 59.325 sec and confidence interval 52 sec, while in case of CPU, standard deviation is 235.942 sec and confidence interval is 206.809 sec. The average peak performance of the GPU for V=8 is 336.33904, which is approx 6.8704× better than the CPU.

VI. CONCLUSION

In this paper the simulation of PTS technique for PAPR reduction has been carried out under the GPU environment by providing parallel processing of data efficiently. The complexity of different number of phase factors by varying the number of sub-blocks, while keeping number of sub-carriers constant, has been analyzed under GPU environment. The simulation model uses massively parallel architecture of GPU using simple SMs leading to enhanced performance in terms of computational time. The use of GPU based high performance computing provides path for fast processing of the data and enhance performance of the system by providing parallel support by large number of small SPs. The massive computational power of the GPU in comparison to the CPU for highly intensive part of the PTS technique for PAPR reduction has been presented and analyzed.

REFERENCES

[1] H. Schulze and C. Lüders, *Theory and applications of OFDM and CDMA: wideband wireless communications*. John Wiley & Sons, 2005.

[2] S. H. Han and J. H. Lee, “An overview of peak-to-average power ratio reduction techniques for multicarrier transmission,” *Wireless Communications, IEEE*, vol. 12, no. 2, pp. 56–65, 2005.

[3] R. W. Bäuml, R. F. Fischer, and J. B. Huber, “Reducing the peak-to-average power ratio of multicarrier modulation by selected mapping,” *Electronics Letters*, vol. 32, no. 22, pp. 2056–2057, 1996.

[4] A. E. Jones, T. A. Wilkinson, and S. Barton, “Block coding scheme for reduction of peak to mean envelope power ratio of multicarrier transmission schemes,” *Electronics letters*, vol. 30, no. 25, pp. 2098–2099, 1994.

[5] B. S. Krongold and D. L. Jones, “PAR reduction in OFDM via active constellation extension,” *Broadcasting, IEEE Transactions on*, vol. 49, no. 3, pp. 258–268, 2003.

[6] X. Li and L. J. Cimini, “Effects of clipping and filtering on the performance of OFDM,” in *Vehicular Technology Conference, 1997, IEEE 47th*, vol. 3. IEEE, 1997, pp. 1634–1638.

[7] S. H. Han, J. M. Cioffi, and J. H. Lee, “Tone injection with hexagonal constellation for peak-to-average power ratio reduction in OFDM,” *Communications Letters, IEEE*, vol. 10, no. 9, pp. 646–648, 2006.

[8] D.-W. Lim, H.-S. Noh, H.-B. Jeon, J.-S. No, and D.-J. Shin, “Multi-stage TR scheme for PAPR reduction in OFDM signals,” *Broadcasting, IEEE Transactions on*, vol. 55, no. 2, pp. 300–304, 2009.

[9] S. H. Muller and J. B. Huber, “A novel peak power reduction scheme for OFDM,” in *Personal, Indoor and Mobile Radio Communications, 1997. Waves of the Year 2000. PIMRC'97., The 8th IEEE International Symposium on*, vol. 3. IEEE, 1997, pp. 1090–1094.

[10] P. Varahram, W. F. Al-Azzo, and B. M. Ali, “A low complexity partial transmit sequence scheme by use of dummy signals for PAPR reduction in OFDM systems,” *Consumer Electronics, IEEE Transactions on*, vol. 56, no. 4, pp. 2416–2420, 2010.

[11] A. Jayalath and C. Tellambura, “Reducing the peak-to-average power ratio of orthogonal frequency division multiplexing signal through bit or symbol interleaving,” *Electronics Letters*, vol. 36, no. 13, pp. 1161–1163, 2000.

[12] A. R. Brodtkorb, T. R. Hagen, and M. L. Sætra, “Graphics processing unit (GPU) programming strategies and trends in GPU computing,” *Journal of Parallel and Distributed Computing*, vol. 73, no. 1, pp. 4–13, 2013.

[13] T. Nylanden, J. Janhunen, O. Silvén, and M. Juntti, “A GPU implementation for two MIMO-OFDM detectors,” in *Embedded Computer Systems (SAMOS), 2010 International Conference on*. IEEE, 2010, pp. 293–300.

[14] S. Bhattacharjee, S. S. Yadav, and S. K. Patra, “LTE physical layer implementation using GPU based high performance computing,”IEEE International Conference on Advanced Communication, Control and Computing Technologies ICACCCT, Syed Ammal Engineering College”,” May 8-10, 2014, Ramanathapuram, Tamilnadu, India (In press).