# IMPLEMENTATION OF AN SDR PLATFORM USING GPU AND ITS APPLICATION TO 2X2 MIMO WIMAX SYSTEM

Jaehyuk Ju(HY-SDR Research Center, Hanyang Univ., Seoul, Korea: stop1017@dsplab.hanyang.ac.kr); Chiyoung Ahn(HY-SDR Research Center, Hanyang Univ., Seoul, Korea; ahncy@dsplab.hanyang.ac.kr); June Kim (HY-SDR Research Center, Hanyang Univ., Seoul, Korea; nzneer@dsplab.hanyang.ac.kr); Seungheon Hyeon (HY-SDR Research Center, Hanyang Univ., Seoul, Korea; hsheon@dsplab.hanyang.ac.kr); and Seungwon Choi*(corresponding author, HY-SDR Research Center, Hanyang Univ., Seoul, Korea; choi@ieee.org)

## ABSTRACT

Conventional communication systems have been implemented using Digital Signal Processing (DSP) and/or Field Programmable Gate Array (FPGA) especially for the Software defined radio (SDR) functionality. We propose a scheme of using Graphics Processing Unit (GPU) instead of those two conventional devices for implementing the SDR-based communication system. GPU, a high-speed parallel processor and numerous powerful arithmetic logic units, is adopted for the signal processing of physical layer required for the parallel processing of SDR system. Noting that Compute Unified Device Architecture (CUDA) based on C language provides Software Development Kit (SDK) for the modem application of GPU, we utilize the CUDA SDK to perform the modem function which operates on real-time basis. This paper presents an implementation of 2x2 Multi-Input Multi-Output (MIMO) WiMAX system using GPU as its modem. Mounting an RF module on top of our GPU modem, we demonstrate a real-time transmission of video data. The system performance of our GPU-based system is shown in terms of operation time.

## 1. INTRODUCTION

Software Defined Radio (SDR) system can support various communication protocols through the software download without changing its hardware. Conventional SDR system has been implemented using Digital Signal Processing (DSP)s and/or Field Programmable Gate Array (FPGA)s for its modem solution. The major drawbacks of using the DSPs and/or FPGAs are as follows. First, although DSP provides a good flexibility of coding given functions and development environment, the arithmetic operation capability of DSP is not sufficient for supporting all the requirements of modern communications in real time. On the other hand, FPGA suffers from relatively high price and extremely complicated procedures for development and debugging although its computational capability is a lot superior to DSP's. In order to overcome above-mentioned problems of using DSP and/or FPGA, researches for using Graphics Processing Unit (GPU)s, which are particularly robust for parallel processing, have been very actively performed in the modem implementation of SDR systems [1].

The application of GPU has been focused mainly on very high-speed floating-point parallel arithmetic operations. Having been provided Compute Unified Device Architecture (CUDA) which is c-based programming environment, one can develop high-speed applications using GPU with far more flexibility than using FPGA.

Parallelization of each of signal processing algorithms is very important for maximizing the performance of high-speed parallel arithmetic operations. The parallelization is a procedure of, first, distributing the operations that involve mutually independent data, and then, allocating each of the operation using the independent data to a corresponding thread. In this paper, we present an implementation example of a GPU-based WiMAX Multi-Input Multi-Output (MIMO) system through the parallelization of the signal processing algorithms. We also provide an analysis of required operation time of the implemented system.

This paper consists of the following structure. Section 2 explains GPU and CUDA itself. Section 3 shows the system architecture and GPU implementation of parallelized modem. Section 4 exhibits the system performance and evaluation. Finally, Section 5 concludes the paper.

## 2. OVERVIEW OF GPU/CUDA

As GPU consists of a Single Instruction Multiple Data (SIMD) architecture, it is inherently a good processor for parallel processing [2]. As shown in Figure 2.1, the GPU with SIMD processor architecture processes a single instruction using various Arithmetic Logic Unit (ALU)s with each of many data being processed at each of the corresponding ALU.

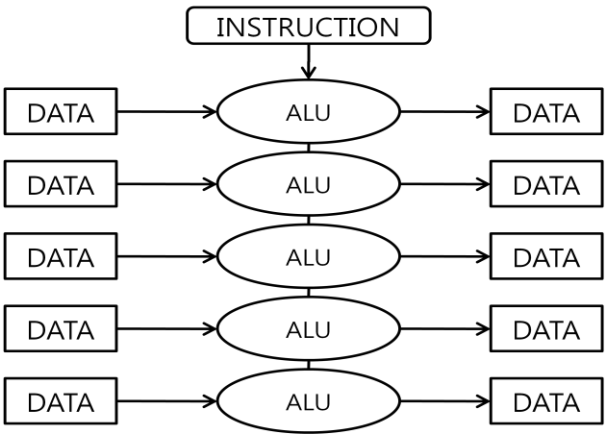


Figure 2.1 SIMD processor architecture

As shown in Figure 2.1, GPU includes a lot more ALUs than CPU or DSP does for 3D graphic operations. CUDA is a c-based programming environment for efficiently managing the many number of ALUs in GPU. Figure 2.2 illustrates the structure of CUDA [3].

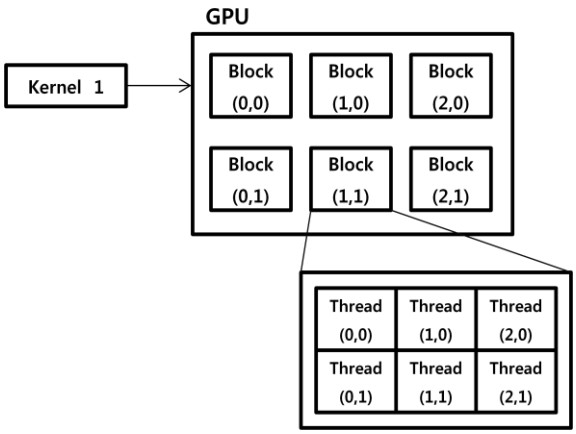
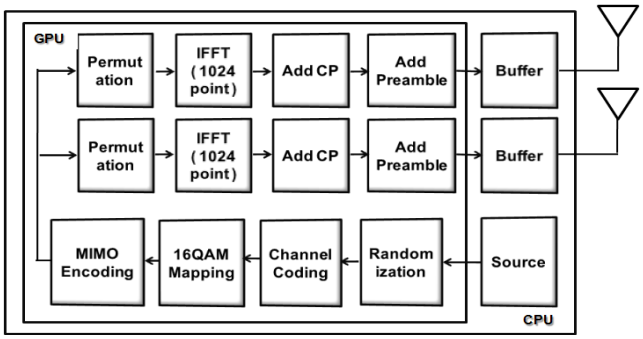


Figure 2.2 CUDA structure for GPU application

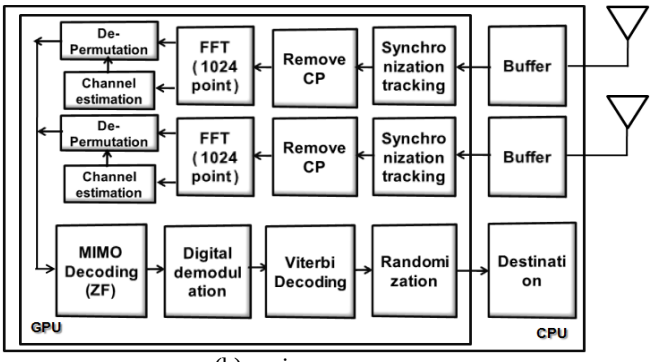
Kernel shown in Figure 2.2 is a function that enables the GPU to implement various functions generated in the host. As shown in Figure 2.2, a kernel consists of blocks each of which is composed of a number of threads. Note that all the threads in a block perform a single instruction. In order to maximize parallel processing capability of GPU, we have to allocate as many as possible threads to each instruction involving independent set of data. For that, GPU provides an IDentifier (ID) for distinguishing each thread. Programmer should control the input/output signal of each thread using the ID, which enables the parallel processing of independent set of data [4], [5].

## 3. IMPLEMENTATION OF WIMAX SYSTEM USING GPU

Figure 3.1 illustrates the entire system block-diagram of the 2x2 spatial multiplexing(SM) MIMO WiMAX system implemented in our lab.



(a)transmitter



(b)receiver

Figure 3.1 Block diagram of 2x2 SM MIMO WiMAX system

The core part of GPU implementation is the parallelization of signal processing algorithms appropriately for the SIMD architecture. Keeping that in our mind, we have designed the modem in such a way that as many threads as possible can be activated by dividing the data to be processed at each block into as small pieces as possible. For example, when the sum of 10 data is to be obtained, we would need 9 clocks if we perform serial additions. That would be reduced to 4 if 5 pairs are added in parallel. The WiMAX system implemented in this paper fully exploited the SIMD architecture as described in the following sub-sections.

### 3.1. Encoder Implementation

Encoder consists of Channel coding, Permutation, and IFFT component. IFFT has been implemented using the function given in CUDA [3].

In our paper, we take permutation component as an example of parallel technique. Permutation is a procedure of re-ordering the clusters consisting of 14 neighboring sub-carriers using the predetermined permutation table. For implementing that, we need threads as many as the number of clusters and destination memory for re-ordering the clusters as given in the permutation table as shown in Figure

3.2. Note that each thread provides the address of destination memory for storing the cluster that has been permutated using the ID of the thread. Permutation is performed by copying the cluster, which is matched to the ID of the thread, to the destination address. Therefore, the operation time for performing the permutation implemented in this paper is determined solely by the operation time for memory copying of the 14 data regardless of the number of clusters.

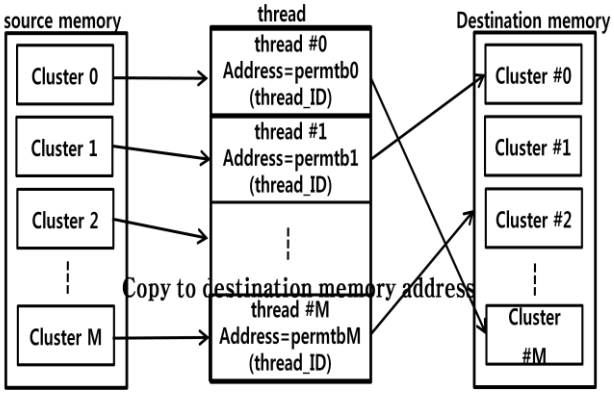



Figure 3.2 Block diagram for permutation

### 3.2. Decoder Implementation

Decoder consists of FFT, Channel Estimation, De-permutation, Compensation, and Viterbi Decoder component. In this paper, for simplicity, we explain the parallelization technique for the channel estimation only. Figure 3.3 illustrates cluster structure of WiMAX system in which 4 pilot bits at each cluster are used for channel estimation associated with the surrounding data subcarriers.

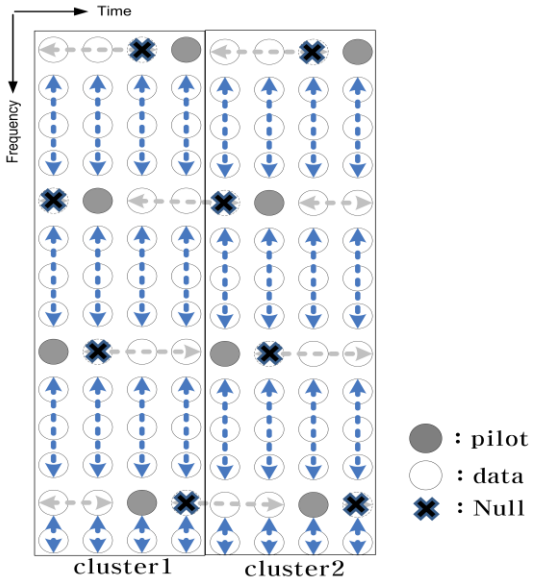



Figure 3.3 Structure of WiMAX cluster

In this paper, we have adopted 2D linear interpolation consisting of 2 steps for the channel estimation, which means a linear interpolation along the time-axis and frequency-axis has been processed at each of the corresponding 2 kernels. At the first step, using the pilots allocated at 1st, 5th, 9th, and 13th subcarriers, a linear channel estimation is performed along the time axis. We need 2 pilots for interpolation, which means we used 2 clusters for the channel estimation along the time axis. Each line including the pilots has been assigned to each thread, which means our parallelization involves 4-line channel estimation along the time axis as shown in Figure 3.3. At the second step, using the channel estimation obtained along the time axis, we perform the channel estimation along the frequency axis for the linear interpolation of data subcarriers. In order to parallelizing the channel estimation along the frequency axis, we have assigned a single cluster at a single block. Then, we divide the cluster into 3 regions consisting of 1-5, 5-9, and 9-14 as shown in Figure 3.3. At each region, we perform the channel estimation at each corresponding thread. As there are 4 columns of subcarriers at each cluster, there are 12 threads performing channel estimation at each cluster. It can be observed from the above parallelization architecture that the entire computation time required for the channel estimation of each frame of WiMAX is equivalent to the time required for processing just 2 kernels only. It particularly means that the entire computation time for the channel estimation of an entire frame is equivalent to the time for performing the interpolation in between the pilot signals. More specifically, as shown in Figure 3.3, since there are 4 symbols and 3 symbols in between the pilot symbols along the time axis and frequency axis, respectively, the entire time for the channel estimation required in our implementation is equivalent to the time needed for estimating for just 7 symbols only, regardless of the number of clusters!

## 4. PERFORMANCE EVALUATION

Figure 4.1 illustrates the GPU-based WiMAX MIMO system implemented through the parallelization of signal processing algorithms proposed in this paper.

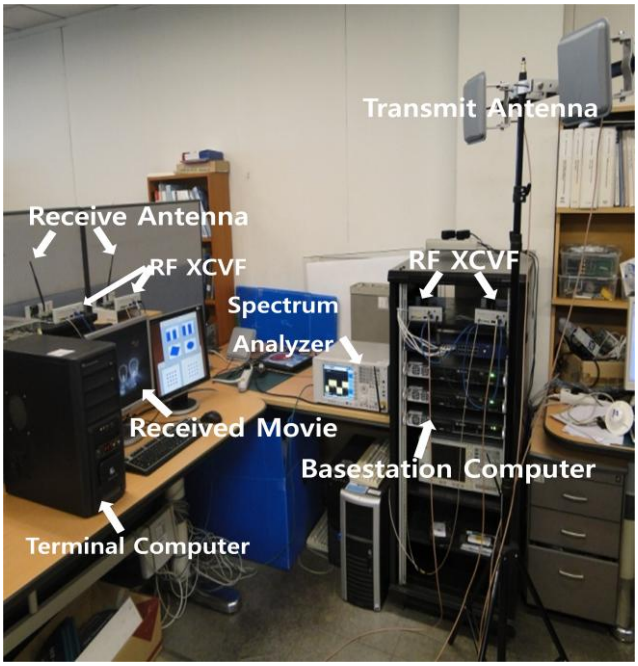


Figure 4.1. 2x2 SM MIMO WiMAX system

The test procedure for evaluating the performance of the implemented system is as follows. At the GPU which realizes the modem of the base station shown at the right-hand side of Figure 4.1, the parallelized signal processing algorithms are executed for encoding the video stream data in accordance with the WiMAX format. The encoded data are transmitted through the transmit antenna via the radio frequency (RF) transceiver (XCVR). The transmitted RF signal is captured at the receive antenna shown at the left-hand side of Figure 4.1. The received RF signal is processed at the decoder GPU inside the terminal computer shown at the left bottom of Figure 4.1 after frequency down-conversion and analog to digital conversion at the receive RF XCVR. The retrieved video data are shown in the monitor in real time as shown in Figure 4.1.

In order to measure the processing time consumed at the physical layer components, we have used the profiler provided by the GPU manufacturing company. Figure 4.2 illustrates the operation time consumed by the physical layer components inside the GPU in comparison to the case of using DSP. We have observed that the computation time for the case of GPU is 3-14 times less than that of using DSP. The main reason for each component to exhibit different performance in comparison to the case of DSP is due to the difference in parallelization performance at each component.

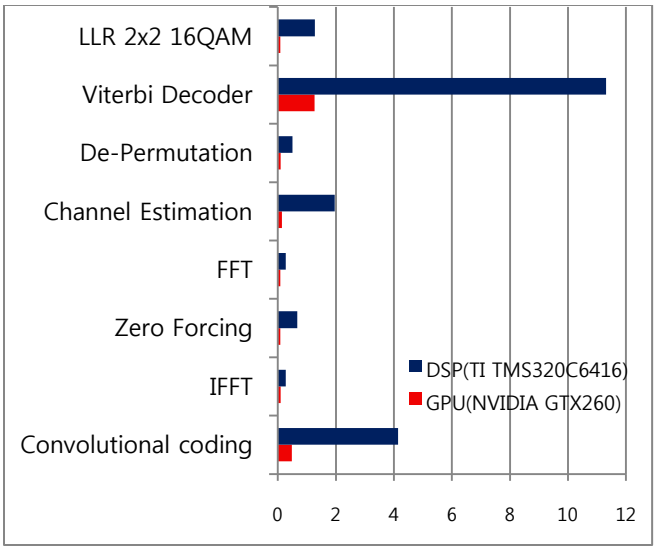


Figure 4.2 Comparison of computation time: GPU vs DSP

As shown in Figure 4.2 most operations can be performed within 0.1 ms using GPU. However, some operations such as convolutional coding and viterbi decoding consumes relative a large computation time due to the difficulty in parallelization.

| | GPU Processing time | DSP Processing time |
|---|---|---|
| Downlink | 754.976 $\mu$s | 6019.89 $\mu$s |
| Uplink | 2499.388 $\mu$s | 20148.2 $\mu$s |
| Total time | 3.254 ms | 26.168 ms |

Table 4.1 Processing time for 1 frame: GPU vs DSP

Table 4.1 shows that the computation time required for processing 1-frame data. Using GPU, downlink takes about 754.976 $\mu$s while uplink takes about 2499.388 $\mu$s, which means total computation time, 3.254 ms, is within the frame time 5 ms. Note that a single DSP cannot provide a successful real-time processing of the WiMAX data according to table 4.1.

## 5. CONCLUSION

This paper presents a parallelization technique using GPU as a modem solution of 2x2 WiMAX MIMO system. In order to maximize the parallelization capability of GPU, we first classify all the operations according to the independency of data used in each operation. Then, each of the operation is allocated to a thread. The modem of the WiMAX system has been implemented using a GPU based on the parallelization technique. Operation speed of the implemented system has been measured to find that the total computation time for processing a single frame of WiMAX system of which the duration is 5ms, is about 3.254 ms which is short enough for the real time processing.

Consequently, this paper proposes a proper parallelization technique for using GPU as a modem solution of WiMAX system. The technique proposed in this paper can be applied to SDR system once the GPU is properly ported with corresponding modem code.

## ACKNOWLEDGEMENTS

This work was supported by Seoul R&BD Program (PA090743).

## REFERENCES

[1] June Kim, Seungheon Hyeon, "Implementation of an SDR System Using", Communications Magazine of the IEEE, vol. 48, March 2010.
[2] JD Owens, M Houston, D Luebke, S Green, JE Stone, JC Phillips, "GPU Computing", Proceedings of the IEEE, vol. 96, May 2008.
[3] NVIDIA CUDA Compute Unified Device Architecture Programming Guide, 2009
[4] John D. Owens, "GPU Computing", Proceedings of IEEE, Vol 96, Issue 5, pp. 879-899, May 2008.
[5] E Lindholm, J Nickolls, S Oberman, J Montrym, "NVIDIA Tesla: A Unified Graphics and Computing Architecture," IEEE Micro, vol. 28, No.2, March-April 2008.
[6] Loutfi Nuaymi, John Wiley & Sons, "WIMAX, Technology For Broaband Wireless Access", 2007