**INDUSTRIAL TRAINING REPORT**

**TRAINING ORGANIZATION : LONDON STOCK EXCHANGE**

**GROUP TECHNOLOGY**

**PERIOD OF TRAINING : FROM 05/04/2021 TO 04/10/2021**

**FIELD OF SPECIALIZATION : COMPUTER ENGINEERING**

**M. P. U. PREMATHILAKA**

**E/15/280**

**ACKNOWLEDGMENTS**

The success I achieved during my training was not a solo effort by any means. Hence, it is with great pleasure that I acknowledge and extend my gratitude to those who contributed to the success of my internship.

I would like to extend my heartfelt gratitude to Mr. Janaka Perera, Associate Architect at the Hardware Accelerated Systems division, LSEG. Technology, my line manager throughout the internship. He was the driving force behind my projects, constantly guiding, advising, and motivating me to do my best.

Next, I would like to thank Mr. Priyanthan Parameswaran, Senior Engineer attached to Hardware Accelerated Systems division, LSEG Technology, for being my mentor and for the immense support, guidance, and invaluable knowledge imparted to me.

Further, I would like to thank my team members (GPU team), Mr. Sahan Henadeera, Associate Tech Lead, Mr. Athavaloshan, and Mr. Beniel Thileepan, Senior Engineers at Accelerated Systems, for sharing their experience and knowledge with me.

I am grateful to Hardware Accelerated Systems division lead by Mr. Thayaparan Sripavan, Head of Hardware Accelerated Systems, and the rest of the team members for giving me this opportunity to work in a great place.

I would like to thank Ms. Inthira Satheeskumar, Senior Executive, VPA at LSEG, for helping me with the recruitment process. Also, I would like to thank Ms. Sathmi Keenawinna, Senior Executive - Learning & Talent, and Mr. Mohommed Nisthar, Intern – HR, for organizing valuable sessions, exposing me to the company culture, and helping me to build up soft skills.

I would like to express my gratitude to our training coordinator, Dr. Janaka Alawathugoda, Senior Lecturer, Prof. Roshan Ragel, Department of Computer Engineering, University of Peradeniya, for preparing and giving advice for working on internship training.

And also, I would like to acknowledge Mr. W. R. M. U. Wickramasinghe and all the staff members in ITCGU at the University of Peradeniya and NAITA for all the commitments they have made for our internship.

Last but not least, I would like to thank all who made this internship a success by guiding me and supporting me.

**CONTENTS**

**ACKNOWLEDGMENTS** i **CONTENTS**  ii

**LIST OF FIGURES** iii

**LIST OF TABLES** iv

**LIST OF ABBREVIATIONS**  v

**Chapter 1 INTRODUCTION 1**

1.1 Training Session 1

1.2 Introduction to the Training Organization 1

1.3 Summary of Training Exposure 3

**Chapter 2 OVERVIEW, BACKGROUND AND RELATED WORK 5**

2.1 Introduction 5

2.2 Overview of Deeplight 5

2.3 Key Contributions 8

2.3 Background of Screen Camera Communication 9

2.4 Related work 11

**Chapter 3 DRAWBACKS OF PREVIOUS WORK AND DESIGN GOALS 13**

3.1 Introduction 13

3.2 Drawbacks of Previous Work 13

3.3 Design Goals 18

**Chapter 4 DESIGN AND IMPLEMENTATION 19**

4.1 Introduction 19

4.2 System Design 20

4.3 Experimental Setup 28

**Chapter 5 EVALUATION AND DISCUSSION 30**

5.1 Introduction 30

5.2 Evaluation 30

5.3 Discussion 31

**CONCLUSION 32**

**LIST OF FIGURES**

**Figure 1.1.** SMU Logo 1

**Figure 1.2.** Singapore Management University 2

**Figure 1.3.** SMU organization structure (Hierarchy of reporting officers) 3

**Figure 2.1.** Conceptual illustration of Screen-Camera Communication (SCC) 9

**Figure 2.2.** Illustrating Manchester coding for flicker suppression 11

**Figure 3.1.** Screen detection performance of HiLight using computer vision feature 11

**Figure 3.2.** Hilight: BER vs. Screen offset 15

**Figure 3.3.** Manchester coding with invariant (static) and variant (dynamic) data 15

**Figure 3.4.** Illustrating the Problem of Transition Frames 16

**Figure 4.1.** Overview of DeepLight decoding pipeline 19

**Figure 4.2.** ScreenNNet Pipeline 20

**Figure 4.3.** TransitioNNet Pipeline 21

**Figure 4.4.** IoU performance of ScreenNNet on different venues against different fps. 23

**Figure 4.5.** Screen detector accuracy when users hold the camera by hand. 24

**Figure 4.6.** Screen detection performance at different viewing angles. 24

**Figure 4.7.** Inception network for single shot detection 26

**Figure 4.8.** LightNet 27

**Figure 4.9.** Accuracy comparison between Basic and LightNet DNN pipeline. 28

**Figure 4.10.** Screen-Camera setup 29

**Figure 4.11.** Nvidia DGX server (obtained from web) 29

**Figure 5.1.** MOS of SCC systems against different screen framerates 30

**LIST OF TABLES**

**Table 3.1**. Summary of the previous work 13

**Table 4.1.**  SPAGE values for ScreenNNet on different venues 23

**ABBREVIATIONS**

BER Bit error rate

CNN Convolution neural network

DNN Deep neural network

FC Camera frame

FD Display frame

IoU Intersection over union

LED Light emitting diode

LSTM Long-term short-term memory

LARC Living Analytics Research Center

ML Machine learning

MOS Mean opinion score

RC Camera frame rate

RD Display frame tare

RGB Red green blue

SCC Screen to camera communication

SIS School of Information Systems

SMU Singapore Management University

VLC Visible light communication

Chapter 1

INTRODUCTION

**1. 1. TRAINING SESSION**

The training establishment that I underwent with respect to the TR400 Industrial Training module was London Stock Exchange Group Technology (LSEG Technology), also called Millennium IT Software (Private) Limited in Sri Lanka, situated at 01 Millennium Drive, Malabe. My training period commenced on 05th April 2021 and terminated on 04th October 2021, extending for 26 weeks.

**1.2. INTRODUCTION TO THE TRAINING ORGANIZATION**

**1.2.1 London Stock Exchange Group**

London Stock Exchange Group (LSEG) is a leading global financial markets infrastructure and data provider headquartered in the City of London, England, and operates in 70 countries across. London Stock Exchange was founded in Sweeting's Alley in London in 1801. In 2007, it merged with Milan Stock Exchange Borsa Italiana, creating the London Stock Exchange Group, and currently, it employs around 25,000 people globally, more than half located in the Asia Pacific. It is a public company leading the global financial markets infrastructure and data provider, trusted to deliver excellence by customers, partners, and markets worldwide.

The company's purpose is to drive financial stability, empower economies, and enable customers to create sustainable growth. Figure 1.1 below, illustrates the logo of the London Stock Exchange Group.

Figure 1.1: LSEG logo

**1.2.1 MillenniumIT Software (Private) Limited**

In the early 90’s Sri Lanka had a rudimentary stock trading system, and due to the lack of technology, the authorities could not expand and attract foreign investors. In 1996, MillenniumIT (MIT) was founded by Tony Weerasinghe, and they provided a software-based solution to the Sri Lankan trading System at Colombo Stock Exchange (CSE).

In 2009 London Stock Exchange Group acquired MillenniumIT with a contract for a new enhanced, fast trading platform for equities. This acquisition was a significant milestone in MIT history because, after that, the company entered the international market with a considerable number of orders from all around the world. Now the company is rebranded as LSEG Technology, the technical support provider for the mother company LSEG. It delivers six different capital market specialized products & services

* Millennium Exchange
* Millennium Surveillance
* Millennium SOR
* Millennium Market Data
* Millennium PostTrade
* Millennium LiveOps

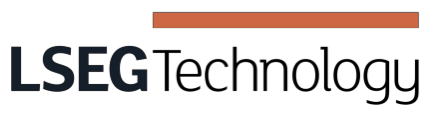
 Figure 1.2 below illustrates the logo of LSEG Technology, and Figure 1.3 depicts a magnificent view of the LSEG technology in Malabe.

Figure 1.2: Logo LSEG Technology

Figure 1.3 LSEG Technology, Malabe

**1.2.2 Company Vision**

“MillenniumIT is the partner of choice for organizations needing real-time, high performance, agile & resilient capital market technology. We set the global standard for multi asset, pre to post-trade STP solutions for financial markets.”

**1.2.2 LSEG Technology Divisions**

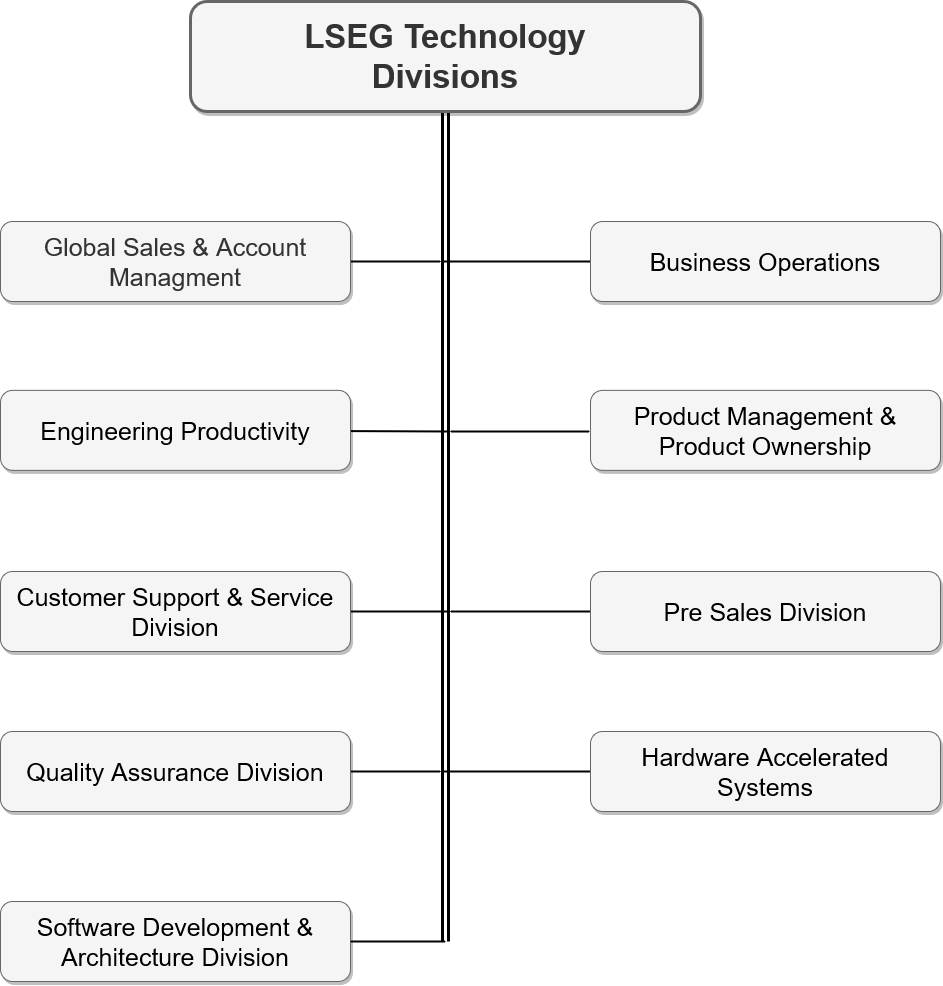
**** Figure 1.4 below depicts different division of LSEG Technology. There are around 9 divisions collaboratively work to achieve their goals.

Figure 1.4: LSEG Technology Divisions

**1.2.3 LSEG Technology Management Structure**

Since 2009, the group has made several other acquisitions increasing the size and scope of the LSE group. Figure 1.5 illustrates the leadership structure expanded according to different divisions. As a trainee, I was fortunate to work in the Hardware Accelerated Systems division.

**1.3. SUMMARY OF TRAINING EXPOSURE**

LSEG Technology, Hardware Accelerated Systems division requires interns to work on a research project. I started working on a project called “Heteregenous Framework for Gross Exposure Calculation.” This project is related to Hardware Acceleration, and some financial calculations related to the stock market are accelerated using a GPU-FPGA heterogeneous platform.

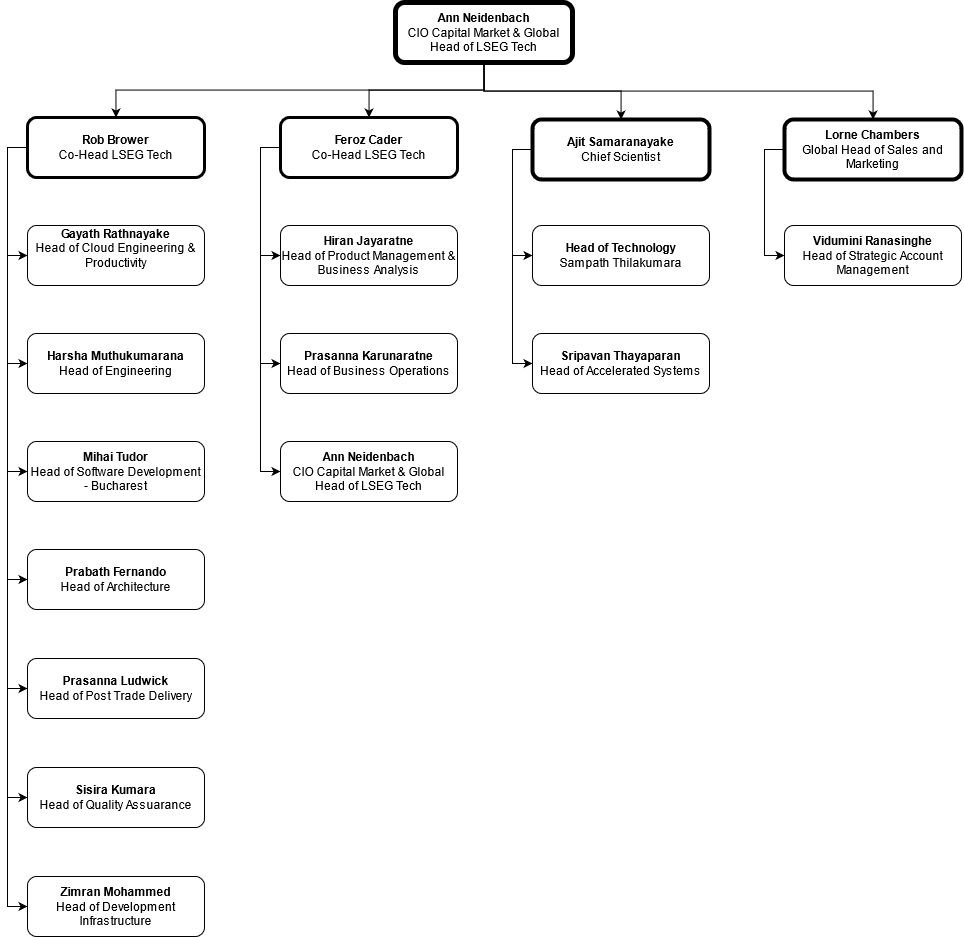
**** The Training period was spent learning about existing heterogeneous computing techniques using OpenCL, Vitis development environment, Writing OpenCL Kernels for NVIDIA GPUs. Also, how to properly write test cases and industry practices like agile, scrum, and different associated events. In the latter part of the training, I evaluated and compared the system. Finally, I put my time into documenting my work for future use.

Figure 1.5: LSEG Technology Leadership

Chapter 2

HARDWARE ACCELERTED SYSTEMS DIVISION

## **2.1. INTRODUCTION**

This chapter will describe the Hardware Accelerated system division, the leadership and their main scope of work, and my internship experience.

## **2.2. LEADERSHIP**

As shown in Figure 2.1, the Hardware Accelerated System division is led by Mr. Thayaparan Sripavan, Head of Hardware Accelerated Systems division, and there are two logical teams inside the division. They are the GPU team and the FPGA team.

The FPGA team is lead by Mr. Chandima Indatissa, Associate Architect, and Mr. Nisal Panagoda, Tech Lead, and their work with FPGA at the RTL level. There are around ten members of the team, and their main platform is XILINX.

I worked with the GPU team, and there were six members, including me. GPU team is led by Mr. Janaka Perera, Associate Architect, and their work is based on using High-Level Synthesis (HLS) tools to accelerate projects, including financial market calculations. They also have expertise in CUDA programming, and their main platforms are NVIDIA, XILINX, and INTEL.

Diagram

Description automatically generated

Figure 2.1 Leadership Team of the Hardware Accelerated Systems Division

## 

## **2.3. PRODUCTS**

The Hardware Accelerated Systems division has supported many solutions, and out of them, one of the two key products is Group Ticker Plant (GTP) and Millennium Risk Product.

**2.3.1 LSEG Group Ticker Plant (GTP)**

The LSEG GTP is a real-time market data feed protocol, and also it is a low latency market data feeding protocol that is designed on a common architecture supporting multi-asset class. The GTP feed disseminates market data via multicast, broadcast in UDP network packets, and delivers overload balanced IP multicast channels. These IP multicast channels support TCP/IP replay (message gaps up to 65,000 most recent) and recovery services (for recovery from larger gaps).

**2.3.2 Millennium Risk Product**

Millennium Risk is a resilient and extensible multi-instrument real-time risk management & collateral platform. It is used in trading & clearing workflow to minimize counterparty risk & enable stakeholders to maximize capital utilization. Millennium Risk helps post-trade businesses stay compliant with CPMI-IOSCO principles of the financial market infrastructure (PFMI) and provides controls for trading venues that aspire to offer pre-emptive gating of transactions or reactive collateral calls in real-time through the stringent market, credit, and liquidity risk measures.

**2.4 EXPERIENCE GAINED**

When I was joining the team, they were working on sprints, and each of the work items was related to Vitis HLS. The whole team was preparing/learning about an upcoming large project which they are planned to implement using Vitis HLS.

It was a good learning experience because since all the team members are learning, they scheduled sessions to share their knowledge and findings with other team members. These sessions were called Knowledge Transferring Sessions (KT Sessions). In KT sessions, after the presentation, other team members were asked questions to make clarifications.

Chapter 3

INTRODUCTION TO THE GROSS EXPOSURE CALCULATION

## **3.1. INTRODUCTION**

This chapter describes financial terms I have come across during the internship and the Gross Exposure calculation that I accelerated as the internship project.

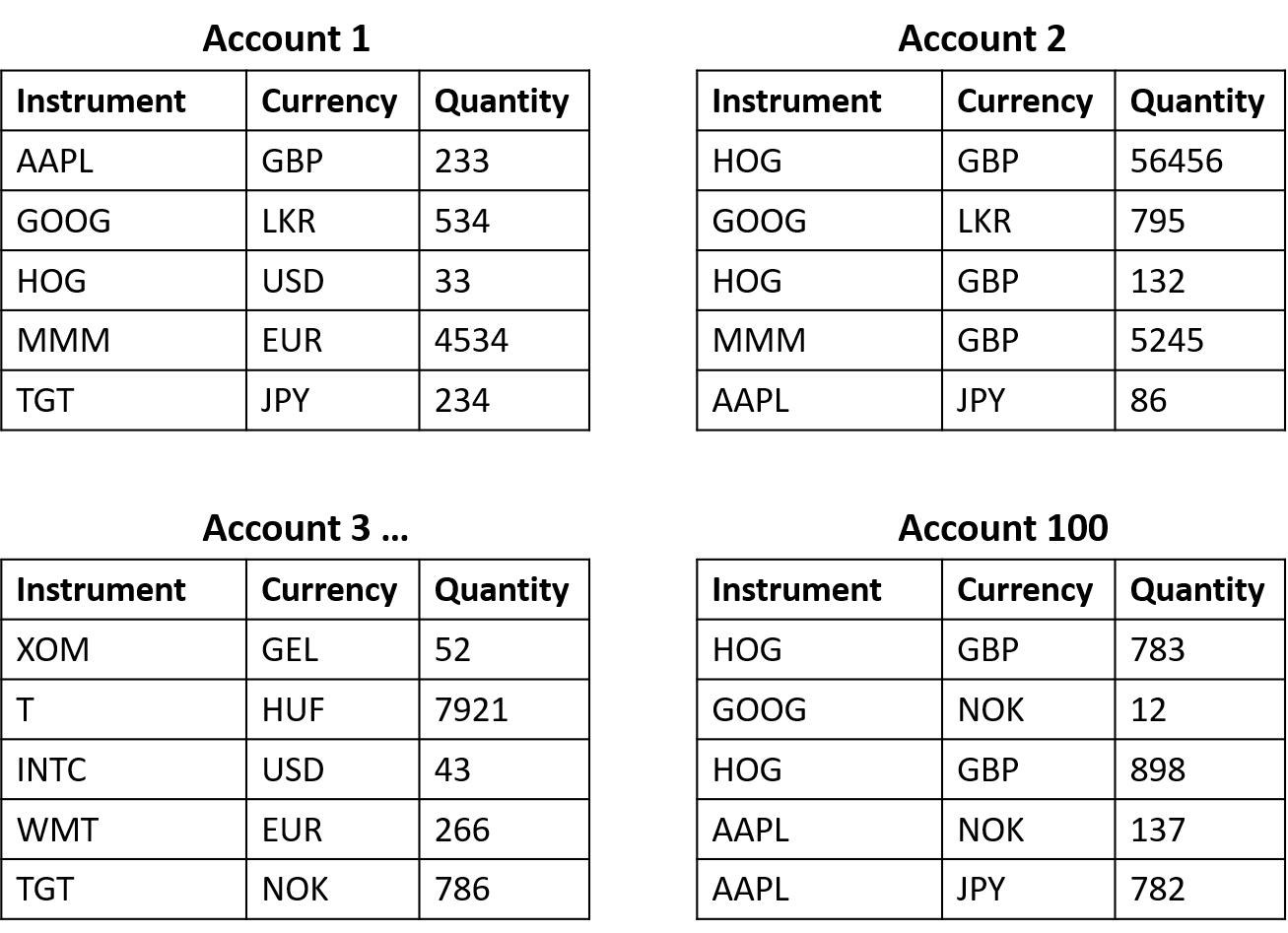
## **3.2. GROSS EXPOSURE CALCULATION**

In the stock trading system, the gross exposure calculation is a time-sensitive real-time calculation that should be sensible for microsecond scale delays. Since my project is related to the financial market, one of the major challenges was learning those financial terms to understand the project. To learn these terms, I got help from my mentor and resources from the internet. Table 3.1 describes financial terms I have come across during this project.

Table 3.1: Financial Terms

|  |  |
| --- | --- |
| Financial Term | Explanation |
| Instrument | An implement with which to store or transfer value or financial obligations |
| Currency | Medium of exchange for goods and services |
| Quantity | Number of shares one wishes to buy or sell |
| Market Price | The market price of a stock is the price that it sells for on the open market at a given point in time |
| Position | A single stock that a trader owns in his portfolio |
| Position Update | Adding/removing a position from a trader’s portfolio |
| Market Data Update | Price change in a single instrument-currency pair |
| Gross Exposure | Gross exposure measures an investment fund's total exposure to financial markets |

There are several accounts (portfolios) for each of the users. When my mentor explained the gross exposure calculation to me, he told me to consider these accounts as different stock-broker firms. Therefore, for the sake of understanding, I am referring to them as different stock-broker accounts. In each of the accounts, there are several positions. Simply, we can think of a position as a tuple that includes an instrument, currency, and quantity. Figure 3.1 illustrates a sample system with a set of accounts.

Figure 3.1: Account Details

As you showed in Figure 3.1, we can use instrument-currency pair as a unique entry in an account. Some of the essential statistical data are,

* There can be 20,000 maximum number of positions per account.
* The total number of currencies is 15.
* The total number of instruments is 1000.

Position distribution in accounts is purely random. Each instrument-currency pair has a price at a given time. Table 3.2 illustrates a sample price list at a particular time.

Table 3.2: Sample Price List

|  |  |
| --- | --- |
| Insturment-Currency pair (IC) | Price |
| AAPL-GBP | 432.34 |
| TGT-JPY | 89.55 |
| AAPL-USD | 4589.00 |
| MMM-EUR | 38.93 |
| HOG-USD | 7865.95 |
| GOOG-LKR | 624.00 |

To calculate the account level gross exposure, we need to take the multiplication of quantity and price of every position and sum them up in each account.

An example calculation using sample data of Account 1 in Figure 3.1 and Table 3.2 is shown below.

**3.3 Experience Gained**

Chapter 04

DESIGN AND IMPLEMENTATION – GROSS EXPOSURE CALCULATION

## **4.1. INTRODUCTION**

The GPU team needed to investigate the performance of gross exposure calculation using a GPU-FPGA heterogeneous computing system. They assigned the project to me as a research project. As I started working on my project, first, my mentor asked me to create a timeline. Therefore, I created a timeline including the following milestones.

* A quick literature survey to find existing solutions/approaches to the current problem.
* A design that should include my approach to the problem.
* Implementation of the project.
* Find bottlenecks and Optimize the run-time.

In this section, I intend to describe my work during each of these milestones.

Since I got experience in OpenCL programming from my final year university project, my line manager and the rest of the team members suggested using OpenCL in a GPU-FPGA heterogeneous environment to accelerate the gross exposure calculation.

**4.2. Proposed Solution**

Two operations mainly change Account-level gross exposure, and they are Market Data Update and Position Update. In the below two sections, I describe these two operations and how they affect the account level gross exposure.

**4.2.1 Market Data Update**

According to Table 3.2 in Chapter 3, each instrument-currency pair has a unique price. In the stock market environment, this price is changed in real-time. Since this price is used to calculate the account level gross exposure, when the price is changed of a particular instrument-currency pair, the gross exposure is each account that has a position associated with it invalidate.

GPUs use Single Instruction Multiple Data (SIMD) Parallelism. When a price for a particular instrument-currency pair is changed in each account, the latest gross exposure needs to efficiently calculate. Therefore, in the proposed solution, the Market Data Update calculation is run on the GPU.

**4.2.2 Position Update**

As shown in Table 3.1 in Chapter 3, there are several positions in each account. A position update is a process of adding a new position to an account. When a new position is added to an account, the gross exposure of the account is changed. To get the latest gross exposure, we need to execute the following formula.

In the proposed solution, an FPGA device is proposed for this calculation. Since FPGAs are customizable devices, the expectation was to finetune the hardware design to do this calculation in nanoseconds.

**4.3 Heterogeneous Computing with OpenCL**

OpenCL is a software system that lets programmers write a portable program capable of using all resources available in some heterogeneous platform. A heterogeneous platform may include multi-core CPUs, one or more GPUs, FPGAs, and other compute devices. In principle, OpenCL can also be used for homogeneous programming systems such as Intel multi-core processors, where one core can be a host and the others provide improved performance by massively parallel processing. The OpenCL specification is defined in four parts, which it refers to as “models.” These models are briefly described in the below sections.

**4.3.1 Platform Model**

An OpenCL platform consists of one host and one or more devices. It provides an abstract hardware model for devices. A device consists of multiple compute units, and a compute unit consists of multiple processing elements (PEs). There can be several platform models from different vendors that can coexist in a single host. Table 4.1 shows some of the available platforms.

Table 4.1: Different OpenCL Platforms

|  |  |
| --- | --- |
| OpenCL platform | Software Development Kit (SDK) version |
| CPU (host) and CPU (device) | Intel SDK for OpenCL applications |
| CPU (host) and GPU (device) | Nvidia SDK for OpenCL |
| CPU (host) and GPU (device) | AMD APP SDK 3.0 for 64-bit Linux |
| CPU (host) and FPGA (device) | Intel FPGA SDK for OpenCL |
| CPU (host) and FPGA (device) | Xilinx SDAccel Environment |

LSEG Technology Hardware Accelerated Systems division works with Xilinx FPGAs. The reason for choosing Xilinx devices is that Xilinx gives good support compared to Intel in the initial times. Therefore, the company decided to work with Xilinx. Our team members join for Xilinx sessions as well, and they raise any problem which needs to clarify from the Xilinx development team. Also, when our team members find any bug/issue in Xilinx Environment, they use these sessions to raise them as well. Also, they work with NVIDIA GPUs.

**4.3.2 Execution Model**

OpenCL execution model defines how the host communicates with devices. The execution model includes OpenCL context. A context is created with one or more devices. A context is associated with a command queue which is to communicate between the host and devices. The host issues command such as data write, data read, execute kernels, and these commands are stored in the command queue and issued appropriately.

Diagram, text

Description automatically generated There are two types of command queues in OpenCL, and we can specify the command queue type when we create them. The first type is the in-order command queue. In-order command queues guarantee that the execution order of commands issued by the host will be in the order they are put to the command queue. An additional synchronization mechanism is unnecessary for in-order command queues. Figure 4.1 shows the command execution pattern in in-order command queues.

Figure 4.1: OpenCL In-order Command Queue

Diagram

Description automatically generated The other type is out-of-order command queues in which the order of the execution is not guaranteed. As shown in Figure 4.2, the command execution may overlap in out-of-order command queues. Moreover, we must take any dependencies into account between commands. If there are dependencies, we must use OpenCL events to preserve synchronization.

Figure 4.2: OpenCL Out-of-order Command Queue

**4.3.3 Kernel Programming Model**

Kernels are functions executed on the device end. There are two types of Kernel programming models in OpenCL, namely “Single work item” and “NDRange kernels”. NDRange kernel has many work-items, whereas a Single work item kernel has only one. NDRange kernels use a GPU-like programming style where multiple work-items are processed in parallel. Single work item kernel uses a CPU-like programming style, where only one work-item is processed throughout the execution.

**4.3.4 OpenCL Memory Model**

In the heterogeneous system, mainly there is the host memory and the device memory. From the host side, we can divide device memory further. In the OpenCL memory model, there are four memory types.

1. Global Memory: Accessible from both host and the device.
2. Lobal Memory: Accessible to all the work-items in a work-group.
3. Private Memory: Accessible to a work-item.
4. Constant Memory: Accessible to the device, and it is a read-only memory.

**4.4 Literature Survey**

One of the crucial challenges is the data transferring between the devices. Therefore, my literature survey focused on data transferring mechanisms between the GPU and FPGA without the host (CPU) interaction. I spent several days reading-related research papers.

According to the literature survey, EngineCL is a useful framework in an OpenCL heterogeneous computing environment. It is an extension of OpenCL. EngineCL framework can be used to divide a particular calculation among different devices, and the final output will be the merge of individual calculations of each device. After testing the EngineCL framework, I realized that it does not support my project since the computation in GPU and FPGA are totally different.

SnuCL is another framework developed by SEOUL National University, South Korea. Initially, when I was reading about the framework of a research paper, I was impressed. Then I emailed the developers to get further details. As per the reply, since FPGA and GPU that I was going to use are from two different vendors, the SnuCL framework will use host memory to accomplish the data transfer.

I was looking for a Direct Memory Access (DMA) approach between devices. After the literature survey, I realized that direct data transfer between devices is not possible since FPGA and GPU are from different vendors.

**4.5 Design**

According to the timeline, I started working on the design. My mentor advised me to use Object Oriented Programming (OOP) concepts and create a class diagram. Since I have not used OOP concepts in a high-performance computing system, it was a novel experience. In my university projects, I have used OOP to build software, and I realized that even kernels are wrapped as host-side functions and embed to an OOP design.

I was inexperienced with C++ OOP implementations and somehow managed to learn it using resources on the internet. Since I had learned OOP concepts in Java at the university, it helped me a lot to grasp OOP using C++ in a short period.

**4.5.1 Class Diagram**

Diagram

Description automatically generated As shown in Figure 4.3, the system class contains an account object array that has 100 accounts. Request queue contains either position update requests or market data update requests in the order they come to the system. A C++ map is used to map the “instrument + currency” string to a unique integer value called instrument-currency-id. The system class has a GpuDeviceHandler object and a FpgaDeviceHandler object which includes device platformId, context, command queue, and kernel program.

Figure 4.3: Class Diagram of Gross Exposure Calculation System

**4.5.2 Flowchart**

Figure 4.4 depicts the execution flow when a request comes to the system. When a position update comes to the system, first, a new position is added to the account corresponding to the position update request. At the same time, a position writes command will be added to the GPU command queue. Then position update kernel will be invocated using the FPGA command queue, and the updated gross exposure is read, and it will be saved in the corresponding account object.

Diagram

Description automatically generated When a market data update comes to the system, it will update the latest price corresponding to the instrument-currency-id on the host side, and accounts that affect the request will re-calculate the gross exposure.

Figure 4.4: Flowchart of Gross Exposure Calculation

**4.5.3 Sequence Diagram**

Graphical user interface, application

Description automatically generatedOne of the crucial challenges in multi-device calculation is the synchronization between devices. Since the calculation runs in parallel on both devices, it is necessary to have a proper synchronization mechanism. Figure 4.5 shows the synchronization mechanism I came up with.

Figure 4.5: Sequence Diagram

**4.6 Kernel Design**

**4.6.1 FPGA Memory Buffer Allocation**

It is necessary to understand the device memory allocation when we need to talk about the kernel design. I pre-allocated an array with the size of 100 doubles to store the updated gross exposure on the device memory.

**4.6.2 FPGA Kernel**

FPGA kernel is a simple delta calculation. When a position update request comes to the system, it will use previous gross exposure, quantity, and price to calculate the latest gross exposure.

**4.6.3 GPU Memory Allocation**

Since there can be 20,000 positions per account, I pre-allocated a large array size of 20,000 \* 100 positions. When a position is added to the account, it will be stored as a structure. This structure has ‘instrument-currency-id’, ‘quantity’, ‘stock-exposure’ as values. To store the current price of each ‘instrument-currency-id’, I pre-allocated a large array size of 15,000 prices. Figure 4.6 shows the GPU buffers.

Graphical user interface, application

Description automatically generated

Figure 4.6: GPU Buffers

**4.6.4 GPU Kernel**

When there is a price change in a particular instrument-currency-id, all the accounts which have that instrument-currency-id need to calculate the Gross Exposure. It is necessary to do this with minimum latency. Basically, this calculation is an addition of all the quantity \* price values in an account. Using the parallel computing capability of the GPU, we can use the parallel reduction approach to calculate the account level gross exposure in a very less time. Also, since GPU can allocate resources for multiple kernels, all the accounts with invalidated gross exposure can re-calculate in parallel. As shown in Figure 4.6, the market data update is calculated using two GPU kernels.

After executing kernel 1, each work group will execute the reduction and will generate a result. To get the final gross exposure value, we need to add these partial results. There are two approaches to this. The first approach is to fetch partial results back to the CPU and do an addition to get the updated gross exposure. Then it will not allow the GPU to execute other kernels thus, GPU will not calculate gross exposure for each account in parallel.

Diagram

Description automatically generated To overcome this issue, I designed another nested kernel (Kernel 2 in figure 4.6) which will be called soon after the execution of Kernel 1. It will use a single work-group and using parallel reduction approach will add all the partial results in each work-group to get the updated gross exposure, and the updated gross exposure will be saved in the Account Gross Exposure buffer (in figure 4.6)

Figure 4.6: GPU Kernel for an Account

**4.4 Implementation**

**4.5 Difficulties**

**4.6 Optimizations**

**Experience Gained**

what I learnt – how industry is doing literature survey.

Chapter 5

RESULTS AND DISCUSSION

5.1. INTRODUCTION

This chapter elaborates the evaluation of the proposed system and the experimental setup used to test the implementation.

5.2. EXPERIMENTAL SETUP

Table 5.1: Experimental Setup

|  |  |  |
| --- | --- | --- |
|  | GPU | FPGA |
| Host | Intel(R) Xeon(R) CPU E5-2687W v3 @ 3.10GHz | Intel(R) Xeon(R) CPU E5-2687W v3 @ 3.10GHz |
| Accelerator | Tesla P100 | xilinx\_u200\_xdma\_201830\_2  (HW-Emulator) |
| Vendor | NVIDIA | XILINX |
| Operating System | CentOS 7 | CentOS 7 |
| Compiler/SDK | CUDA 10.2 | Vitis\_HLS 2020.2 |

5.2. EVALUATION

**5.2.1 Evaluation of Market Data Update**

One of the major challenges was to test our system in real-time because it is hard to build a real-world test scenario in the performance testing phase.

First, I distributed random positions (positions with random instrument-currency-id) among the accounts, and I have ensured that this is a normal distribution. Figure 5.1 illustrates the number of position distributions among accounts. The peak number of position count is 3500 in account id 50.

I also created a random position distributor to select instrument-currency-id for each of the positions when they are an addition to an account.

Figure 5.1: Position Distribution

**5.2.2 Results**

Performance values for 10 runs of market data update is shown in Table 5.2

Table 5.2: Performance Values of GPU Kernel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Run | Price Write Time (ms) | Kernel Time (ms) | Gross Exposure Read Time Device to Host (ms) | Number of Invalidated Accounts |
| 0 | 0.00096 | 0.346624 | 0.001728 | 30 |
| 1 | 0.00112 | 0.466784 | 0.001536 | 42 |
| 2 | 0.00096 | 0.529568 | 0.001344 | 48 |
| 3 | 0.000928 | 0.522432 | 0.001152 | 48 |
| 4 | 0.00096 | 0.055424 | 0.00112 | 5 |
| 5 | 0.000928 | 0.426656 | 0.001152 | 39 |
| 6 | 0.000928 | 0.455552 | 0.00112 | 41 |
| 7 | 0.00096 | 0.478464 | 0.001152 | 44 |
| 8 | 0.000928 | 0.447808 | 0.001152 | 41 |
| 9 | 0.00096 | 0.413984 | 0.001152 | 37 |

**5.2.1 Evaluation of Position Update**

Timeline

Description automatically generated Position update evaluation is done on Vitis hardware emulator. I used Vitis profilers, reports, waveform analyzer (Shown in Figure 5.2, 5.3) to obtain performance values.

Graphical user interface, application, table

Description automatically generatedFigure 5.2: Kernel Observation using Vivado Waveform Analyzer

Figure 5.3: Kernel Report using Vitis Hardware Emulation

**5.2.2 Results**

|  |  |
| --- | --- |
| Kernel Time (ns) | Latency (cycles) |
| 493 | 148 |

Chapter 5

NON-TECHNICAL EXPOSURE

5.1 INTRODUCTION

Most of the LSEG Technology divisions including the Hardware Accelerated Systems division uses Scrum practices. Therefore, I was able to get a good exposure on industry level scrum practices. In first few sections I intend to describe my experience on it.

LSEG Technology organizes a number of events and fun activities for employees’ welfare. ‘MiClub’ is an association of employees of LSEG Technology, that organize these events to get together with each other and have fun together. Due to the pandemic situation all the employees were working from home. Therefore, they have limited the number of events and I was able to participate all the virtual events they have conducted. Most of these events are organized by the Human resource management department. Towards the end of this chapter I will be describing my experience of attending to these events.

5.2 SCRUM PRACTICES IN LSEG TECHNOLOGY

Scrum is a framework that helps teams work together. Since it is based on continuous learning and development it is assumed that the scrum team does not know anything at the start of the project and will evolve though learning and sharing. A sprint in the Hardware Accelerated Systems division will last for two weeks.

5.2.1 PRODUCT BACKLOG

Product backlog prioritized list of work for the development team delivery. The Accelerated systems division product backlog is saved in a confluence page. It handles by Mr. Janaka Perera, Associate Architect and Mr. Sahan Henadeera, Associate Tech Lead. This backlog is populated with different work items and they are come from other LSEG divisions as high-performance requirements. In the backlog we can a list of work items which are assigned to different developers and each of the work item is associated with story point which is in the range of 0-5.

SPRINT PLANNING

DAILY SCRUM MEETING

In our team daily scrum meeting is lead by the Scrum Master, Mr. Sahan Henadeera, Associate Tech Lead and it is named as ‘GPU-sync’. Everyday the development team has to participate for the GPU-sync and they are presenting their current work progress. The expectation of this meeting is to ensure that all the developers are working smoothly, and the scrum master needs to know if there are any blockers the team has. At this meeting, any team member can raise their issues and the scrum master will take them to account and he will resolve them.

SPRINT GROOMING

In the sprint grooming session, Mr. Janaka Perera takes the lead and he will be evaluating the last ended sprint and he will be asking from each team member how much work is spilling from that particular work item he worked on. Depending on that, he will create new work items from spilling tasks and will be added to the next sprint as new work items.

Each time when he is adding a new work item, the team will go for a planning poker session and estimate the number of story points to assign. It was a novel experience for me, because it was the first time, I engaged in a planning poker session.

SPRINT RETROSPECTIVE

5.2.2 SCRUM ROLES

Product Owner

In my team the Mr. Janaka Perera plays the role of product owner and he ensures manages