國 立 中 正 大 學

資訊工程研究所碩士論文

用於資料密集型運算之高擴展性FPGA硬體加速平台

High Scalability FPGA-based hardware accelerator for data-intensive computation

研究生 ： 李岱樺

指導教授 ： 鍾菁哲 博士

中華民國 一零四 年 七 月

國立中正大學碩士班研究生

學位考試同意書

本人所指導
資訊工程學系

研究生 李岱權 所提之論文

用於資料密集型運算之高擴展性FPGA 硬體加速平台 High Scalability FPGA-based Hardware Accelerator for Data-Intensive Computation

同意其提付 碩 士學位論文考試

指導教授 陳雪喜 簽章

104 年 6 月 11 日

國立中正大學碩士學位論文考試審定書

資訊工程學系

研究生 李岱橿 所提之論文

用於資料密集型運算之高擴展性FPGA硬體加速平台 High Scalability FPGA-based Hardware Accelerator for Data-Intensive Computation
經本委員會審查，符合 碩 士學位論文標準。

學位考試委員會
召集人 李順龍 簽章

委員

王廷基	盛鐘
鍾葉青	

指導教授 鍾葉青 簽章

中華民國 104 年 7 月 2 日

博碩士論文授權書

（本聯請裝訂於論文紙本書名頁前空白處，供學校圖書館做為授權管理用）

ID:103CCU00392090

本授權書所授權之論文為授權人在 國立中正 大學(學院) 資訊工程研究所 系所 _____ 組 103 學年度第 二 學期取得 碩 士學位之論文。

論文題目： 用於資料密集型運算之高擴展性FPGA硬體加速平台

指導教授： 鍾葆哲、Ching-Che Chung

茲同意將授權人擁有著作權之上列論文全文(含摘要)，提供讀者基於個人非營利性質之線上檢索、閱覽、下載或列印，此項授權係非專屬、無償授權國家圖書館及本人畢業學校之圖書館，不限地域、時間與次數，以微縮、光碟或數位化方式將上列論文進行重製，並同意公開傳輸數位檔案。

紙本論文：茲同意將授權人擁有著作權之上列論文全文(含摘要)，提供讀者基於個人非營利性質之閱覽或列印，此項授權系非專屬、無償授權國立中正大學圖書館做為編目上架及公開陳列閱覽使用。

☐ 校內外立即開放
☐ 校內立即開放，校外於 2020 年 12 月 31 日後開放
☑ 校內於 2020 年 12 月 31 日；校外於 2020 年 12 月 31 日後開放
☐ 其他

授權人：李岱樺

簽　名：李岱樺　　日期：2015 年 08 月 15 日

摘要

隨著物聯網興起，不論是智慧型手機、社群平台或感測裝置，所有設備都希望能夠經由網路跟伺服器交換資料，讓使用者能夠即時得到回覆，也因此面臨大數據 (Big data) 的時代，海量的資料、不同的資料型態、隨時隨地的資料傳遞，隱藏在資料裡的價值，都是處理大數據會面對的問題。

分散式系統和雲端計算也越來越普遍，希望藉由多個運算伺服器或叢集做平行化的運算，和互相使用存取空間讓容量上升，來彌補個人電腦儲存的空間不足和 CPU 運算速度的瓶頸。也因為個人電腦上的瓶頸，越來越多人使用硬體加速平台(hardware accelerator)來分擔運算的負擔，最常見的就是可程式邏輯陣列(FPGA)和圖形處理加速器(GPU)。硬體加速平台適合做高密度且獨立的運算，它具有很多運算單元可以達到運算平行化。

K-means 分群演算法是資料探勘(data mining)的一種，可以用來分析資料間的關聯性或是圖片的優化，而本論文也實現 K-means 分群演算法分析大型的資料，呈現出硬體加速平台的優勢。因此，我們建立了一個以 FPGA 為基礎的擴充平台，利用網路分享器來提升擴充性(scalability)。電腦端把資料整理好傳給 FPGA，經由FPGA計算完後，再把結果傳回給電腦端。電腦端負責較不規則的運算處理，工作分配和處理 FPGA 的運算結果，而 FPGA 只要專注於運算的部分。最後再以執行時間來評斷系統的表現。

關鍵詞: 大數據，現場可程式邏輯門陣列硬體加速平台，K-means 分群演算法，以 FPGA 為基礎的擴充平台

Abstract

Nowadays, the smart phones, web systems, and wireless sensors are enabled to connect the Internet, and response to user in real time. Therefore, we are living in the internet of things (IoT) era with big data generation. The “4Vs” characteristics of big data such as variety, volume, velocity, and value make them difficultly to be handled.

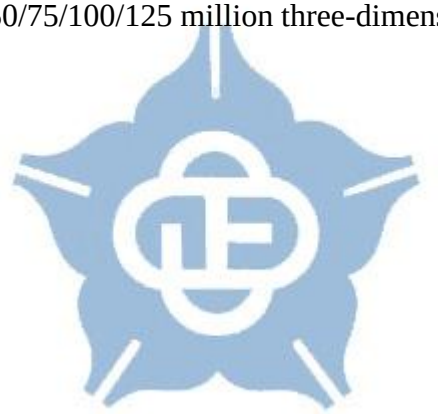
The personal computer is hard to deal with big data, because the capacity of memories and storage devices is not enough and the limited processing rate of the CPU. Therefore, the distributed file system and cloud computing have become popular. Both of them can compute in parallel and share the data of disks. Moreover, the hardware accelerator is suited for data-intensive computation, and the function units of hardware accelerator are processed in parallel. Graphic processing units (GPUs) and field programmable gate arrays (FPGAs) are potential hardware acclerators.

K-means clustering algorithm is one of the data mining techniques. K-means is used to find the relation between the data, or for image processing. Therefore, in this thesis we implement k-means clustering algorithm to analyze the dataset. The FPGA-based hardware accelerators that communicate with computer through switch are proposed. The computer wraps data into packets to FPGA, and it receives data after the FPGAs are finished computation. In addition, the host computer is employed as the master to manage data and dispatch jobs, and the FPGAs are focused on accelerating data computation. Finally, the proposed system performance is compared with the benchmark execution time.

Keywords: big data, field programmable gate, K-means clustering algorithm, PGA-based hardware accelerators

Content

List of Figure



List of Table

Chapter 1

Introduction

1.1 Introduction to Big Data

Nowadays, the growing popularity of web systems, mobile devices, social media, surveillance videos, and wireless sensors generate large amounts of data from difference sources. Specific technologies such as network, internet, server, email and mobile are growing up [1]. The smart phones have launched a wave of revolution, not only the user can directly access vast amounts of smart phone applications but also enables individuals to use applications to reach mass audiences [2]. The conception of Internet of Things (IoT) is connectivity of network entities embedded with device that can exchange data to physical objects. However, Internet of Things (IoT) is evolved to Internet of Anything (IoA), Fig. 1.1 shows the IoA ecosystem by [3].

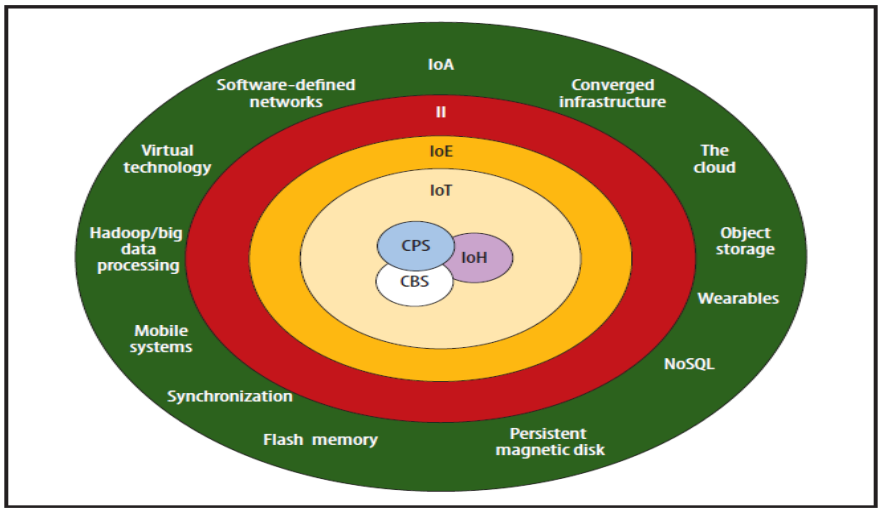


Figure 1.1: Presents [3] vision of the IoA ecosystem

IoA can possibly imagine everything as part of the network ecosystem, like internet operating system – a common applications have ability to accommodate any and all sensor inputs, system states, operation conditions, and data context [3].

An IDC report [4] predicts that data volume will increase from 130 exabytes to 40,000 exabytes by a factor of 300, from 2005 to 2020. According to reference [5], the dataset is produced rapidly. For instance, the user searches by Google is more than 38000 per second, and 2 terabyte of photos are uploaded to social media every day, and videos on Youtube are watched 4 billion hours long for a month. The data is exponential growth as shown in Fig. 1.2. It is obviously the data explosion generation is coming.

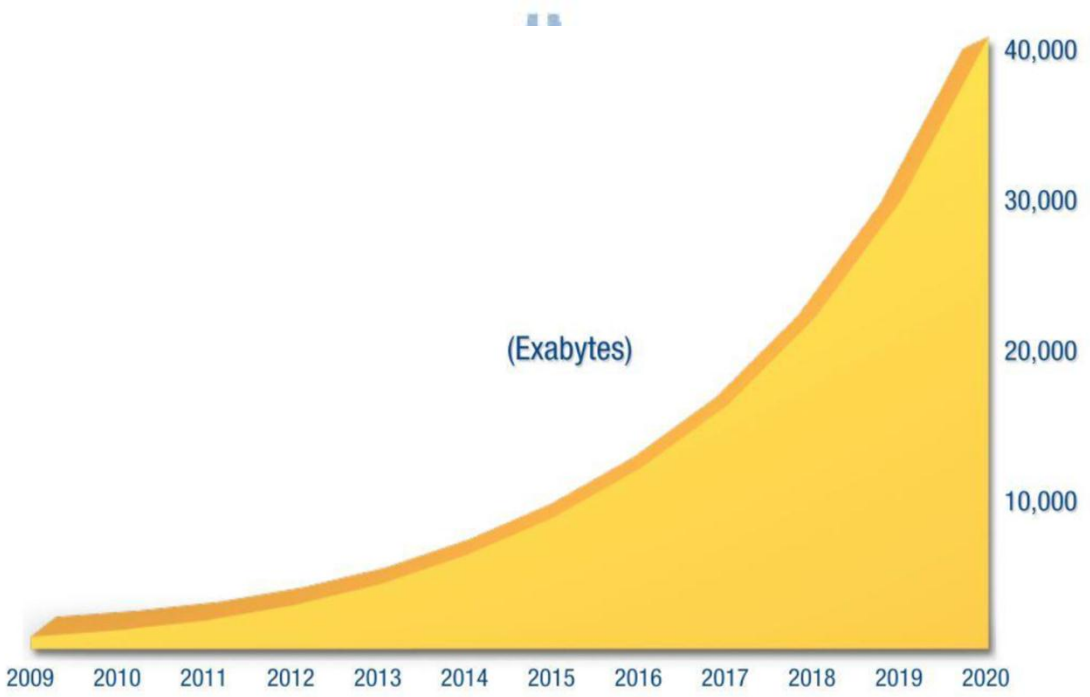


Figure 1.2: 50-fold Growth form beginning of 2010 to the end of 2020 [4]

In fact, big data is difficult to be handled, not only the amount of data is large, but also it has special properties. The properties of big data make it not easy to be handled. The properties include of big data variety, volume, velocity and value, the "4Vs" is widely applied to the definition of big data [6].

The variety means the data produced is not of one flavor, they have structured, semi-structured and unstructured data, so traditional database systems are hard to handle them. For example, users generated contents in social media, IT industries log files, and sensor devices data, all these data are different structure.

The volume means the volume of big data is quite larger than traditional data. In Fig. 1.2, the growth rate is incredible and traditional database or hardware are notable to store big data.

The velocity means big data must be analyzed at a rate that matches the speed of data production. In the Internet of Everything (IoE) environment, real-time applications report must be quickly. User wants to know everything immediately via device, such as smart phones, detection sensors and RFID.

Finally, by analyzing big data, some useful values can be found, for instance, business trends and commercial benefits. But, it has a gap in between the different specialty, like business leaders concern how to add value in product and get more benefit unlike IT leaders concern the low power or processing [7].

There are more characteristics like complexity and variability [7] [8]. Variability means data flow is inconsistencies, to maintain data loads is a challenge. Complexity means the degree of transform data across systems coming from different sources.

For these properties, big data have many challenges and issues [6][7][8][9], we summarize above references discussion and show as follows:

Data Privacy and Security: User can store privacy information in many of online services, cloud, social media and mobile phones, but we don’t know where is data stored. Everything enables to connect the Internet, personal information, business privacy and financial data are probably steal by hacker. We can encrypt privacy data and carefully upload data to protect ourselves.

Data management and store: Data are excessive large that the storage devices not have enough space to storing datasets. In addition, many datasets are heterogeneous in type, semantics, structure, video and text. To conquer this problem, the storage devices need scalability, expanding system capacity should be easy and convenient, while increasing capacity does not need to shut down the system. Distributed file system allows user to share files and share data to many hosts. Many hosts can interconnect to the server, and it can handle heterogeneous data and have great scalability. However, distributed file system has others challenges, like requirements response time, power consumption, data transform, and static storage may not be satisfied by dynamic data growth.

Data analysis and application: To analyze entire dataset is difficult, volume of dataset is huge but lots of information is worthless, the analysis time is hard to be real-time. In addition, the available data becomes challenge, privacy dataset like enterprise financial information, technique or valuable logs, are hard to be shared, but we need to cooperate with different professionals. However, we expect the valuable benefits in big data, machine learning and data mining [10] are sophisticated analytical technologies to explore tendency.

Nowadays, computing is transformed to commodity, that delivers infrastructure, platform and software in a manner like traditional utilities water, electricity natural gas and telephone network. It seems the 5th utility that services consumer's requests any time and pays as the usage of service [11]. National Institute of Standards and Technology (NIST) [12] defines cloud computing (shown in Fig. 1.3) as follows five essential characteristics, four deployment models, and three service models.

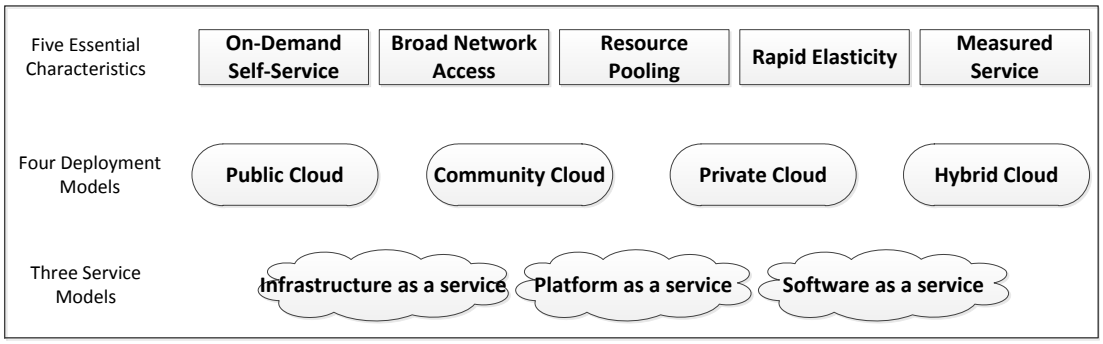


Figure 1.3: Cloud computing definition by NIST

Five essential characteristics:

- On-demand self-service: Customer can unilaterally use computing capabilities without requiring service provider.
- Broad network access: User can access cloud service at any time through standard mechanism with client platforms.
- Resource pooling: Provider uses a multi-tenant model pool computing resources to serve multiple consumers.
- Rapid elasticity: Capabilities can be elastically provisioned and released with customer demands.
- Measured service: Cloud provider measures and monitors resource usage, user pays appropriately and transparently based on utilization.

Four deployment models:

- A public cloud represents publicity accessible via the internet and third-party service providers, and it may be owned by a large organization or a company offering cloud services.
- A community cloud is shared by several organization, community members can jointly use cloud data and applications.
- A private cloud is owned by single organization, it offers increased security at

a greater cost.

- A hybrid cloud is composed of two or more different deployment models that maintain unique entities, but enable data and application portability between clouds by standardized or proprietary technology.

Three service models:

- Infrastructure as a service (IaaS): The capability provided machines, storages, networks, and other fundamental computing resources, that consumer can deploy or run software such as operating systems or applications, but disables to manage or control the underlying cloud infrastructure. (ex: Amazon EC2, OpenNebula)
- Platform as a service (PaaS): The capability provided platform that consumer can deploy applications using programming language or tools onto virtualized cloud platform. (ex: Google App Engine, Hadoop, Microsoft Windows Azure)
- Software as a service (SaaS): The capability provided customer can use provider's application running on a cloud infrastructure through web browser or a program interface. (ex: Google Apps, mail services, EyeOS)

Obviously, processing big data is a challenge, both hardware and software are evolved to adapt the characteristics of big data. Therefore, many software frameworks and file systems are developed to against big data issue, such as NoSQL database, Google file system, Facebook's photo storage haystack and Hadoop.

NoSQL is complemented replacement relational database management systems (RDBMS). One of the most important is that NoSQL not uses SQL as a query language nor based on tables. NoSQL systems have six features [13] and high scalability, reliability, and availability, that are suitable for unstructured data and management of datasets. The primary way differs from relation database is NoSQL

systems supports many data models, such as Key Value Databases, Column Oriented Databases, Document Databases, Graph Databases and Extensible Record Databases [6][14].

Google file system (GFS) [15] runs on hundred inexpensive machines on Linux operation system, and GFS has fault tolerance and atomic data recovery when inexpensive commodity hardwares are failure. GFS is composed of a single master, multiple clients and multiple chunkservers. Its architecture is shown in Fig. 1.4. GFS client program communicates with the master and chunckservers through system APIs. GFS master manages all file system metadata, metadata records namespace and data location of chunkserver. Files are divided into chunks (64MB) and saved in chunckservers. Each chunk is identified by an unique 64-bit chunk handle and is replicated across least three chunckservers. GFS chuckserver sends read or write command by the chunk handle and byte range. [16] lists GFS's main characteristics like big data are divided into chunks and simplify management system. However, there are some disadvantages in GFS, such as small size files have poor performance and single master may restrict scalability and reliability of system.

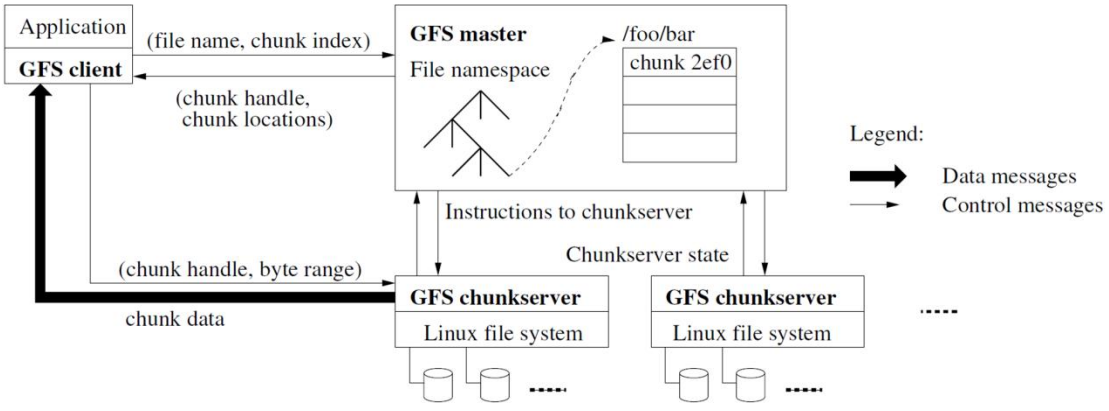


Figure 1.4: Google File System architecture [15]

Facebook developed haystack [16] to store billons small size of photos. Haystack achieves four goals: high throughput and low latency, fault-tolerant, cost-effective and simple. Haystack is composed of three core components: Haystack Store, Haystack Directory and Haystack Cache. Haystack Store manages the metadata of photos in file system. Haystack Directory maintains metadata correspondence between physical volumes and logical volumes, that physical volumes are storage capacity and logical volumes point to physical volumes on different machines. Haystack Cache is internal content delivery network (CDN), which can store for popular photos and protect metadata if CDN nodes fail. Fig. 1.5 shows the flow of serving a photo, haystack directory establishes the URL to each photo, the URL contains information like following: http://<CDN>/<Cache>/<Machine id>/<Logical volume, Photo>. The system uses the logical volume and photo id looks for photos. If the CDN cannot find the photo then delete the CDN address from the URL and contacts the Cache. There is similar behavior when data miss in Cache.

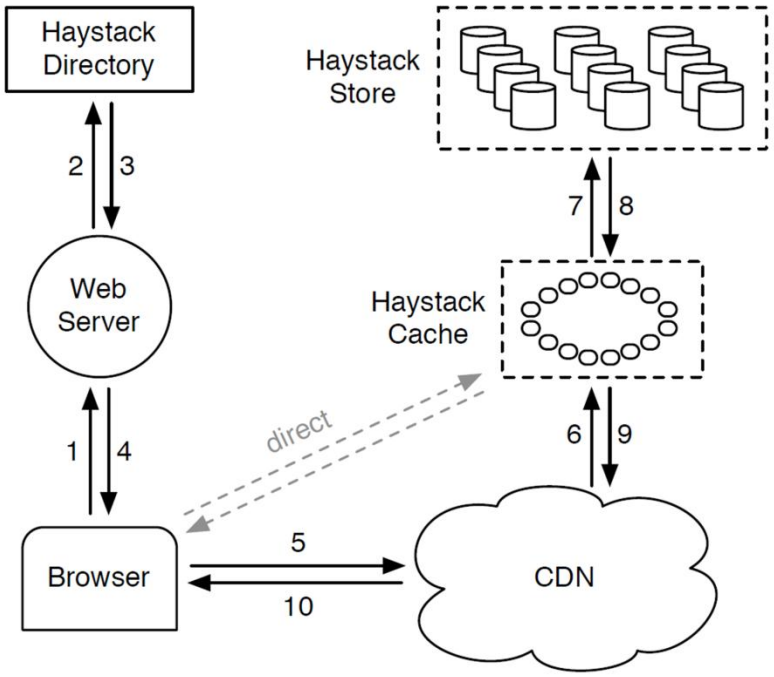


Figure 1.5: Serving a photo in Haystack [16]

Hadoop [17] is an open source software framework that enables massive data storage and distributed processing over large clusters of computing servers. It is mainly composed of two modules: Hadoop distributed file system (HDFS) and MapReduce. An HDFS cluster has a single Namenode that manages file system metadata, namespace and edit log. Secondary Namenode can rebuild Namenode data by the namespace and edits log when system is crashed. In addition, there are several Datanodes that stores the actual data. In HDFS, a file is split into one or more blocks, and each block has several replications to prevent missing data. The MapReduce framework consists of a single master JobTracker and one slave TaskTracker per cluster node. The master JobTracker schedules jobs for the slaves, and monitors and re-executing the failed tasks. The MapReduce framework enables the automatic paralleling and distribution of large-scale computation applications on the large cluster of computing servers. Therefore, it becomes easier to implement big data analysis applications. However, some cases are not suited for Hadoop like low-latency data access or lots of small files, Hadoop is designed for batch-type that is hard to report in real-time. In addition, block size of Hadoop is larger than many small files, and it makes Namenode storing unnecessary metadata. In fact, there are many approaches to improve Hadoop framework, such as a flexible data flow, blocking operators, I/O optimization, scheduling, joins, performance tuning and energy optimization [6].

Besides the development of software frameworks and distributed file systems, the computing servers also require new system capabilities. Computing demand is kept increasing and with only CPUs is dissatisfaction. Therefore, heterogeneous hardware accelerator helps to share computing loading and becomes more attracted in recent years.

1.2 Heterogeneous Hardware Accelerator

A single core computer has reached its performance limitations, the multiple cores work in parallel can achieve speed up, but part of sequential fractions unlikely to be amenable to parallel processing [18]. In Amdahl's law [18], the simple experiment shows that one of the key factors to speedup is to enhance the non-sequential fractions. If we assume that the fractions of the program can be executed in parallel is α ($0 \leqq \alpha \leqq 1$) and can be execution by N processors, consider the execution time T is the execution time to run same program on a single-processor. The equation of enhanced execution time T' can be written in Eq. 1.1 by Amdahl's law.

$$T' = T \times \left((1 - \alpha) + \frac{\alpha}{N} \right) \tag{1.1}$$

The speedup S is shown in Eq. 1.2(a), and the system efficiency E is shown in Eq. 1.2(b). We can estimate the system that has several processors have appropriate benefit. However, if N is very large, the system efficiency becomes very low. For example, if we assume the non-sequential fraction α is 0.5 and number of processors N are 128, and the system efficiency is 1.5%. It means most processors are idle, and the system which run with 128 processors.

$$\text{(a) } S = T'/T \qquad \text{(b) } E = S/N \tag{1.2}$$

N processors or level of N parallelism doesn't means to speed up a factor of N. Many factors create latency or overhead between processor and system such as the data transmit turnaround time and signal deliver between two more processors. We should evaluate benefits when we determine the architecture, and the overhead affects the actual speedup. For example, mobile cloud computing helps the smartphone calculates the intensive computing, but it also has additional costs like data file

migration, application turnaround time and energy consumption [19]. Besides, a single complex processor as a host processor operates with many small and simple cores achieved better speedup and energy efficience than homogeneous cores architecture [20][21]. In this thesis, we focus on the system cooperates with hardware accelerators. Heterogeneous architecture property can separate jobs to hardwares and softwares, and hardware accelerators can excellently work on computational intensive jobs, and software programs can process complex functions and manage data.

Graphic processing units (GPUs), and field programmable gate arrays (FPGAs) are potential hardware accelerators. The main advantages of the GPU are high memory bandwidth, many programmable cores with multi-thread execution in parallel, coding with high level languages like CUDA, and changing functions easier than FPGAs. FPGAs have high density arrays of logic blocks which can execute computation in parallel. User can use Verilog or VHDL to implemented circuits in FPGA, and the vendors provide useful IPs to help developer design [22]. However, design different hardware accelerator architecture have many challenges for developer. It takes a lot of time to implement the design, but it is difficult to know the performance before implementation [23]. In FPGA, developers require co-design software and hardware, and understand hardware design concepts [24], and design skills are also important in GPU. For instance, the programmer uses if statement may cause performance poorly [22]. In addition, operation frequency of GPU is faster than FPGA, but the cores are grouped that the data transfer have latency between groups. Furthermore, local memory size in each group is small, and the GPU inappropriately executes sophisticated algorithms with shared shard arrays because the limitations in memory access [25].

However, there are unclear which hardware accelerators are better because it is

doesn’t exist standard benchmark. Many researchers present the performance for diverse applications. In reference [22], they implement Gaussian Elimination, Data encryption standard, and Needleman-Wunsch on an FPGA and the GPU, they find independent data flow. The computation which can process in parallel is good fit to GPU, and applications which have many memory access times are bad fit to GPU. FPGA is good fit at computation low-level bit-wise operations and is bad fit at complex algorithm and data flow. [23] experiments four applications which are matrix multiplication, N-body simulation, Heston Pricing, and finite difference modeling. It really needs to understand different application’s characteristics to select property computing architecture for the best performance and energy efficient. Moreover, [24] chooses specific four benchmarks with different properties, such as memory bandwidth limited (STREAM), computationally limited (SGEMM), memory latency limited (Large FFTs) and parallelism limited (Monte-Carlo Methods), as shown in Fig.1.6. GPUs have great performance for streaming applications because of higher floating-point operation performance and high memory bandwidth, but they have poor performance in Fast Fourier Transform because non-contiguous memory blocks transfers takes higher penalty. FPGA is suited for low memory bandwidth and parallel applications.

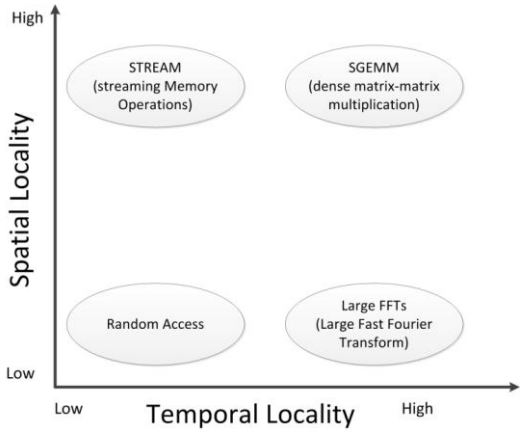


Figure 1.6: Different memory access characteristics benchmark applications [24]

After discussion characteristics in both FPGA and GPU, we survey prior embedded hardware accelerators systems architectures.

In [26], the IBM Power8 processor doubles L1 to L3 data cache size per core for big data analytics. In addition, the execution functional units are also increased to enhance per-core throughput. Since server workloads will continue to evolve, the IBM Power8 processor introduces the coherent accelerator processor interface (CAPI) to support the general purpose cores for a heterogeneous computing solution with off-chip hardware accelerators. These accelerators can be plugged into PCIe slots and implemented in FPGA or ASIC chips. Fig. 1.7 shows the architecture of IBM power8.

In [27], the similar hybrid CPU/FPGA architecture is discussed. Since it is hard to calculate large amount of data by only CPUs, the FPGA can help to enhance the throughput of data processing.

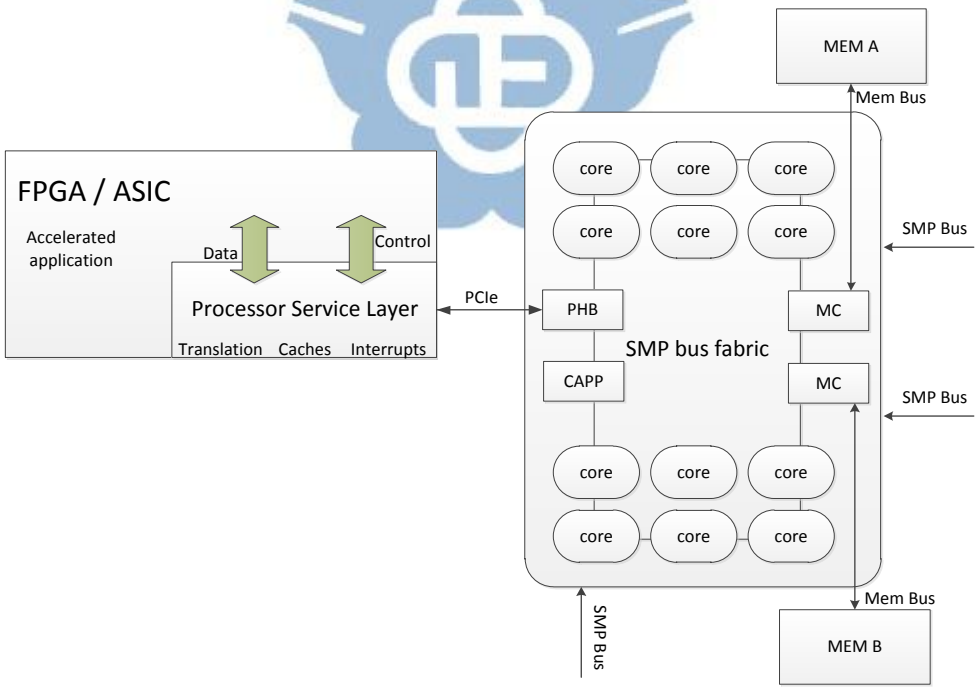


Figure 1.7: Architecture of CAPI [26]

In [28], they implement the three vision services, stabilization, moving target indication and contrast normalization on heterogeneous computing system. The hardware-based vision IP modules are implemented on Xilinx Zynq platforms and service-based software runs on Linux. Software is a middleware that manages hardware accelerator, provides common API functionalities, supports different requirements. Fig. 1.8 shows vision IP hardware architecture. VIN devices accept multiple video formats, and VOUT devices display result on display. VDMA devices can read and write between memory interface and crosspoint switches, and VIP modules are processing in parallel to achieve high performance with low latency. One feature of this architecture is high level of parallelism, and the other is the unified memory that allows multiple devices to share memory with the ARM processor cores.

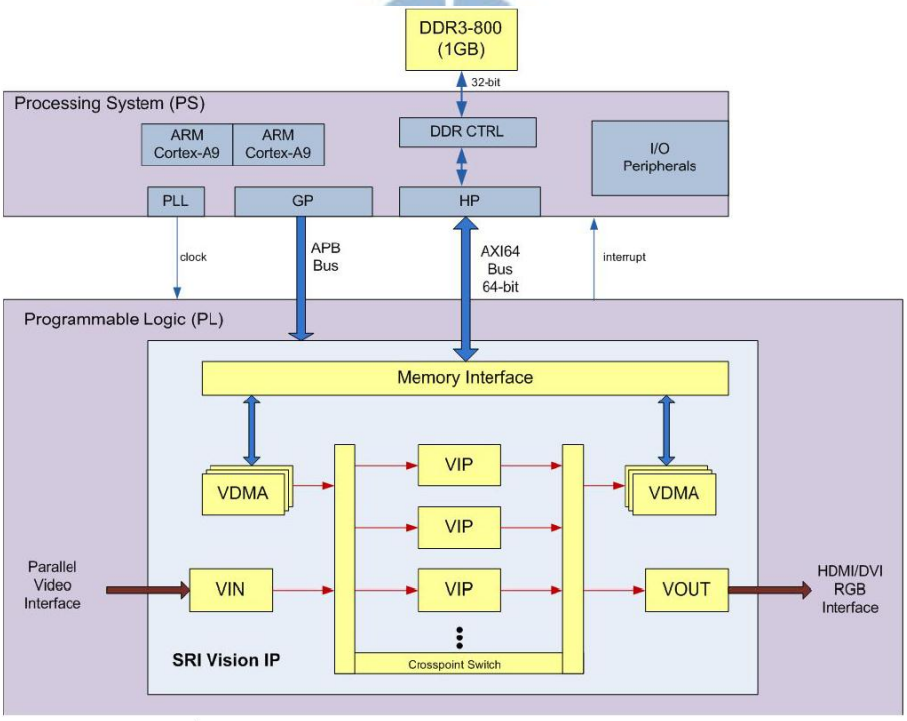


Figure 1.8: Hardware architecture of Vision IP [28]

In [29], they discuss FPGA supercomputer architectures. Fig. 1.9 shows a typical topology. The data are transferred between the host processor and FPGA memory through PCI bus. The advantage of this architecture is that the host operating system

can handle connection protocol, assemble packets, and manages data flows, etc. However, if host CPU needs to perform computation, resources are shared.

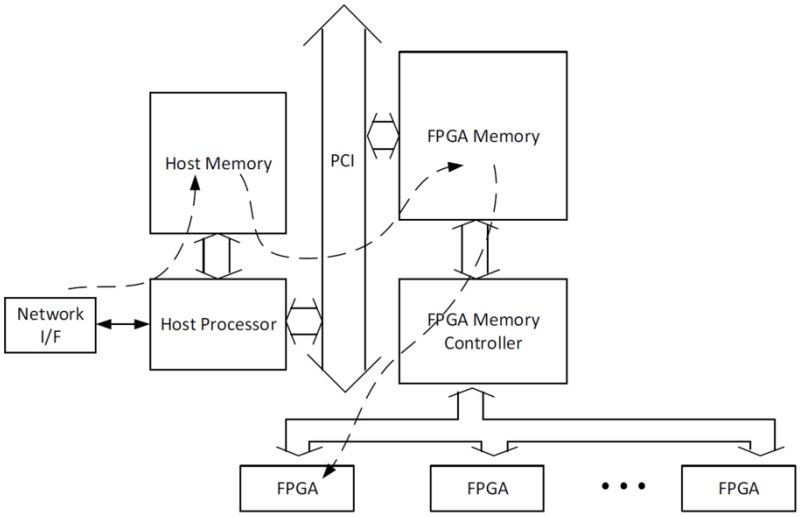


Figure 1.9: Typical FPGA supercomputer data flow [29]

Another architecture is network connections on FPGA side, as shown in Fig. 1.10. The property of this architecture is that the data are directly processed by FPGAs without CPU interfering. However, the disadvantages of this architecture are the costs of computation power and need the master support when data processing in different FPGAs.

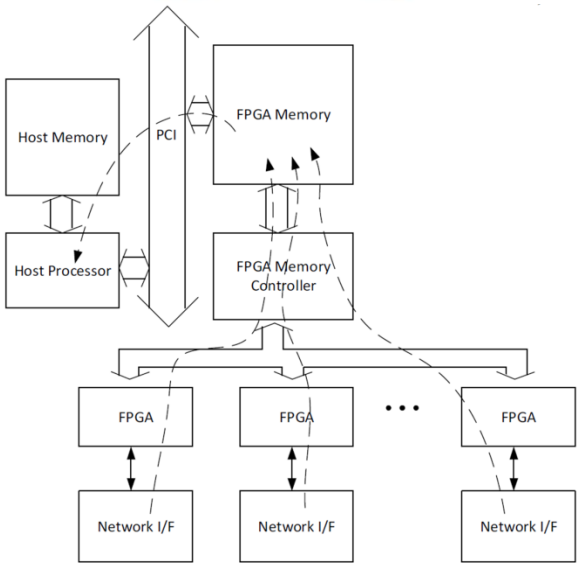


Figure 1.10: Network-oriented FPGA supercomputer data flow [29]

The other architecture in Fig. 1.11 is focused on processing stream data, connecting host processor and FPGA local memory with PCIe. The host processor enables to support managing data flow, and FPGA can process intensive computation. However, this architecture is difficult to be expanded.

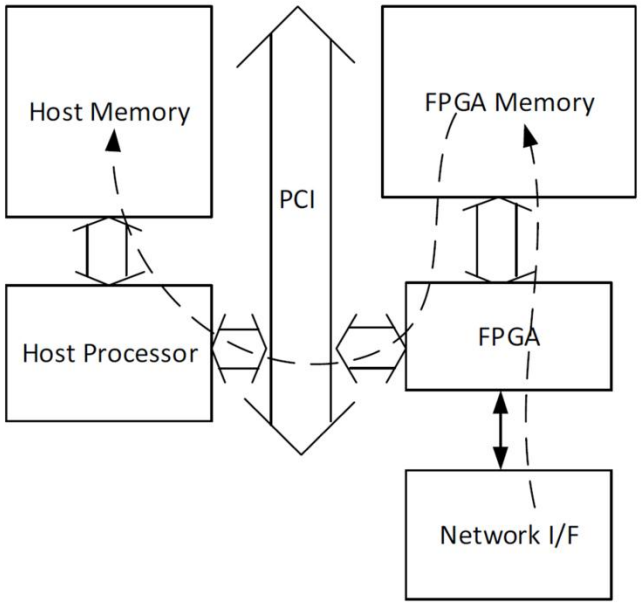


Figure 1.11: Network-oriented FPGA co-processor card data flow [29]

In [30], they integrate ZedBorad platform which combines ARM processor and FPGA, and Hadoop to be a new Zynq-based Hadoop cluster. It can inherit Hadoop frameworks, like the namenode, the scheduler and the HDFS. A CPU system is employed as the namenode that maintains file metadata. This architecture can achieve speedup as compare to pure software approaches.

In this chapter, we disscus hardware accelerator characteristics and heterogenous architectures. Finally, we implement K-means algorithm in Xilinx VC707 platform. K-means algorithm have intesive computation which can work in parallel and can reduce execution time.

1.3 K-means algorithm on FPGA-based Hardware Accelerator

1.3.1 K-means algorithm

Data mining is a technique to find association rule or value in big data. There are many methods, like statistics, on-line analytical processing (OLAP), machine learning, expert system, pattern recognition, etc. K-means clustering algorithm is one of the clustering algorithms for data analysis, and K-means clustering algorithm has intensive computation and non-sequential working. These properties are suited for hardware accelerators.

K-means algorithm is used to image processing, cluster analysis, and feature learning. The goal of K-means algorithm is to separating the input data into the number *k* of clusters. The data which are in a cluster have similar properties to each other and be dissimilar in other clusters. Assume input data is on a set X of D-dimensional real vector, that $X = \{x_n \epsilon R^D, with\ n = 1, \dots, N\}$, and partition X into K ($K \leq N$) sets $S = \{S_1, S_2, \dots, S_k\}$, each cluster is associated with center value. Therefore, the output of objective function in Euclidean distance is $\sum_{i=0}^{k} \sum_{x \in S_i} ||x - u_i||^2$, where u_i is the mean of points (center) in S_i. K-means algorithm is iterative procedures, there are four steps as followed in Fig. 1.12.

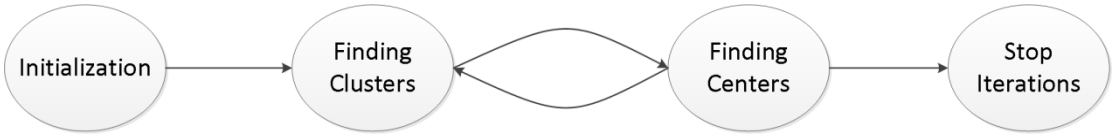


Figure 1.12: K-means algorithm steps flow

First, the initialization step selects initial value to be center and determines the number of clusters. Basically, the programmer sometimes uses random numbers to be centers, but initial values are effected the result of accuracy or iteration times. Many algorithms are used to detect the initial value to improve performance, but we do not discuss in here. Second, the finding clusters step assigns objects to the nearest cluster center. We can get the number of *K* clusters in this step. Third, in finding centers step calculates each cluster center as the mean of objects in the cluster. If the all of new cluster centers are approximate same then K-means is finished. Otherwise, the step returns to finding clusters step with new centers to find new cluster sets.

We list the advantages and the disadvantages of K-means as followed:

Advantages of K-means:

- K-means is easier to understand and is faster than hierarchical clustering.
- K-means has relatively efficient: O(nkdt), where n are the number of nodes, k are the number of clusters, d is dimension of node, t are iteration times.

Disadvantages of K-means:

- Different initial centers cause different result that is hard to comparing quality.
- K-means is unable to resolve if datasets have two tightly or overlapping data.
- K-means is unable to handle non-linear datasets or outliers.

1.3.2 Related Work

K-means algorithm is used to many diverse domains, so we focus on hardware implementation.

In [31], they implement K-means algorithm for image processing on PCI board with FPGA, and connect external two memories. One memory is storing the pixel data from the host computer and the other memory is saving the cluster results. They use the FPGA to find each pixel cluster numbers, and store sum of pixel value in accumulators. In the host computer side, they send complete image to the FPGA memories, and compute new cluster centers. In [32], they modify architecture of [31]. They add floating-point division module in FPGA. This architecture computes new cluster centers on the FPGA side, and saves each iteration result in the FPGA before algorithm is terminated. They use PCIe communication between host and FPGA, Fig. 1.13 is shown the data flow in [32]. First, the host computer sends complete image data to the FPGA memory. Second, initial centers are transmitted to the FPGA from the host. Before K-means algorithm is finished, there is no contact between the host and the FPGA, it saves a lot of transmission time comparing with [31]. Fig. 1.14 shows the structure of K-means in [32]. Inputs of this architecture are image data and cluster centers. Outputs of this architecture are the cluster assignment for each pixel. Both the pixel shift module and validity module are used to synchronize data with computation. The datapath module is found the nearest cluster of pixel data. The accumulators are counted the number of data in each cluster, and are accumulated pixel data associate with clusters. The mean updata division module is calculated new cluster centers. Finally, all cluster centers are replaced by new cluster centers in next iteration.

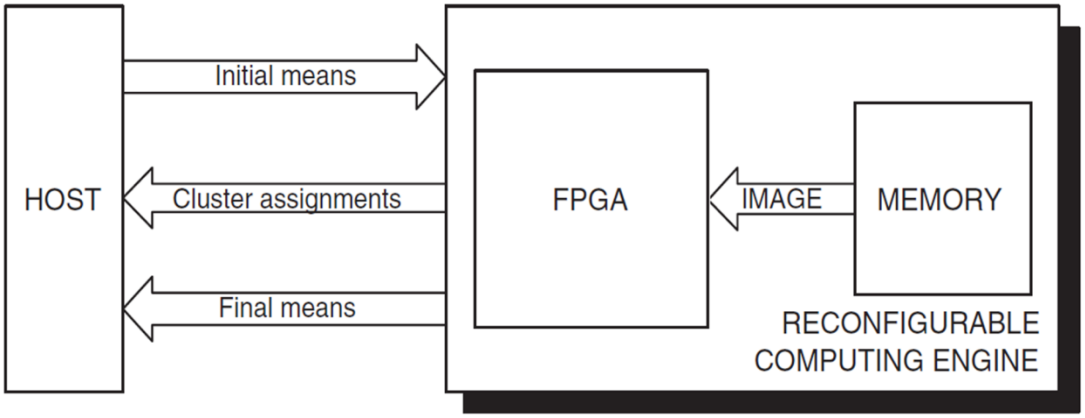


Figure 1.13: Interaction between the host and FPGA [32]

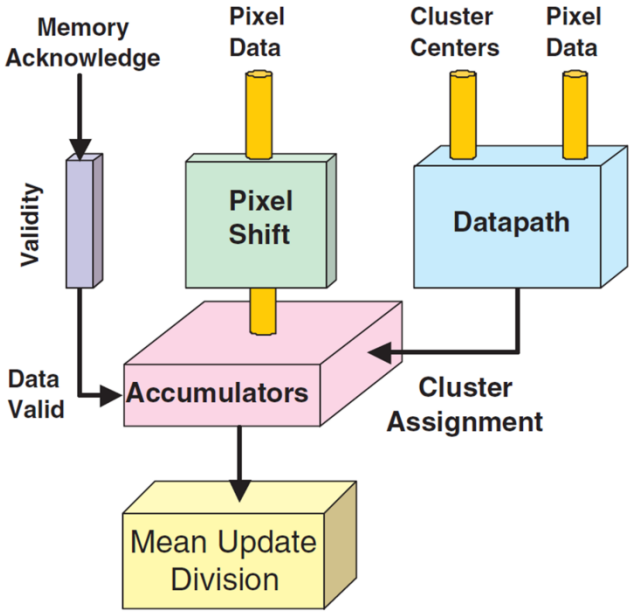


Figure 1.14: K-means algorithm circuits [32]

The execution time is speedup because programming in parallelism and reducing transmission times. Nevertheless, the memory capacity is obstacle, volume of big data or others dataset may larger than size of memory.

In [33], they implement K-means cluster for color image on FPGA with Euclidean distance metric. The hardware circuits are shown in Fig. 1.15. A target pixel is 24-bit full color RGB images. Each SRAM bank is 32-bit, four pixels are stored in three memory banks and results are stored in reset memory banks. This architecture is

computed 96 squared Euclidean distances in parallel and it is calculated new center from four partial sums. Furthermore, they implement filter algorithm that each pixel is less than or equal to 24, and FEKM algorithm [36] can reduce scanning the number of pixels for iteration. They use two techniques to reducing computation time, and the memories can load next image when others image is processing. However, the number of computing pixels for once operation is four that is not enough.

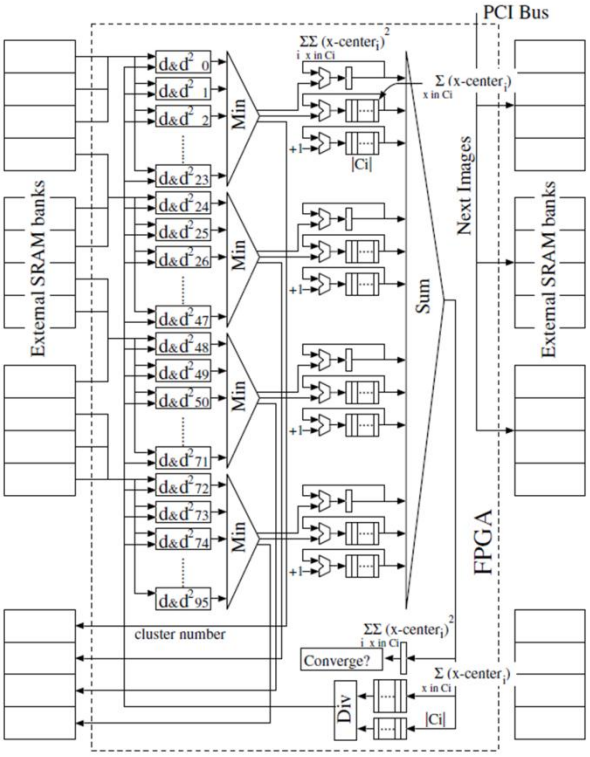


Figure 1.15: K-means hardware circuit [33]

In [34], they implement K-means in FPGA for high dimensional datasets. They use triangle inequality algorithm to avoid calculating square root to reduce consumption, and the result is hardly changed. Fig. 1.16 shows data flow of this architecture. First, data are stored in memory through PCIe and hardware accelerator do clustering. After clustering is completed, the result is send to host through JTAG. Fig. 1.17 presents hardware architecture. The high-dimensional data FIFO is

controlled input data. The distance calculators are calculated square distance in parallel. The distance accumulator is accumulated distance, and it is found the minimum square distance and cluster after the calculation is completed in this iteration. The counterparts in processing units are used to compute the new distance and cluster to compare to old ones. Finally, both upper bound and lower bound are used to store square distance for triangle inequality algorithm.

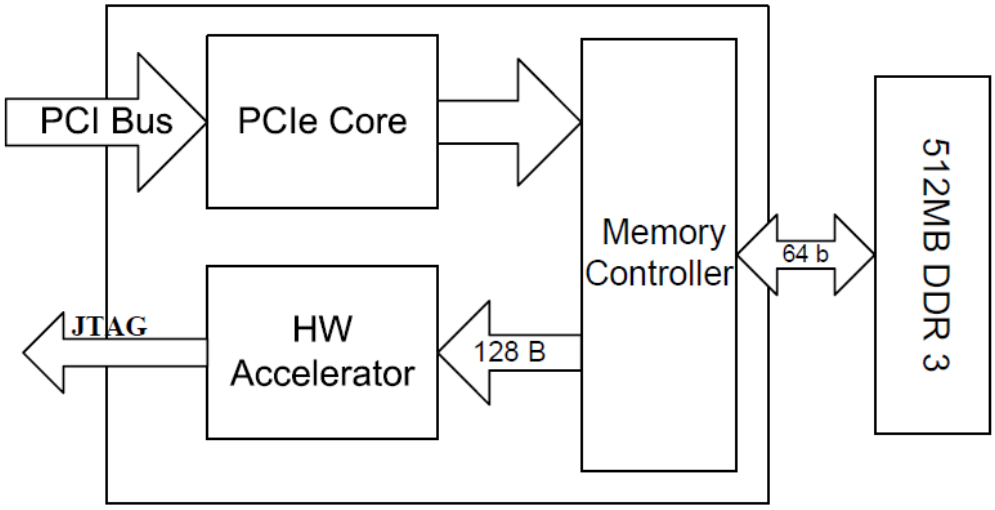


Figure 1.16: Data flow overview [34]

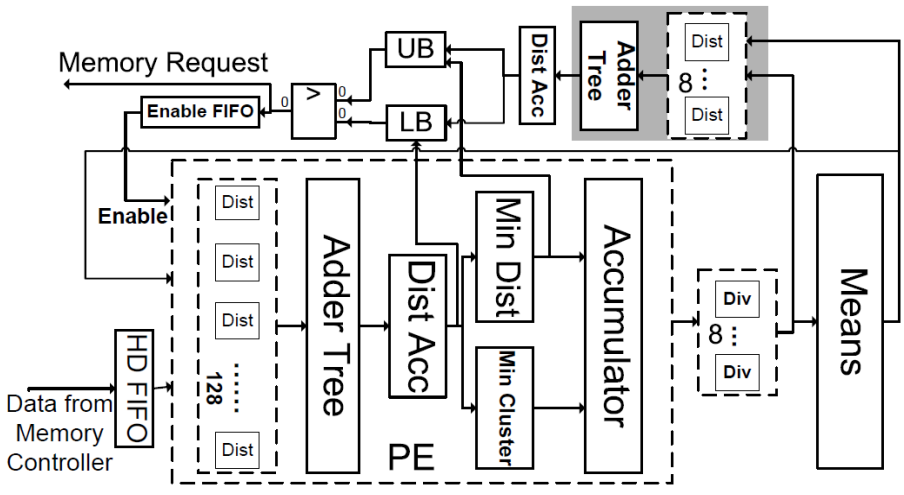


Figure 1.17: Hardware architecture [34]

In [35], they implement K-means in hardware for microarray data, and computation distance and find the minimum distance in parallel. However, all of input data is stored in on-chip memory, that is not work where the dataset is larger than capacity of memory.

1.4 Motivation

Big data analytics requires to analyzing data at the rate that matches the speed of data production. Therefore, some software frameworks such as Hadoop with high scalability and fault tolerance had been proposed to enable massive data storage and processing over large clusters of computing servers. However, the performance of data analytics can be further improved by deploying hardware accelerators to the computing servers.

Data mining algorithms are used to analyze big data. K-means clustering algorithm is one of data mining techniques, and it is simple but powerful. Moreover, K-means clustering algorithm is a data-intensive computation application that hardware accelerators are suited for speed up computation. In addition, several heterogeneous architectures interact with hardware accelerators and host computer through PCIe slots. The transmission rate of PCIe is higher than Ethernet. However, limitations of PCIe slots are hard to be expanded. Moreover, capacity of memory is not large enough to store whole datasets, that is hardly to complete within independent single hardware accelerator.

Therefore, we develop FPGA-based hardware accelerator through Ethernet switch to achieve high scalability. We implement K-means clustering in our FPGA-based hardware accelerators, and cooperate with the host computer and FPGAs, the host computer is employed as the master to manage data and control the status, the FPGAs focus on computation.

The rest of the paper is organized as follows: Chapter 2 introduces our FPGA platform, architecture overview, packet format format between transmission and reception, and communication memory with memory interface. Chapter 3 depicts

implementing K-means clustering algorithm in our FPGA and achieving scalability. Chapter 4 shows our experiment environment and speeds up comparing with Hadoop framework. Finally, in Chapter 5, we make a conclusion and discuss the future works about embedded hardware accelerator issues.

Chapter 2

Proposed Hardware Accelerator Architecture

2.1 System Overview

The proposed accelerator platform is composed of many VC707 FPGA evaluation boards (EVBs) [37], as shown in Fig. 2.1. The computing server communicates with FPGA EVBs with Gigabit Ethernet switch. Then, the workloads of the computing server can be shared in FPGA EVBs. The computing server wraps the data into packets and sends them to VC707 EVBs through a Gigabit Ethernet switch. Then, the workloads of the computing server can be shared in FPGA EVBs.

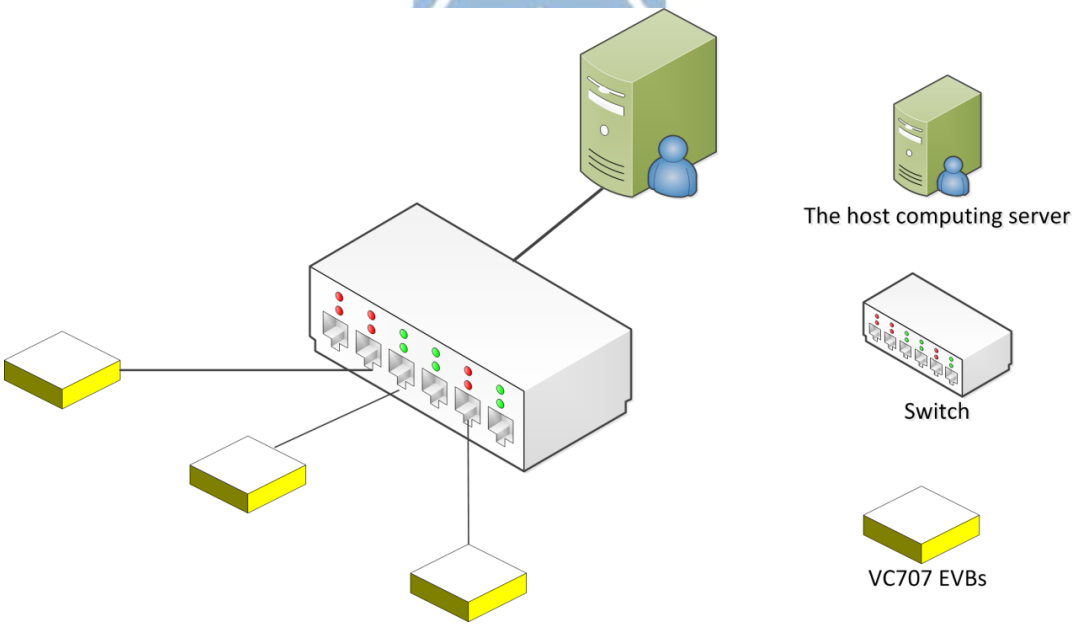


Figure 2.1: The proposed FPGA-based hardware accelerator platform with VC707 evaluation boards

The component locations in VC707 EVB are shown in Fig. 2.2. We describe the components which are used in our implementation in Table 2.1.

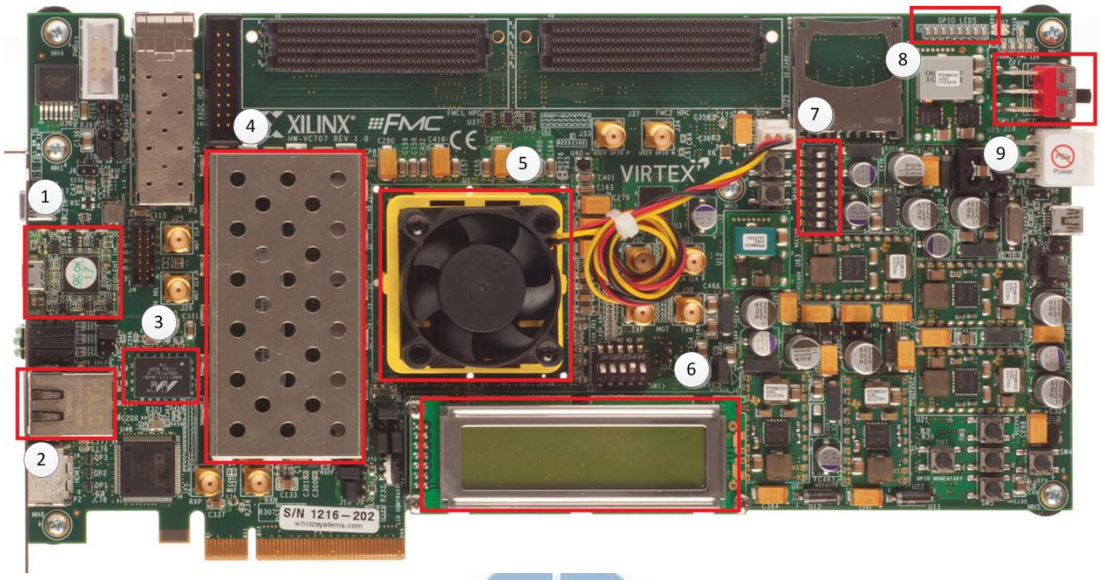


Figure 2.2: VCV707 EVB [37]

Table 2.1: VC707 EVB component description

Locations	Component Description
1	USB JTAG interface
2	Network cable port
3	10/100/1000 Mb/s Ethernet PHY
4	DDR3 SODIMM memory (1 GB)
5	Virtex-7 FPGA with cooling fan
6	LCD character display
7	User DIP Switch
8	User LEDs
9	Power on/off switch

We use ISE tool to design the VC707 EVB. Fig 2.3 is shown the ISE design flow. Our coding language is Verilog, and the vendor provides IPs to help developers to complete complex algorithm. Developers can embed IPs in design by inserting template of IPs. The user constraint file (.ucf) can set constraints and pin locations. After synthesis and implementation in design, the ISE operators reports and messages, and creates bit file. Finally, the impact tool is used to program bit file into VC707 EVM board through JTAG. Moreover, LCD character display or LEDs can show debug information. However, we use ChipScope to record the trace of signal and help us for debuging. The debug tool ChipScope can monitor registers value when VC707 EVB is executed, we will introduce it in detail in section 2.2.4.

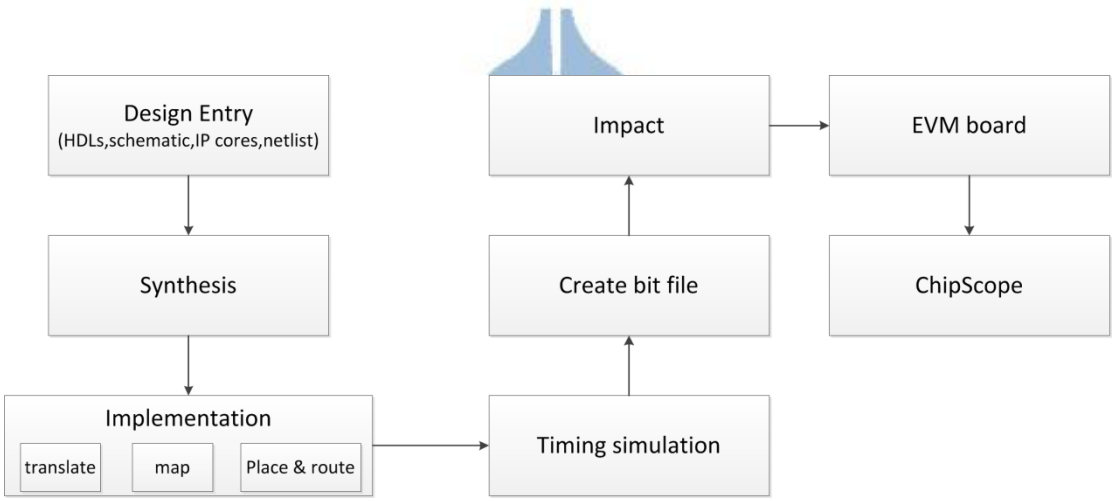


Figure 2.3: ISE design flow

2.2 Hardware Accelerator Development

2.2.1 Architecture of VC707 EVB

We implement four modules in the FPGA including Ethernet physical IP, mixed-mode clock manager (MMCM), DDR3 memory interface controller IP, and user core. Ethernet physical layer IP is used to collect the packets sent from the computing server, and the computation results of the FPGA can be also sent back to the computing server through the Ethernet physical layer IP. The MMCM creates dynamic reconfiguration of the phase, duty cycle, and clock output frequency. The DDR3 memory interface controller IP helps the user core to communicate with the on-board 1GB DDR3 memory. Finally, the proposed computation algorithm or controller are implemented in the user core of user design.

The VC707 EVB clock distribution is shown in Fig. 2.4. The 200MHz system clock (sys_clk) is consisted of a differential clock pair SYSCLK_P and SYSCLK_N. The 125MHz SGMII clock is composed of SGMIICLK_Q0_P and SGMIICLK_Q0_N differential clock pairs. The reference clock (clkin1) of the mixed-mode clock manager (MMCM) is 125MHz. The MMCM provides a phase and frequency related the 62.5MHz clock (clkout0) and the 125MHz clock (clkout1), and the clock (clkfbout) is used to reduce skew. The clock (clkout0) is used for Ethernet controller module, and the clock (clkout1) is used for physical medium attachment (PMA) IP module and serial-gigabit media independent interface (SGMII) module, inside the Ethernet physical IP. In memory interface IP, the clock (clkout1) is for the memory controller, and the output clock (app_clk) provides clock to communicate with memory controller. Finally, the clock (userclk) is used to user core.

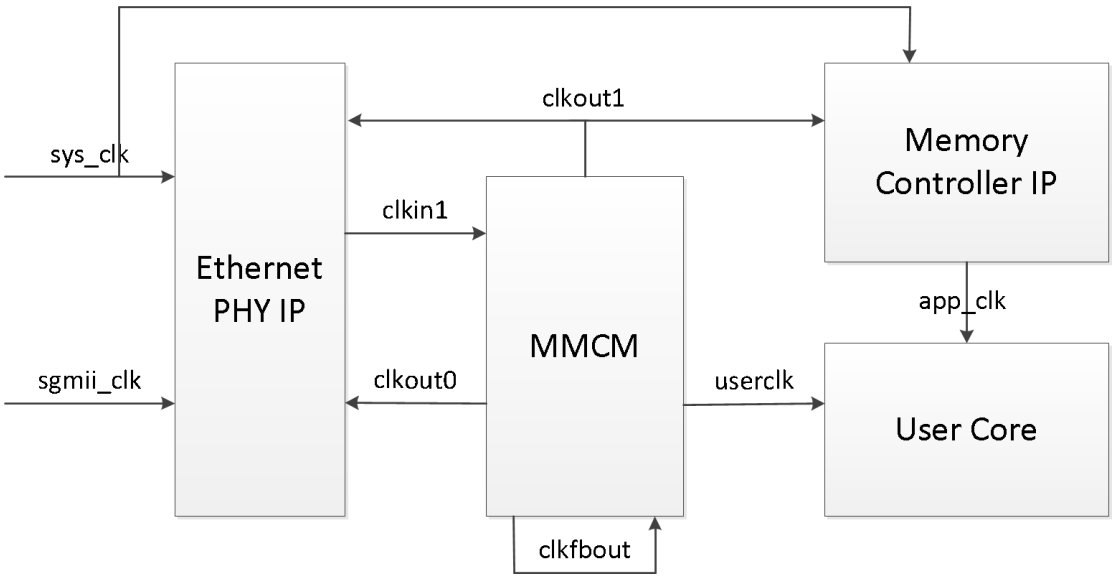


Figure 2.4: VC707 EVB clock distribution

The data flow is shown in Fig. 2.5, Ethernet physical IP receives data rx_d from the host computer and transmits data tx_d to the host computer. The RX FIFO receives data (rx_d) from the Ethernet physical layer IP and combines them. The TX FIFO sends the byte data (tx_d) split from the packet to the Ethernet physical layer IP. Both the RX FIFO and the FX FIFO behavior will describe in section 2.2.2. The commands (cmd) and addresses (addr) can be sent to the memory interface controller IP simultaneously. For a write request, the written data (wm_data) should be prepared before sending write command to the memory interface controller IP. After a write request is finished, the ready signal provided by the memory interface controller IP indicates the completion of the write request to the DDR3 memory. For a read request, after several cycles, the read data (rm_data) are output by the memory interface controller IP. We will describe memory operation in section 2.2.3.

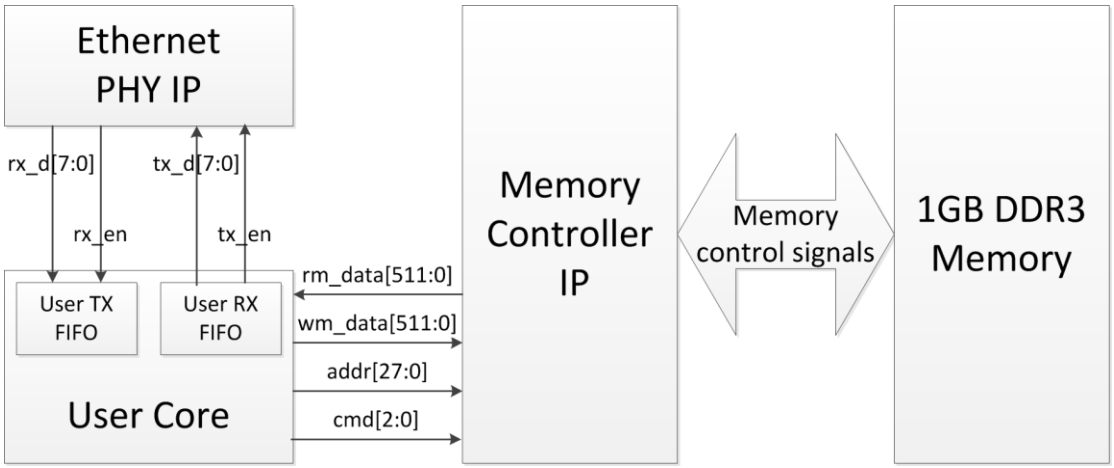


Figure 2.5: Data flow of VC707 EVB

2.2.2 Ethernet PHY Controller

The Ethernet PHY IP module is helped us to handle complex Ethernet protocol, we focus on Ethernet packet format when we process packet. The packet format is shown in Fig. 2.6.

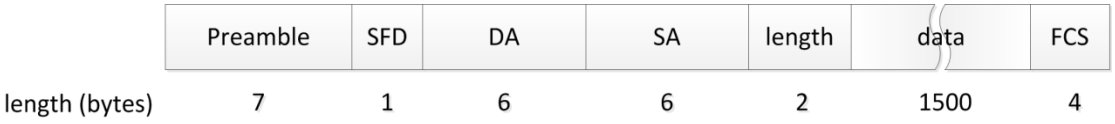


Figure 2.6: Packet format

- Preamble: The Preamble consists of 7-byte pattern of altering 1 and 0 bits, and is used to synchronize the receiver clock rate.
- Start frame delimiter (SFD): The SFD is a one byte which marks the beginning of Ethernet frame.
- Destination address (DA): The DA contains target MAC address.
- Source address (SA): The SA contains source MAC address.
- Length: Length is data length. However, data length must be longer than 64-byte without preamble, the reason is avoiding packet collision. Therefore, minimum data length is 46 bytes and the maximum data length is 1500 bytes.
- Frame check sequence (FCS): The FCS contains 4 bytes that allows check corrupt data in the data entire frame at receiver.

The RX FIFO module operations are shown in Fig. 2.7. We use a counter to combine receiving data (rx_d) when the reception signal (rx_en) is enabled (rx_en). The RX FIFO module outputs the receiving data to the user core module after checking the destination address (DA) and the source address (SA) are correct. Finally,

we assert the signal (rx_finish) when reception is done.

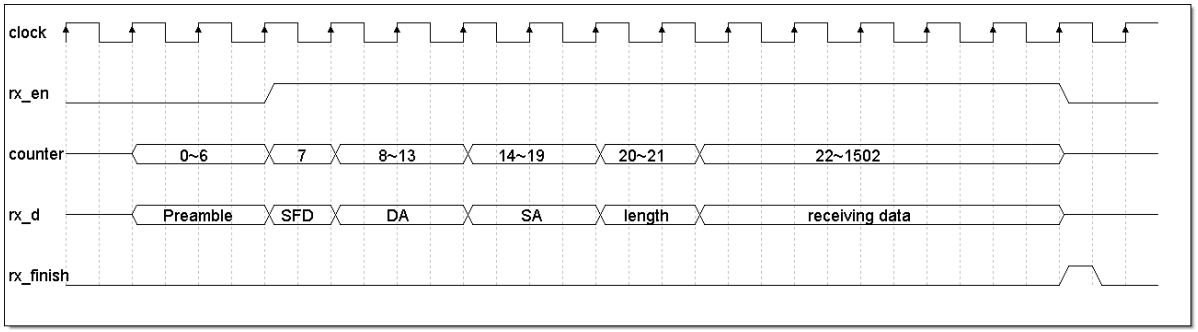


Figure 2.7: Behavior in RX FIFO module

The TX FIFO module behavior is similar to the RX FIFO module. Fig. 2.8 shows the TX FIFO operations. We assert the transmission signal (tx_en) after user core operation is finished. Subsequently, we wrap data into packets when the signal (tx_en) is high. Finally, the packets are sent to the computing server through Ethernet physical IP.

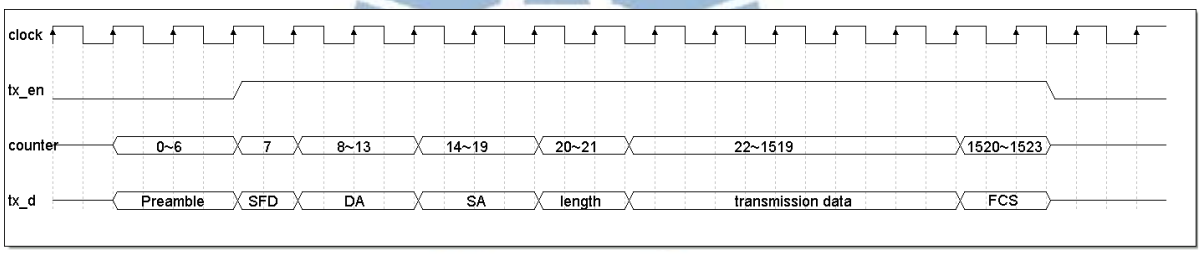


Figure 2.8: Behavior in TX FIFO module

2.2.3 Memory Operation

The DDR3 memory controller IP helps to communicate with DDR3 devices. Table 2.2 shows the DDR3 user interface (UI) primary signals by user core side.

Table 2.2: DDR3 user interface primary signals

Signal	Direction	Description
Command		
app_rdy	Input	This input indicates that UI is ready to accept requests.
app_en	Output	This output asserts when user sends app_addr and app_cmd to UI.
app_addr [27:0]	Output	This output is the address for current command.
app_cmd [2:0]	Output	This output indicates command for current requests. Read = 001, Write = 000
Write		
app_wdf_rdy	Input	This input indicates that write FIFO data are ready to receive data.
app_wdf_data [511:0]	Output	This output contains data that are ready to write into memory.
app_wdf_wren	Output	This output indicates that app_wdf_data are valid.
Read		
app_rd_data_valid	Input	This input indicates that the read data is valid.
app_rd_data [511:0]	Input	This input contains the read data from memory.

The write operation timing diagram is shown in Fig. 2.9. The command (app_cmd) and the address (app_addr) are sent simultaneously when the signal (app_en) is asserted. In addition, we confirm the signal (app_rdy) is high that UI is accepted the requests, and signal (app_wdf_rdy) is high that write FIFO is ready to store data. Subsequently, Fig. 2.9 is shown three scenarios for the write data in memory, as follows:

Case 1: Write data (app_wdf_data) are sent along with the command.

Case 2: Write data are sent before command.

Case 3: Write data are sent after command, but the maximum delay is two clock cycles.

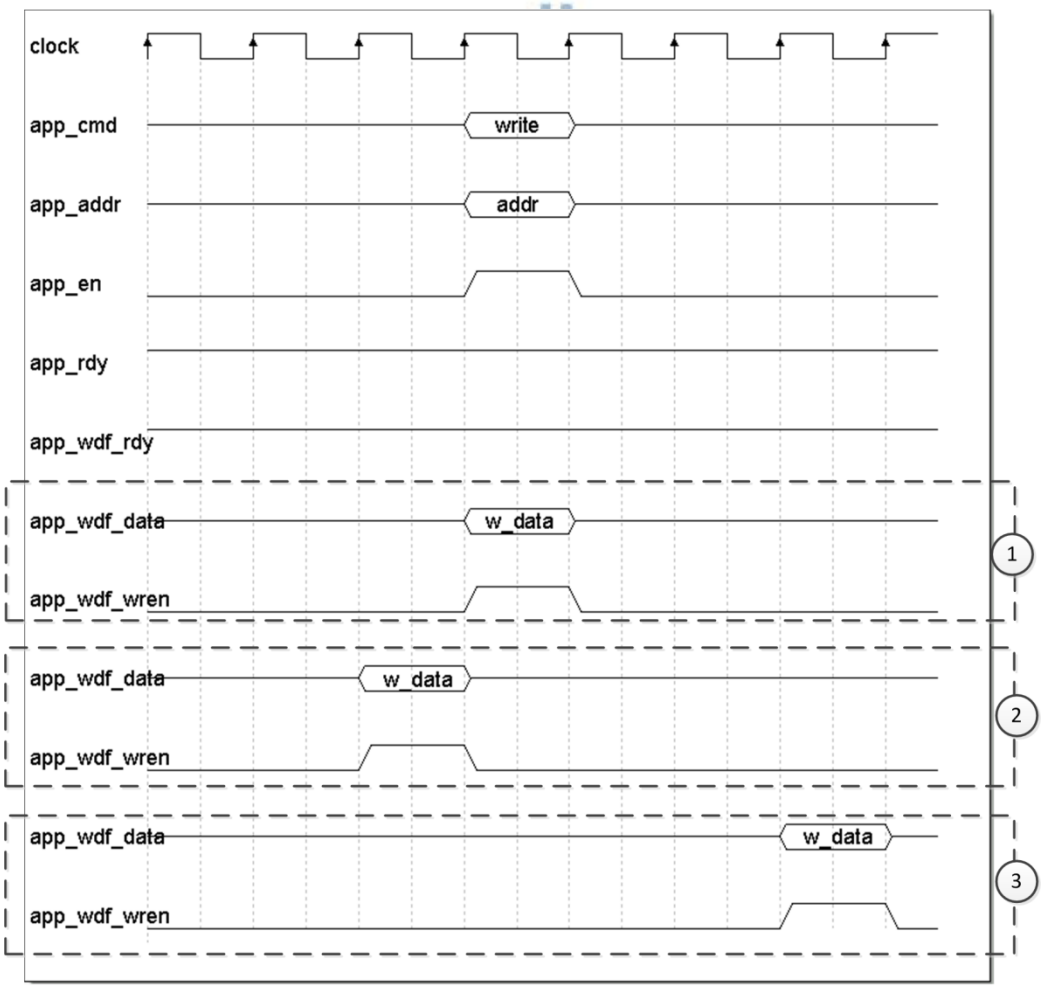


Figure 2.9: Write operation timing diagram

Moreover, the sequential write operation is little different, there is no delay between the write data and command. The write data are sent along with corresponding command. If signal (app_wdf_rdy) is deasserted, we need to hold signal (app_wdf_wren) high and keep the write data (app_wdf_data) value until signal (app_wdf_rdy) becomes high. Fig. 2.10 shows the burst write operation timing diagram.

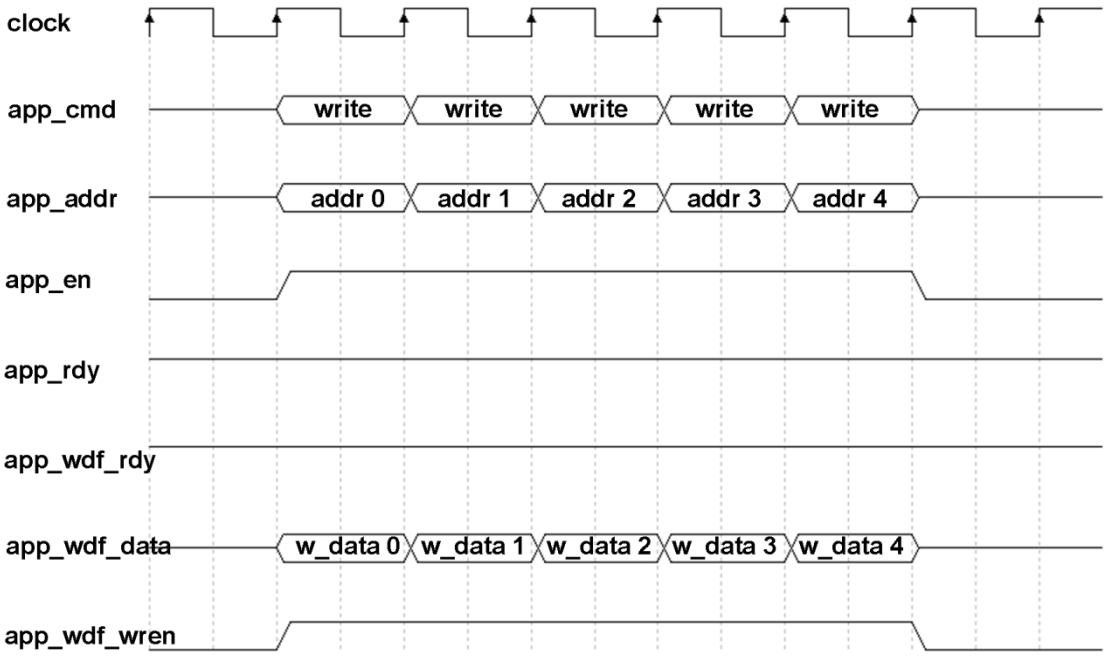


Figure 2.10: Burst write operation timing diagram

The read operation is similar. When signal (app_en) is asserted, we sent read quest command and address. The read data (app_rd_data) return by the UI when signal (app_rd_data_valid) is asserted.

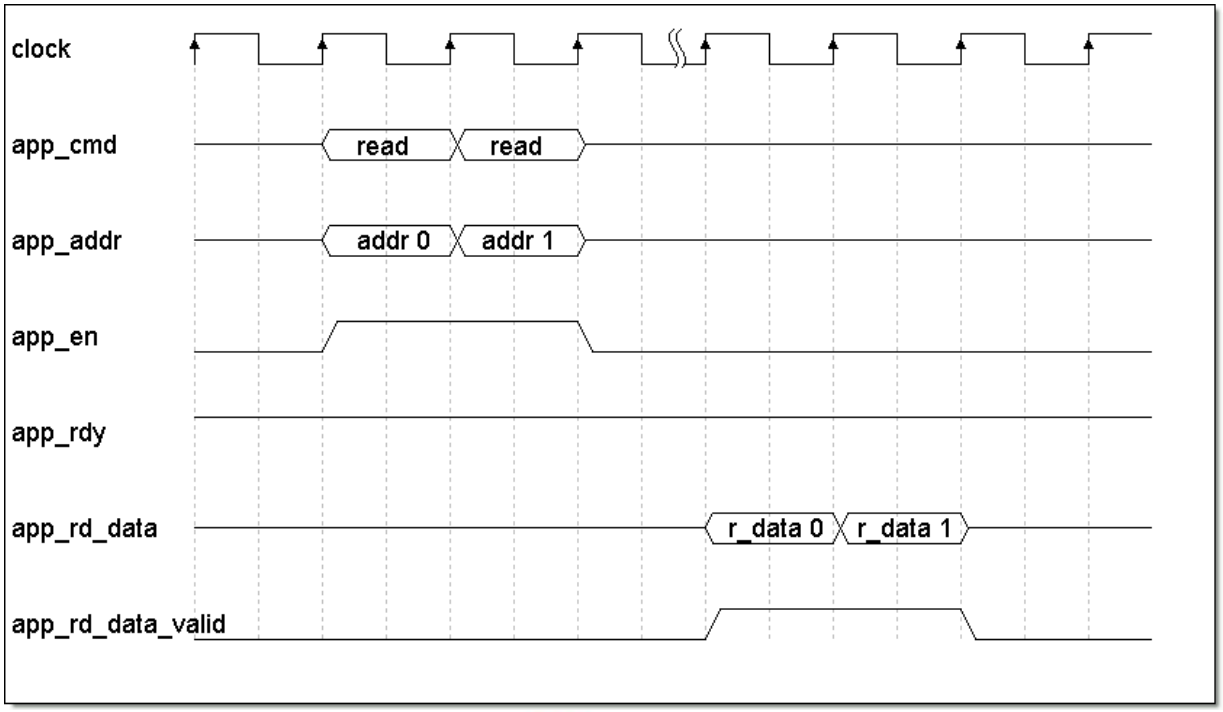


Figure 2.11: Read operation timing diagram

2.2.4 ChipScope Debug Tool

User can run simulation by ISE. However, ISE simulation is unable to show the real time operation of the FPGA. Therefore, we use ChipScope to debug. The ChipScope tools integrate measurement hardware components with target design. We insert Integrated CONtroller cores (ICONs) and Integrated Logic Analyzers (ILAs) in our design. The ICON module communicates between the ILA module and host computer through JTAG, and the maximum number of connect signal with the ILA is fifteen. The ILA module can monitor internal signals in user design, and it sets trigger conditions to the stored signal. Finally, we use ChipScope software in host computer to check signals which are stored in ILA module. Fig. 2.12 shows the ChipScope system diagram.

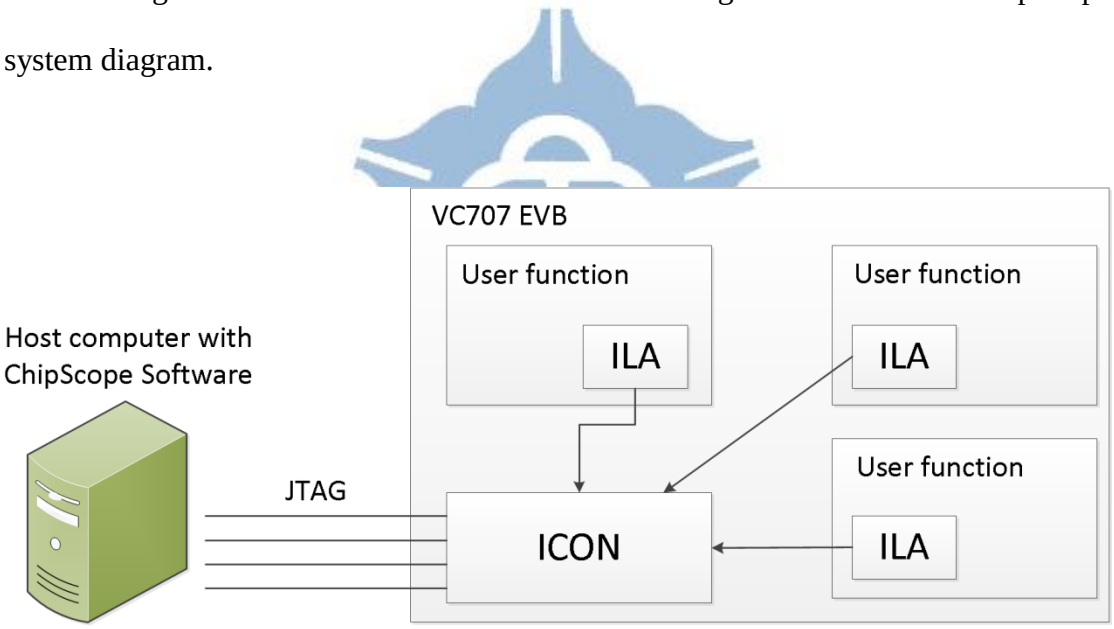


Figure 2.12: ChipScope system diagram

Chapter 3

K-means Clustering Implemented in FPGA-based Hardware Accelerator

3.1 System Architecture for Implementation K-means

We describe K-means clustering algorithm in section 1.3.1. K-means clustering is grouped objects based on unique features into the number of clusters. In addition, K-means is applied to diverse applications, and we implement K-means for dataset analysis. Because K-means is unable to handle non-linear datasets or outliers, we use Matlab to create dataset which is distinct from each cluster. Moreover, every node in dataset is three-dimensional, and thus, we can display the result to check the cluster distribution.

The design flow is shown in Fig. 3.1. First, we create the dataset which contains distinct clusters with random three-dimensional nodes, and dataset format is a text file. Second, the master processor reads dataset into arrays and wraps nodes, cluster centers, control signals into transmission packets. Subsequently, we send packets to VC707 EVBs through Ethernet switch. Third, the VC707 EVBs groups node into the clusters after receiving node data and centers. After that, the VC707 EVBs calculate the cluster number of each node. Consequently, the VC707 EVBs return the cluster

value of nodes to the host computer. Fourth, the master processor is calculated all cluster centers according to receiving the cluster number of nodes from the VC707 EVBs. Step 5, the master processor replaces initial centers with the new centers, and next iteration is processed to find the new cluster centers from step 2 to step 4. Finally, we write the result into the text file, and display the clusters distribution by Matlab, when the iteration times are terminated.

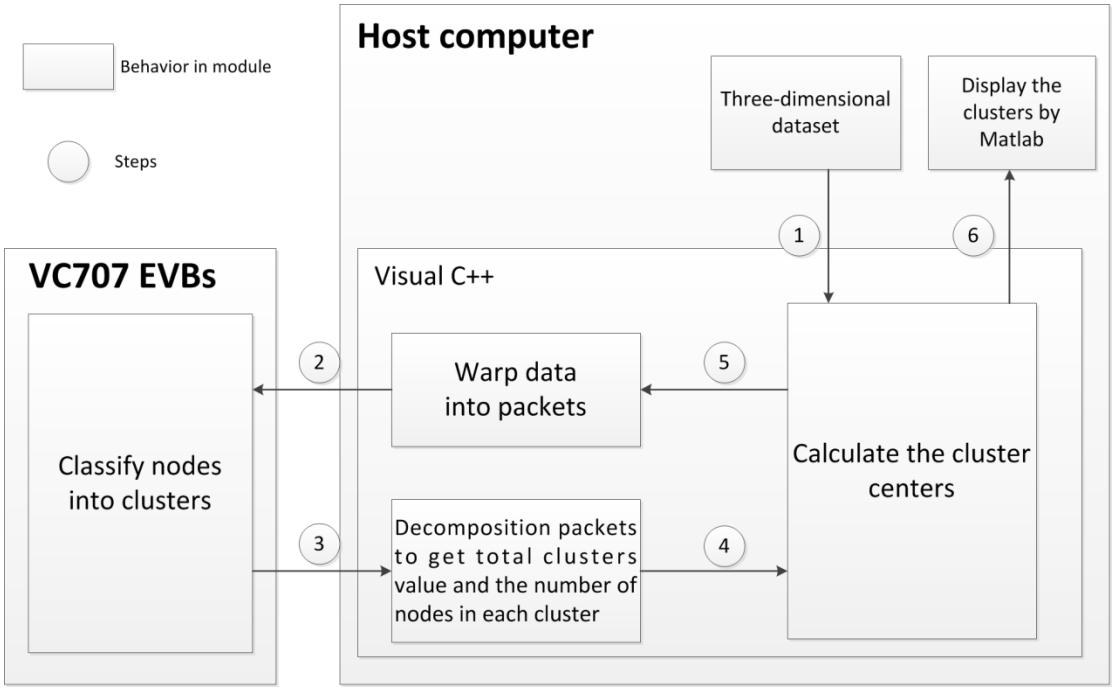


Figure 3.1: System of K-means clustering implementation design flow

3.2 Circuits Design in FPGA

We share the workloads in the three VC707 EVBs, and each VC707 EVB has same components and behaviors. Therefore, we introduce the architecture of single VC707 EVB. We do not use the on-board DDR3 memory in the design, because the size of dataset is larger than capacity of memory. Furthermore, the VC707 EVB memory communication time could cause overload. Besides, the VC707 EVB has two primary modules Ethernet PHY IP and clustering circuits, as shown in Fig. 3.2. In addition, the single VC707 EVB can handle 115 data nodes from the host computer transmission packet simultaneously. After the dataset is complete transmission, the VC707 EVB returns the sum of the clusters coordinate values with three dimensions and the number of nodes in each cluster to the host computer. Finally, the host computer can calculate new centers from the three VC707 EVBs and find the new centers to the next iteration.

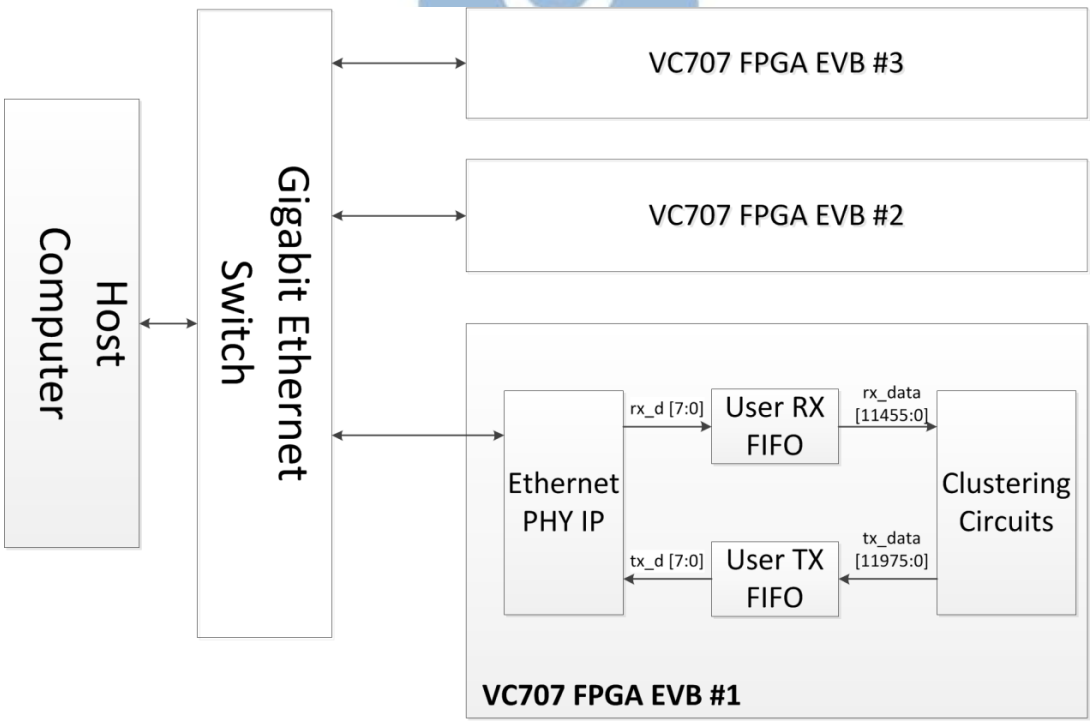


Figure 3.2: K-means implemented module of VC707 EVB

Algorithm1 : K-means clustering algorithm in VC707 EVB

Input: *3-d 115 floating-point nodes (32 bits*3*115=11040 bits) and 4 cluster floating-point centers (32 bits*4*3=384bits)*

Output: *the registers cluster_value[3][2] store clusters coordinate value of 3-d floating-point (32 bits*4*3=384 bits) and the registers cluster_amount[3] store number of nodes in each cluster (96 bits*4=384 bits)*

max_number_of_cluster is 4

max_node_count is 115

```
receive packet input from host computer;
store input in the node data registers and center data registers in sequential;
for(cluster_count=1; cluster_count <= max_number_of_cluster; cluster_count++) begin
        set_cluster_center(cluster_count);
        calculate Euclidean distance to cluster centers in total 115 nodes;
        save 115 Euclidean distances from 115 nodes;
end
In 115 nodes with 4 Euclidean distance, find the shortest distance from 115 nodes;
for(node_count=1; node_count <= max_node_count; node_count++) begin
        if(cluster number of node == cluster number "n" of node with shortest distance ) then
                cluster_value[n][0] = cluster_value[n][0] + node_x_coordinate;
                cluster_value[n][1] = cluster_value[n][1] + node_y_coordinate;
                cluster_value[n][2] = cluster_value[n][2] + node_z_coordinate;
                cluster_amount[n] = cluster_amount[n] + 1;
end
if dataset is complete transmission then
        wrap the registers cluster_value[3][2] and the registers cluster_amount[3] into packet and send to host
        computer;
        reset the registers cluster_value[3][2] and the registers cluster_amount[3] value for next iteration;
else
        wait for packet from host;
end
```

Algorithm 1 shows the pseudo code of the K-means clustering algorithm in the VC707 EVB. In K-means algorithm, the inputs are three-dimensional 115

floating-point nodes and 4 cluster floating-point centers from host computer. We store the node data and cluster centers when the VC707 EVB is received packet. Subsequently, we calculate the Euclidean distance to each cluster centers in total 115 nodes. Consequently, we find the shortest distance in 4 Euclidean distance of each cluster. After that, we accumulate the grouped nodes of one cluster coordinate floating-point values that are x-coordinate, y-coordinate and z-coordinate in three 32-bit registers until 115 nodes are computed. Moreover, we store the number of nodes in one cluster into 96-bit register. Our maximum number of cluster is four, so we have twelve 32-bit registers to store the clusters coordinate values of floating-point, and four 96-bit registers to store the number of nodes in each cluster. When the dataset is complete transmission, the VC707 EVB returns the sum of the clusters coordinate values with three dimensions and the number of nodes in each cluster to the host computer. In addition, we do not compute new centers in the FPGA because we share the workloads to the three VC707 EVBs. We need to compute the new centers in the host computer.

3.2.1 Data Field in Packet

The packet format is described in section 2.2.2. This section shows the value of data field in both transmission and reception packets, as shown in Fig. 3.3 and Fig.3.4.

The reception packet data field (rx_data) starts following packet number (rx_cnt) which indicates packet number. The command (rx_cmd) is used to require FPGA to execution code according to command. The amount of cluster indicates the number of clusters, and maximum the number of clusters is four in our design. The amount of node is used to show the number of nodes in packets. The cluster center contains four cluster centers, and each center has three data because of dataset is three-dimensional. In addition, node value is single-precision (32 bits). Therefore, we use 48-byte register to save four cluster centers. Finally, the node data consisted of 115 nodes value, and each node has three-data.

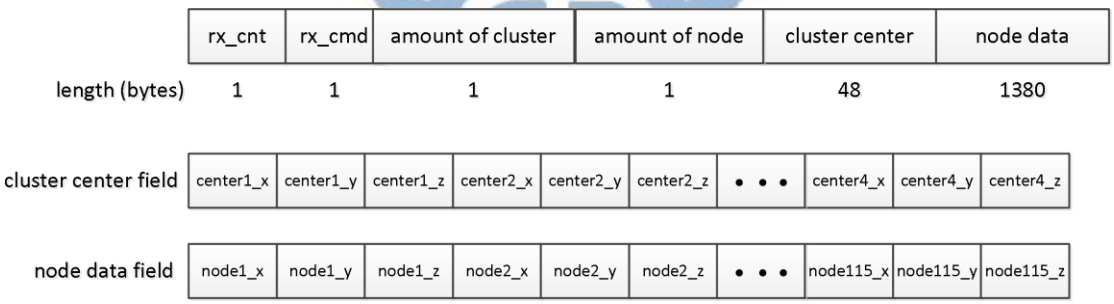


Figure 3.3: Reception packet data field form host computer

The transmission packet data field (tx_data) is composed of packet number (tx_cnt), command (tx_cmd) and the node data. The packter number (tx_cnt) is same value as reception packet to confirm the packet number. The command (tx_cmd) is determined the transmission data which is transmitted to host computer, and the command (tx_cmd) also has same value as reception packet. The last, the node data register contains the sum of coordinate values and the number of nodes in each cluster.

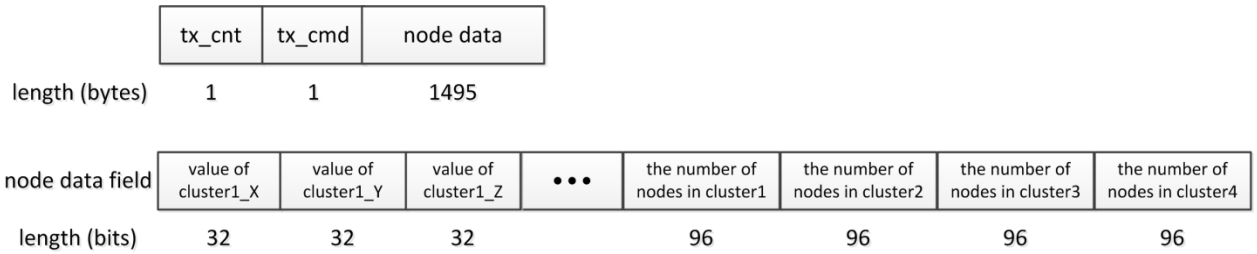


Figure 3.4: Transmission packet data field to host computer

3.2.2 K-means Flow Chart in FPGA

The VC707 EVB is implemented K-means clustering algorithm and the flow chart is shown in Fig. 3.5.

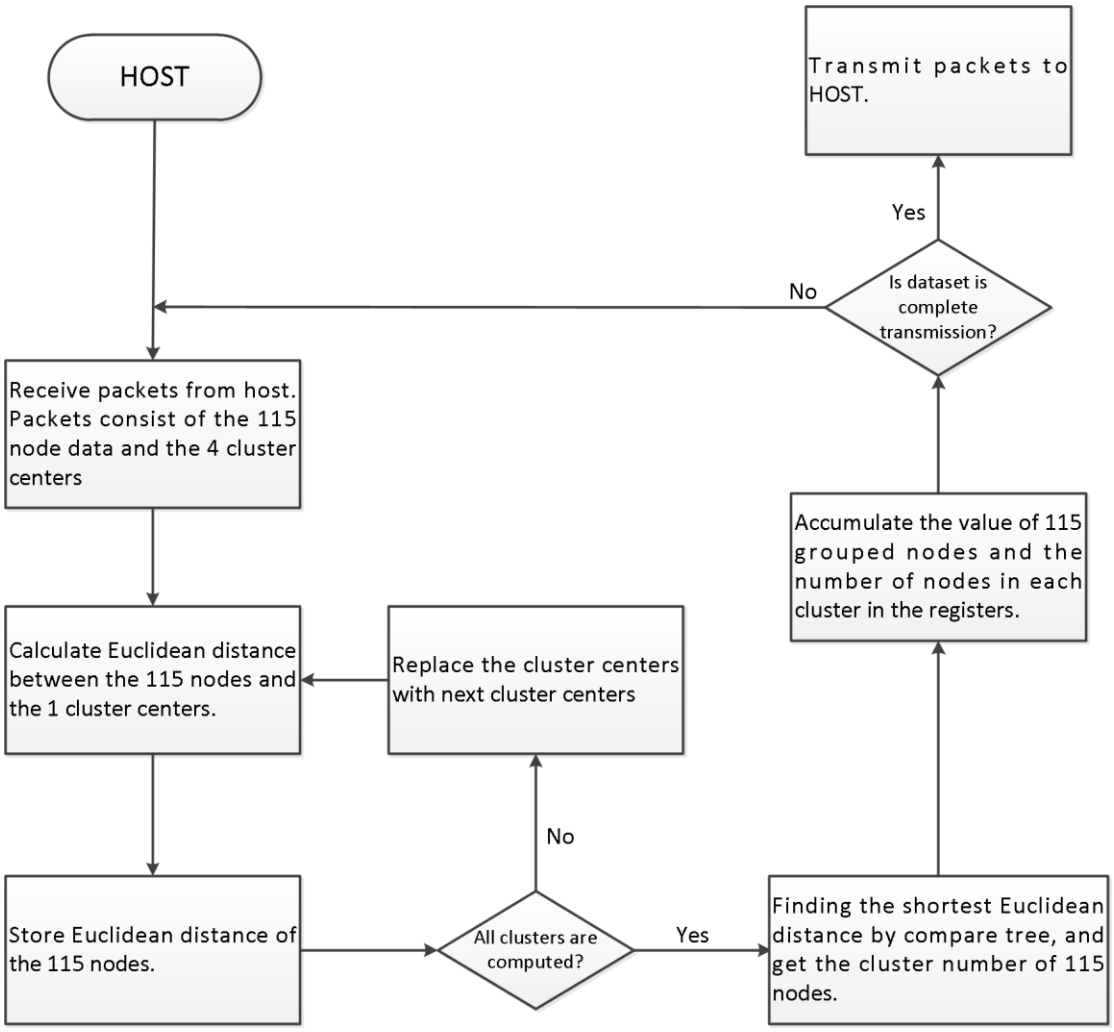


Figure 3.5: K-means flow chart in VC707 EVB

First, the VC707 EVB is waited for command from the host computer. After reception packet is received, we store the node value and the cluster centers in registers. Subsequently, we calculate the Euclidean distance between 115 nodes and one cluster center in parallel, and store result in the registers. If all cluster centers is already calculated with 115 nodes, we can get the cluster number of 115 nodes from the minimum Euclidean distance of each node. Otherwise, we prepare next cluster

centers, and calculate the Euclidean distance again. After that, we accumulate the grouped nodes of the cluster coordinate values and the number of nodes in each cluster into registers. Finally, we warp the cluster coordinate values and the number of nodes in each cluster into the packet, and transmit the packet to the host computer.

3.2.3 K-means Circuits

The Euclidean distance equation is shown in Eq. 3.1. Every value of nodes and centers is a single-precision floating point number, and the floating-point IPs help us to process floating-point operations.

$$\text{distance} = \sqrt[2]{(node_x - center_x)^2 + \left(node_y - center_y\right)^2 + (node_z - center_z)^2} \quad (3.1)$$

The Fig. 3.6 shows the circuit for calculation of Euclidean distance. We calculate 115 Euclidean distance between nodes and centers in parallel. The total of floating-point IP has 1035 modules including 345 subtractors, 345 square operations, 230 adders, and 115 square root operations. Both the input and the output of every IP modules are 32 bits. We get 115 distances of each node after six clock cycles. After calculation is done, we replace center value with next cluster centers.

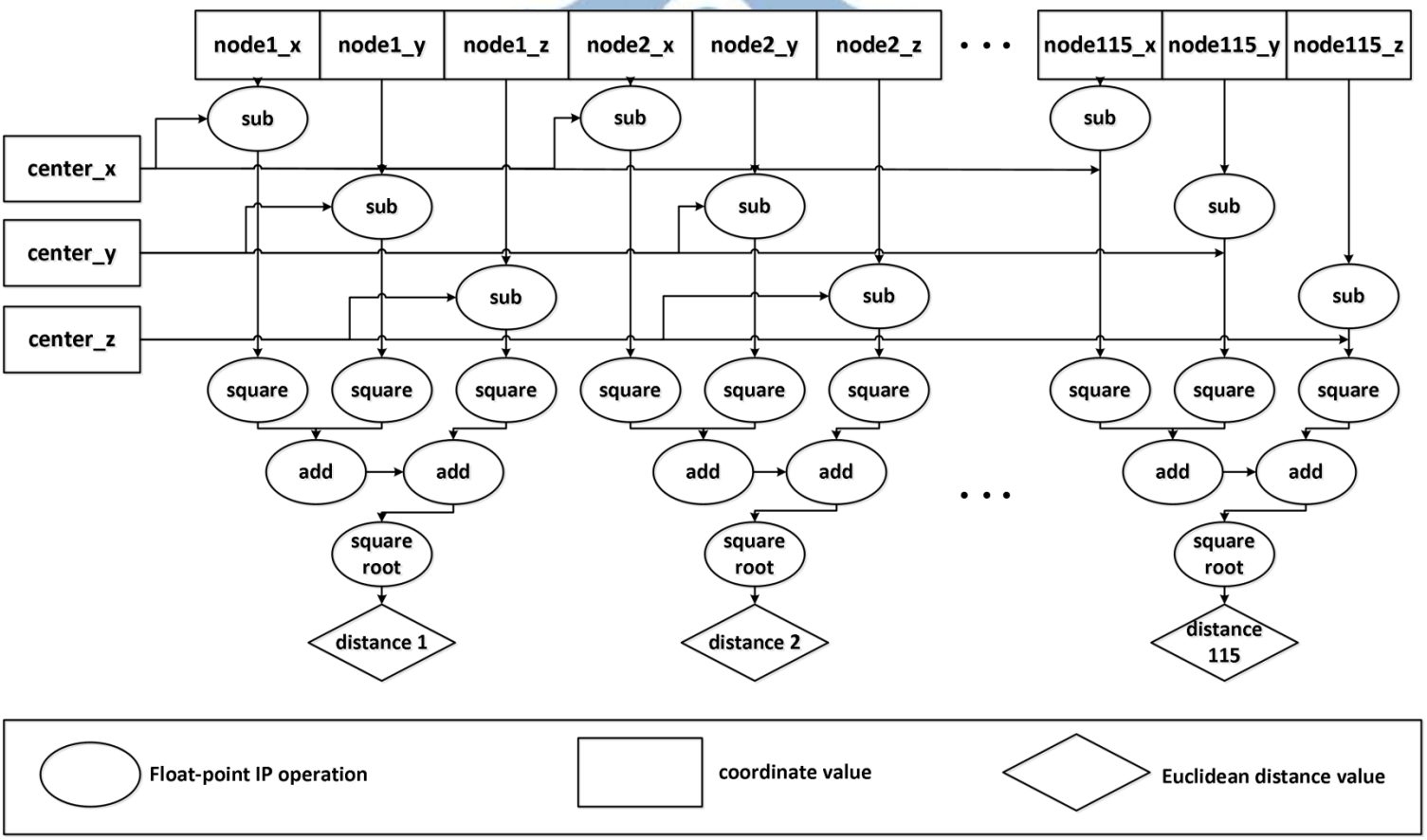


Figure 3.6: Euclidean distance circuits

After the four clusters distance between each node and centers are already calculated, we find the shortest distance in four distances by comparing binary tree, as shown in Fig. 3.7. In addition, we implement 115 comparing tree modules processing in parallel, and get the 115 cluster number of nodes after two clock cycles.

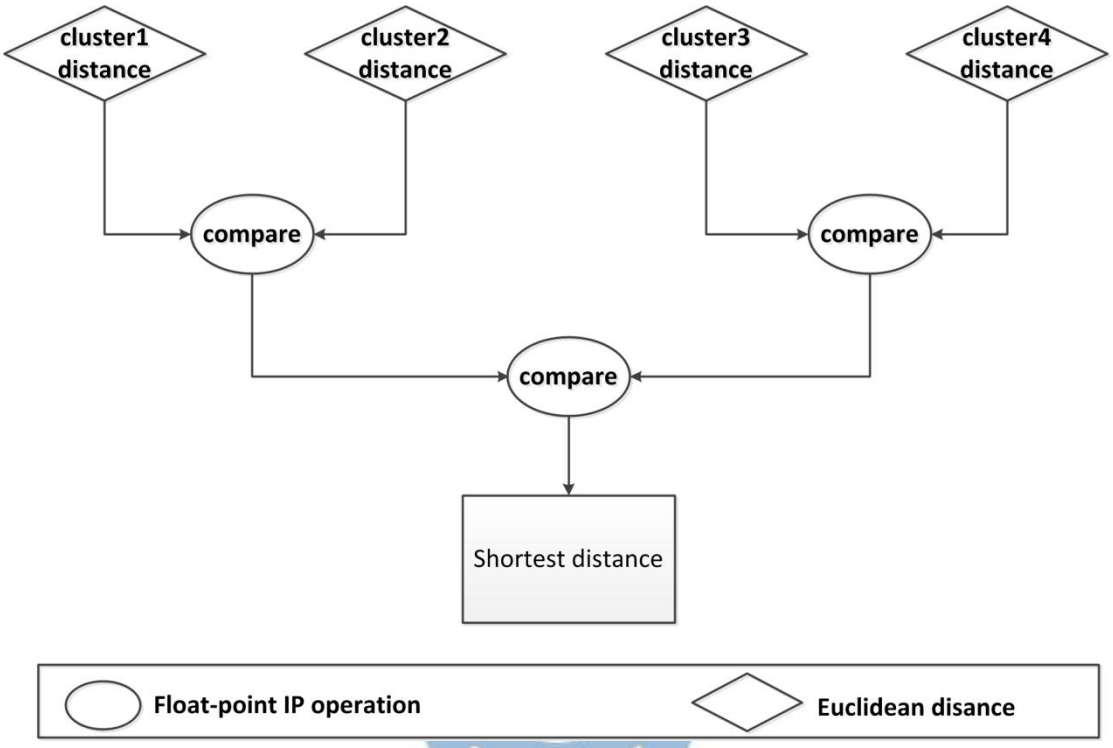


Figure 3.7: Comparing binary tree

Finally, we accumulate the coordinate values of 115 grouped nodes and the number of nodes in each cluster into registers. Besides, we do not calculate the 115 grouped nodes in parallel, because the resource of VC707 EVB is not enough to fulfill parallel circuits.

3.3 Master Processor

The master processor is implemented in host computer by visual studio C++ programing. The master processor is used to composition transmission packets, analysis reception packets, and calculation the cluster centers. We use union structure by C++ library to decompose floating-point into four unsigned characters, an unsigned character is basic unit of packet data. Therefore, we can wrap the floating-point into packets and transmit to the VC707 EVBs. Moreover, the single reception packet from the VC707 EVBs contains the sum of cluster coordinate values and the number of nodes in each cluster, we calculate the cluster centers after all dataset is clustering from three VC707 EVBs. Subsequently, we can do next iteration until the program is finished. Besides, the VC707 EVB is similar as a function, that once transmission to FPGA can process 115 nodes, and once reception from FPGA can get the sum of cluster values and the number of nodes in each cluster in once iteration. Therefore, we reduce the latency of reception time because the VC707 EVBs return packet only once time in once iteration.

Chapter 4

Experimental Results

4.1 Ethernet Transfer Rate

In the proposed FPGA-based hardware accelerator platform, the data transmission time between the computing server and the hardware accelerator platform depends on the I/O speed limitations of the Ethernet physical layer IP. When data are sent from the computing server to the hardware accelerator platform, the transmission data rate is tested and shown in Table 4.1. Oppositely, the transmission data rate from hardware accelerator platform to the computing server is also tested, and shown in Table 4.1.

In the single VC707 EVB, the usage of Ethernet rate is approximate 40 percent. Therefore, we share the workload to three VC707 EVBs and the usage of Ethernet rate is approximate 99 percent.

Table 4.1: Transmission rate between host computer and VC707 EVB

Direction	Packet length	Number of Packet	Time	Transmission data rate
Server to VC707 EVB	1500bytes	5,000	0.124s	483.87Mbps
VC707 EVB to server	1500bytes	5,000	0.909s	66.01Mbps

4.2 FPGA Resource Utilization

Table 4.2 shows the hardware resource utilization of the proposed hardware accelerator for K-means clustering algorithm.

Table 4.2: FPGA resource utilization

Slice Logic Utilization	Used	Available	Utilization
Number of Slice Registers	125,761	607,200	20%
Number of Slice LUTs	233,865	303,600	77%
Number of Occupied Slices	71,005	75,900	93%
Number of DSP48E1s	2,191	2,800	78%

4.3 Execution Time between Benchmark and FPGA-based Hardware Accelerator

Fig. 4.1 shows the execution time of K-means clustering algorithm (iteration three times) without latency of reading and writing data in 125 million three-dimensional node dataset. The execution time is compared to computer server with Intel I7-4770, computer server with Intel I5-3230M, and the proposed hardware accelerator with three VC707 EVBs. As shown in Fig. 4.1.

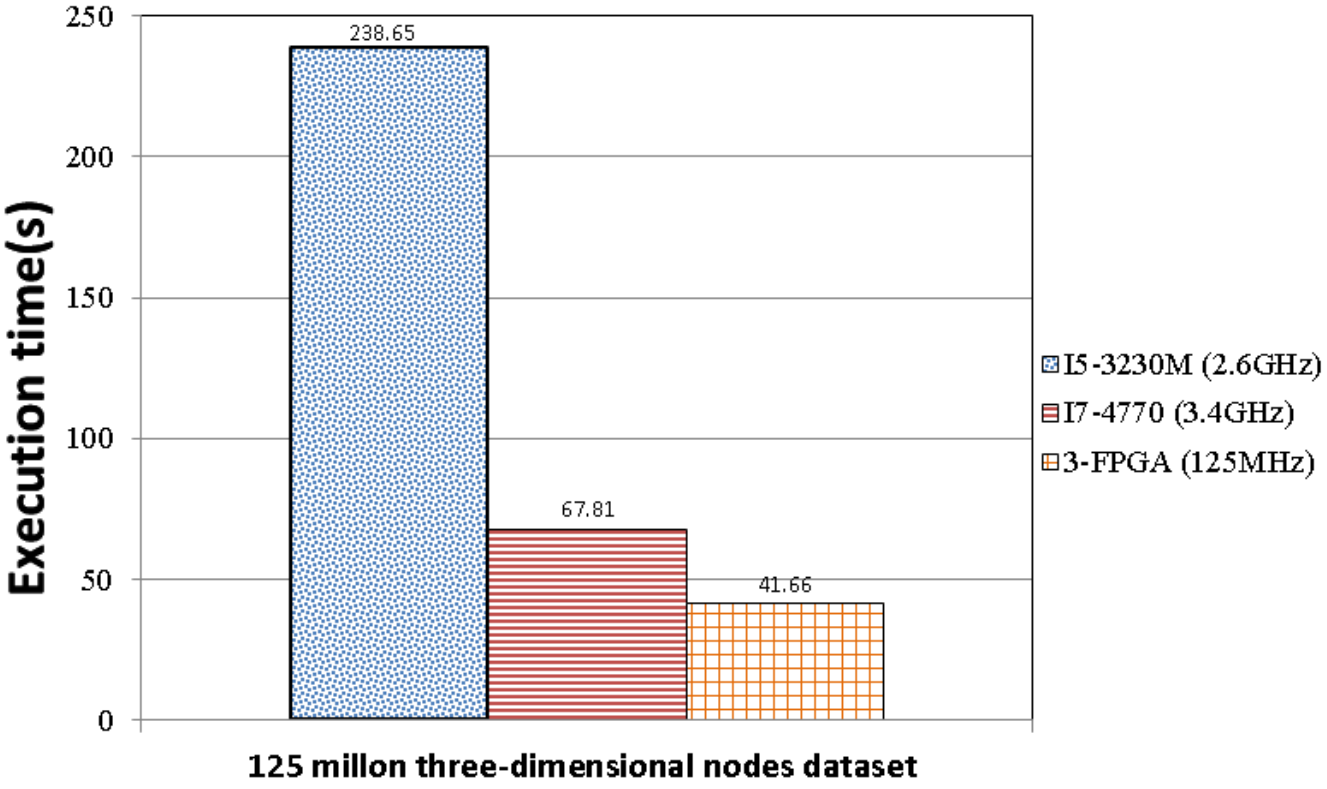


Figure 4.1: Compare 3-FPGA with I5-3230M (2.6GHz), and I7-4770 (3.4GHz) with 125 million three-dimensional nodes dataset

Fig. 4.2 shows the execution time without latency of reading and writing data for the different size of dataset. The execution time of computer server with Intel I7-4770, computer server with Intel I5-3230M, and the proposed hardware accelerator with three VC707 EVBs are growth linear if the datasets are growth.

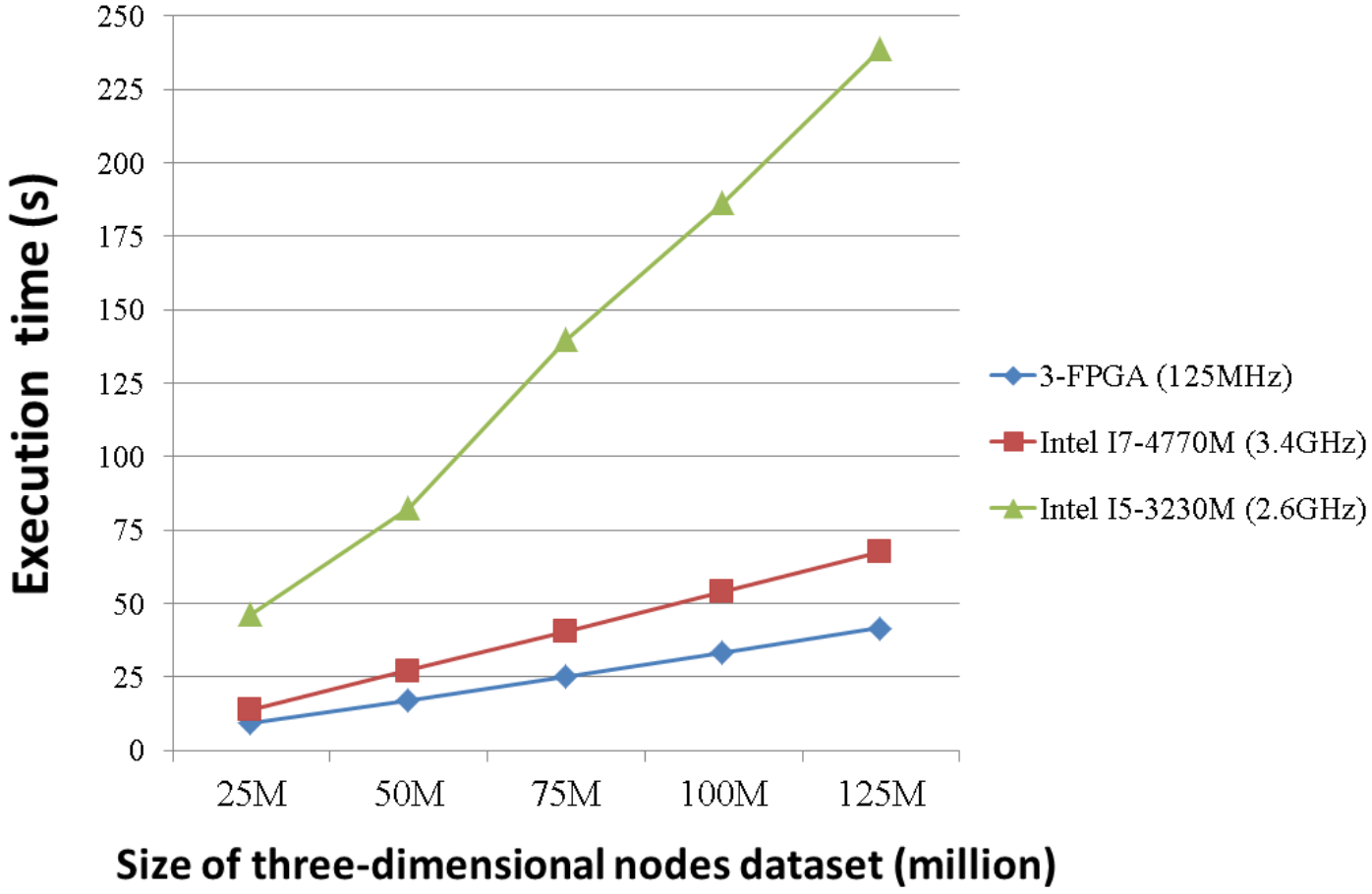


Figure 4.2: Compare 3-FPGA with I5-3230M (2.6GHz), and I7-4770 (3.4GHz) with 25/50/75/100/125 million three-dimensional nodes dataset

Fig. 4.3 shows the execution time with latency of reading and writing data for the different size of dataset. In fact, the disk I/O time is very long latency that is cost more than 90 percent of total time. Because the speed rate of disk is slower than CPU clock rate, and the miss penalty of memory and cache cost the latency. Therefore, Hadoop system can reduce disk I/O latency, because HDFS is able to spilt files into many blocks and transmits the blocks to others computing servers. But, the communication latency between master and slaves also costs latency. In addition, both the proposed hardware accelerator with three VC707 EVBs and computer server with Intel I7-4770 are same hard disk, but computer server with Intel I5-3230M hard disk is different.

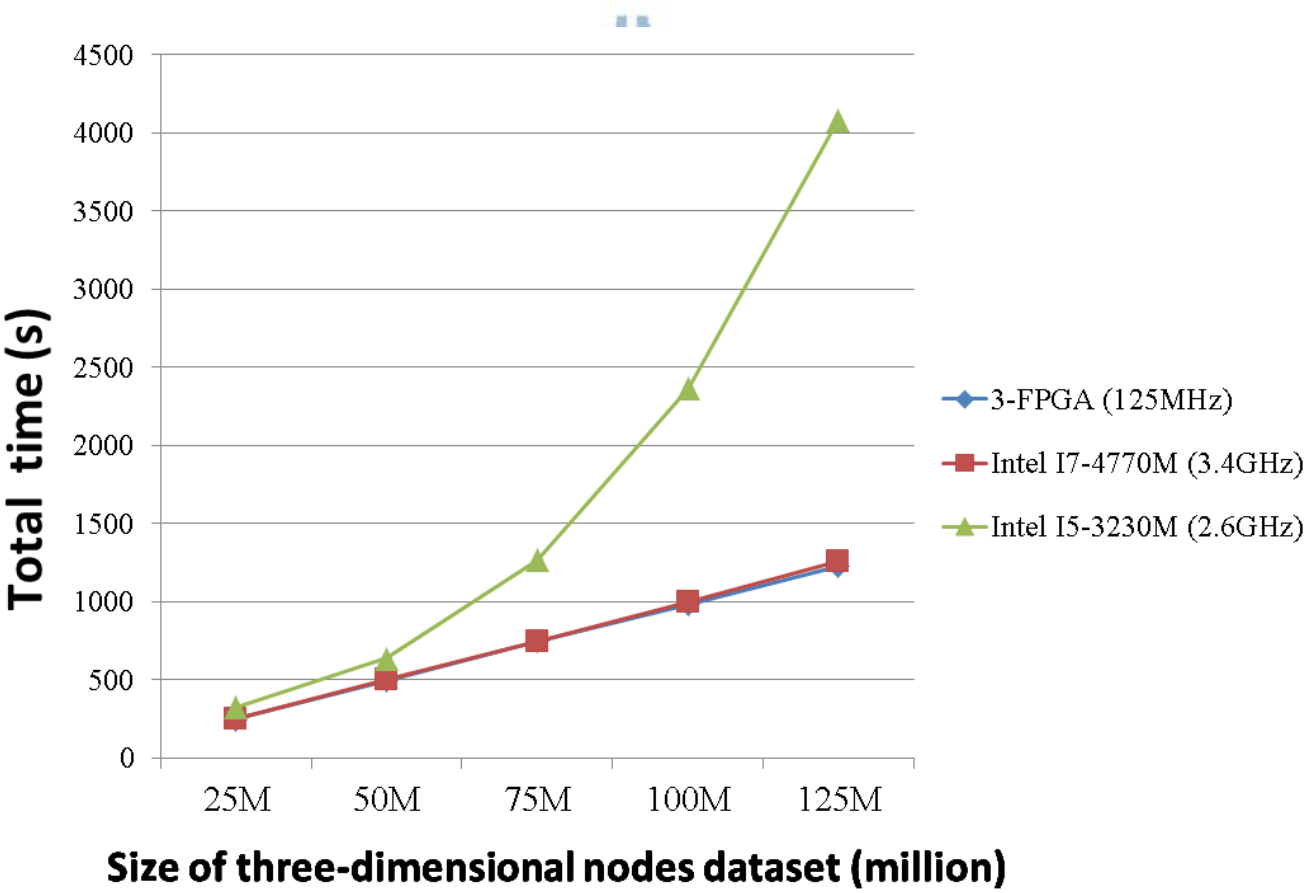


Figure 4.3: Compare 3-FPGA with I5-3230M (2.6GHz), and I7-4770 (3.4GHz) with 25/50/75/100/125 million three-dimensional nodes dataset, with disk I/O latency.

Fig. 4.4 shows the execution time for different size of dataset with several Hadoop (version 1.2.1) clusters servers and the proposed hardware accelerator with three VC707 EVBs. The master server has 8 cores Intel(R) Xeon(R) CPU E5506(2.13GHz), and all slaves are 8 cores including Intel(R) Xeon(R) CPU E5420(2.5GHz), Quad-Core AMD Opteron(tm) Processor 8356(2.29GHz) and Intel(R) Xeon(R)CPUE5520(2.27GHz).We use open source Apache Mahout [38] to implement K-means clustering in Hadoop. Except for Hadoop clusters with single server execution time is growth fast, the others performance is approximate equal. In fact, Hadoop system is needed to adjust the environment such as maximum map or reduce tasks and capacity of block sizes, the different conditions are affected performance. Therefore, we refer the Apache Hadoop [17] to adjust our Hadoop cluster environment. Besides, the execution time of three VC707 EVBs is faster than Hadoop clusters when dataset is small, because the communication latency between master and slaves costs time. But, the time of reading and writing data through hard disk costs latency, that Hadoop clusters execution time is growth slower than host computer. Therefore, Hadoop cluster is suited for computing the bigger datasets. Because of the communication, delivery block size, and job dispatch between Hadoop cluster servers can process in parallel. However, if we can reading or writing data in parallel in the hosts, the performance is improvement because hard disk I/O latency costs more time.

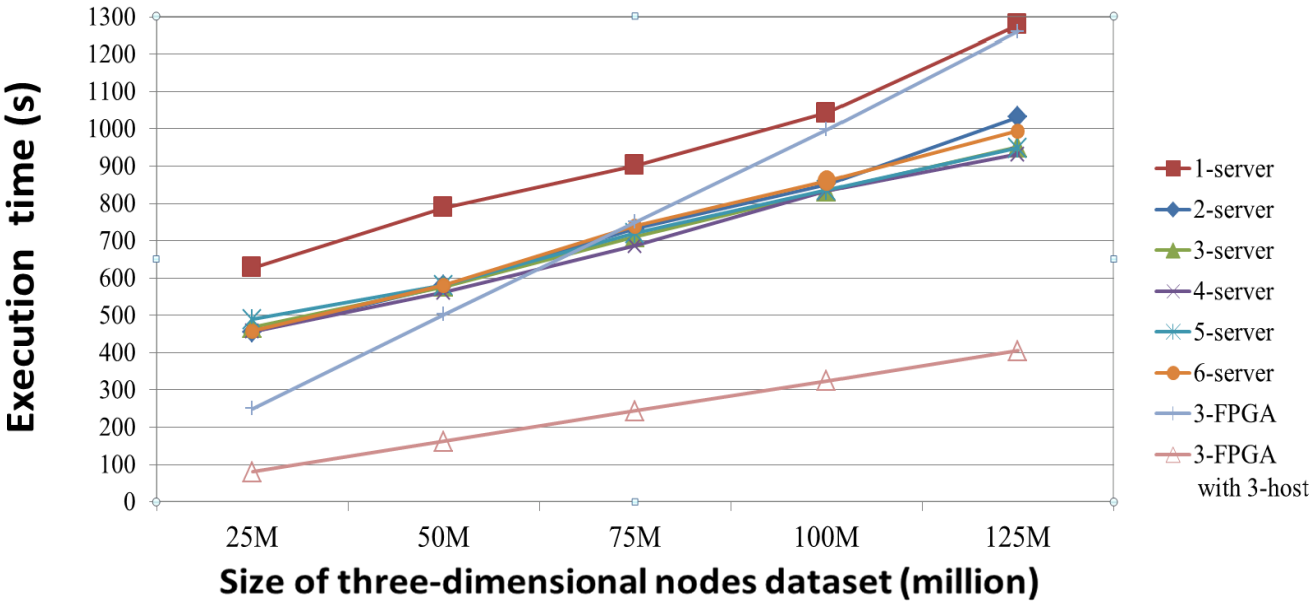


Figure 4.4: Compare Hadoop clusters with 1/2/3/4/5/6-servers, 3-FPGA, and 3-FPGA with 3-host with 25/50/75/100/125 million three-dimensional nodes dataset

Chapter 5

Conclusion and Future Works

5.1 Conclusion

In this thesis, an FPGA-based hardware accelerator platform for K-means clustering algorithm is presented. The proposed accelerator platform can use many VC707 FPGA EVBs to speed up the algorithm processing. Since server workloads will continue to evolve, the proposed FPGA-based hardware accelerator platform provides an easy way to support the CPUs for a heterogeneous computing solution with off-chip hardware accelerators. The experimental results for clustering 25 million three-dimensional node dataset shows that the proposed hardware accelerator platform with three FPGA EVBs at 125MHz clock rate can achieve the 2x speedup as compared with the computing server with an Intel I7-4770 CPU at 3.4GHz, and 10x speedup as compared with Hadoop clusters.

5.2 Future Works

In this thesis, we share workload to three VC707 EVBs and usage of Ethernet rate is approximate 99 percent, the Ethernet rate is reached limitation in our system. The system clock, Ethernet physical FIFO clock, and user core clock are too slow that is obstacle to speedup execution time. Therefore, we should re-build environment in single VC707 EVBs to speedup.

Besides, we can cooperate with Hadoop system and FPGA-based hardware accelerators. The Hadoop system is employed as the master and FPGA-based hardware accelerators are employed as the slaves. The HDFS in Hadoop can help us manage the files and split files into several blocks. The jobtracker can dispatch job to slaves. Therefore, the FPGA-based hardware accelerators focus on computation when get the jobs. However, the cooperation environment is difficult to build. We need to understand entire operations, behaviors, and architectures in Hadoop system, and realize the communication between HDFS, program, and scheduler. However, we consider to embed the hardware accelerators in Hadoop system that can handle the big data and improve performance.

REFERENCE

[1] Aviv Segev, Chihoon Jung, and Sukhwan Jung, "Analysis of technology trends based on big data," *in Proceedings of IEEE International Congress on Big Data (BigData Congress)*, Jun. 2013, pp. 419-420.

[2] Gerd Kortuem and Fahim Kawsar, "Market-based user innovation in the internet of things," *in Proceedings of Internet of Things (IOT)*, Nov. 2010, pp.1-8.

[3] Irena Bojanova, George Hurlburt, and Jeffrey Voas, "Imagineering an internet of anything," *Computer*, vol. 47, no. 5, pp. 983-987, Jun. 2014.

[4] John Gantz and David Reinsel, "The digital universe in 2020: big data, bigger digital shadows, and biggest growth in the far east," *in Proceedings of IDC iView, IDC Analyze the Future*, Dec. 2012, pp. 1-16.

[5] Jinson Zhang and Mao Lin Huang, "5Ws model for big data analysis and visualization," *in Proceedings of IEEE Conference on Computational Science and Engineering (CSE)*, Dec. 2013, pp. 1021-1028.

[6] Han Hu, Yonggang Wen, Tat-Seng Chua, and Xuelong Li, "Toward scalable systems for big data analytics: a technology tutorial," *IEEE Access*, vol. 2, pp. 652-687. Jul. 2014.

[7] Avita Katal, Mohammad Wazid, and R. H. Goudar, "Big data: issues, challenges, tools and good practices," *in Proceedings of International Conference on Contemporary Computing (IC3)*, Aug. 2013, pp. 404-409.

[8] Stephen Kaisler, Frank Armour, J. Alberto Espinosa, and William Money, "Big data: issues and challenges moving forward," *in Proceedings of Hawaii International Conference on System Sciences (HICSS)*, Jan. 2013, pp. 995-1004.

[9] Jinsong Zhang, Yan Chen, and Taoying Li, “Opportunities of innovation under challenges of big data,” *in Proceedings of IEEE International Conference on Computational Science and Engineering (CSE)*, Dec. 2013, pp. 1021-1028.

[10] Machine learning definition from Wikipedia. Available: http://en.wikipedia.org/wiki/Machine_learning

[11] Rajkumar Buyya, Chee Shin Yeo, and Srikumar Venugopal, “Market-oriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilities,” *in Proceedings of IEEE International Conference on High Performance Computing and Communications (HPCC)*, Sep. 2008, pp. 5-13.

[12] Peter Mell and Timothy Grance, “A NIST definition of cloud computing,” Available: http://csr c.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[13] Rick Cattell, "Scalable SQL and NoSQL data stores," *ACM SIGMOD Record*, vol. 39, no. 4, pp. 12-27. Dec. 2010.

[14] Jagdev Bhogal and Imran Choksi, “Handling big data using NoSQL,” *in Proceedings of IEEE International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, Mar. 2015, pp. 393-398.

[15] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, “The Google file system,” *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 29-34, Dec. 2003.

[16] Doug Beaver, Sanjeev Kumar, Harry C. Li, Jason Sobel, and Peter Vajgel, “Finding a needle in haystack: Facebook’s photo storage,” *in Proceedings of OSDI*, Oct. 2010, pp.1-8.

[17] Apache Hadoop. Available: https://hadoop.apache.org/

[18] Dr. Gene M. Amdahl, “Validity of the single processor approach to achieving large scale computing capabilities,” *IEEE Solid-State Circuits Society Newsletter*,

vol. 12, no. 3, pp. 19-20, Feb. 2009.

[19] Muhammad Shiraz, Mehdi Sookhak, Abdullah Gani, and Syed Adeel Ali Shah, "A study on the critical analysis of computational offloading frameworks for mobile cloud computing," *Journal of Network and Computer Applications*, pp. 47-60, Sep. 2014.

[20] Dong Hyuk Woo and Hsien-Hsin S. Lee, "Extending Amdahl's law for energy-efficient computing in the many-core era," *Computer*, vol. 41, no. 12, pp. 24-31, Dec. 2008.

[21] Ami Marowka, "Extending Amdahl's law for heterogeneous computing," *in Proceedings of IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA)*, Jul. 2012, pp. 309-316.

[22] Shuai Che, Jie Li, Jeremy W. Sheaffer, Kevin Skadron, and John Lach, "Accelerating compute-intensive applications with GPUs and FPGAs," *in Proceedings of Symposium on Application Specific Processors*, Jun. 2008, pp. 101-104.

[23] Qiang Wu, Yajun Ha, Akash Kumar, Shaobo Luo, Ang Li, and Shihab Mohamed, "A heterogeneous platform with GPU and FPGA for power efficient high performance computing," *in Proceedings of International Symposium on Integrated Circuits (ISIC)*, Dec. 2014, pp. 220-223.

[24] Brahim Betkaoui, David B. Thomas, and Wayne Luk, "Comparing performance and energy efficiency of FPGAs and GPUs for high productivity computing," *in Proceedings of International Conference on Field-Programmable Technology (FPT)*, Dec. 2010, pp. 94-101.

[25] Shuichi Asano, Tsutomu Maruyama, and Yoshiki Yamaguchi, "Performance comparison of FPGA, GPU and CPU in image processing," *in Proceedings of*

International Conference on Field Programmable Logic and Applications, Aug. 2009, pp. 126-131.

[26] Joshua Friedrich, Hung Le, William Starke, Jeff Stuechli, Balaram Sinharoy, Eric J. Fluhr, Daniel Dreps, Victor Zyuban, Gregory Still, Christopher Gonzalez, David Hogenmiller, Frank Malgioglio, Ryan Nett, Ruchir Puri, Phillip Restle, David Shan, Zeynep Toprak Deniz, Dieter Wendel, Matt Ziegler, and Dave Victor, "The POWER8™ processor: designed for big data, analytics, and cloud environments," *in Proceedings of IEEE International Conference on IC Design and Technology (ICICDT)*, May 2014.

[27] David Andrews, Douglas Niehaus, and Peter Ashenden, "Programming models for hybrid CPU/FPGA chips," *Computer*, vol. 37, no. 1, pp. 118-120, Jan. 2004.

[28] Eduardo Gudis, Pullan Lu, David Berends, Kevin Kaighn, Gooitzen van der Wal, Gregory Buchanan, Sek Chai, and Michael Piacentino, "An embedded vision services framework for heterogeneous accelerators," *in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jun. 2013, pp. 23-28.

[29] Apostolos Dollas, "Big data processing with FPGA supercomputers: opportunities and challenges," *in Proceedings of IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Jul. 2014, pp. 474-479.

[30] Zhongduo Lin and Paul Chow, "ZCluster: a Zynq-based Hadoop cluster," *in Proceedings of International Conference on Field-Programmable Technology (FPT), Dec*. 2013, pp. 450-453.

[31] Mike Estlick, Miriam Leeser, James Theiler, and John J. Szymanski, "Algorithmic transformations in the implementation of k-means clustering on reconfigurable hardware," *in Proceedings of ACM/SIGDA International*

Symposium on Field Programmable Gate Arrays, Feb. 2001, pp. 103-110.

[32] Xiaojun Wang and Miriam Leeser, "K-means clustering for multispectral images using floating-point divide," *in Proceedings of IEEE Symposium on Field-Programmable Custom Computing Machines*, Apr. 2007, pp. 151-162.

[33] Tsutomu Maruyama, "Real-time k-means clustering for color images on reconfigurable hardware," *in Proceedings of International Conference on Pattern Recognition*, Aug. 2006, pp. 816-819.

[34] Zhongduo Lin, Charles Lo, and Paul Chow, "K-means implementation on FPGA for high-dimensional data using triangle inequality," *in Proceedings of International conference on Field Programmable Logic and Applications (FPL)*, Aug. 2012, pp.437-442.

[35] Hanaa M. Hussain, Khaled Benkrid, Huseyin Seker, and Ahmet T. Erdogan, "FPGA implementation of k-means algorithm for bioinformatics application: an accelerated approach to clustering microarray data," *in Proceedings of NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, Jun. 2011, pp. 248-255.

[36] Anjan Goswami, Ruoming Jin, and Gagan Agrawal, "Fast and exact out-of-core k-means clustering," *in Proceedings of IEEE International Conference on Data Mining*, Nov. 2004, pp. 83-90.

[37] VC707 evaluation board for the Virtex-7 FPGA user guide, Xilinx Inc., Available:http://www.xilinx.com/support/documentation/boards_and_kits/vc707/ug885_VC707_Eval_Bd.pdf

[38] Apache Mahout, Available: http://mahout.apache.org/