HARDWARE BASED ACCELERATOR FOR DATABASE QUERY USING M-TREE

CHAI KAH HIENG

UNIVERSITI TEKNOLOGI MALAYSIA

HARDWARE BASED ACCELERATOR FOR DATABASE QUERY USING M-TREE

CHAI KAH HIENG

A project report submitted in partial fulfilment of the
requirements for the award of the degree of
Master of Engineering (Computer and Microelectronic Systems)

Faculty of Electrical Engineering
Universiti Teknologi Malaysia

JUNE 2018

ACKNOWLEDGEMENT

First of all, I would like to express the deepest appreciation to my supervisor Muhammad Nadzir Marsono for continuously giving me advice, support, guideline and supervise on this project. Besides, I would like to thank my coursemates and colleagues in Intel for their supports and assistance in this project. In addition, I would like to express my appreciativeness to my parents and friends for giving me mentally support.

ABSTRACT

Fast indexing is an indexing methods that sort the database and enable content to be accessed quickly. Fast query is part of fast indexing which able to perform the query within a narrow range to reduce the query time. In fast query, machine learning has played an important role on automate the tasks. In state-of-the-art, fast query algorithm are built using software where the performance of the query process is based on the performance of the general-purposed CPU. Besides, the total query time is linearly proportional to the data size where the difficulty of fast query is increasing as the data size increase which result in longer query time. Thus, a hardware accelerator for fast query is proposed in this work. M-tree is a fast indexing algorithm using tree data structure. M-tree data structure is constructed based on metric space and relied on triangle inequality which offer efficient range and k-nearest neighbor (k-NN) queries. The hardware accelerator is implemented using Xilinx's Vivado Design Suite which targeted on FPGA platform. The hardware accelerator is coded using System Verilog HDL. The hardware accelerator is focusing on the fast query algorithm. The hardware accelerator is designed to be generic which could be implement on different FPGA board. The hardware accelerator has been evaluated by running the comparison on the performance with the existing work which is the M-tree algorithm running in software. The hardware accelerator is able to achieve significant speedup at approximatly 1000 times on the performance of query process compare to the M-tree running in software. The overall performance of the hardware accelerator for several scenario also shown the speedup compare to software based fast query.

ABSTRAK

Pengindeksan cepat adalah kaedah pengindeksan yang menyusun pangkalan data dan membolehkan kandungan dapat diakses dengan cepat. Pertanyaan pantas adalah sebahagian daripada pengindeksan cepat yang dapat melakukan pertanyaan dalam jarak sempit untuk mengurangkan waktu permintaan. Dalam pertanyaan pantas, pembelajaran mesin telah memainkan peranan penting dalam mengautomasikan tugas-tugas. Algoritma pertanyaan cepat-cepat yang dibina menggunakan perisian di mana prestasi proses pertanyaan adalah berdasarkan prestasi CPU yang dirancang secara umum. Selain itu, jumlah masa pertanyaan secara linear berkadaran dengan saiz data di mana kesukaran permintaan cepat semakin meningkat kerana peningkatan saiz data yang menyebabkan masa pertanyaan yang lebih lama. Oleh itu, pemecut hardware untuk pertanyaan pantas dicadangkan dalam kerja ini. M-tree adalah algoritma pengindeksan cepat menggunakan struktur data pokok. Struktur data M-tree dibina berdasarkan ruang metrik dan bergantung kepada ketidaksetaraan segitiga yang menawarkan jawapan yang cekap dan k-terdekat jiran terdekat (k-NN). Penderas perkakasan dilaksanakan menggunakan Xilinx's Vivado Design Suite yang disasarkan pada platform FPGA. Penderas perkakasan dikodkan menggunakan Sistem Verilog HDL. Penderas perkakasan memfokuskan pada algoritma pertanyaan pantas. Penderas perkakasan direka untuk menjadi generik yang boleh dilaksanakan pada papan FPGA yang berbeza. Penderas perkakasan telah dinilai dengan menjalankan perbandingan prestasi dengan kerja yang sedia ada yang merupakan algoritma M-tree yang berjalan dalam perisian. Puncak perkakasan boleh mencapai kelajuan maksimum kira-kira 1000 kali pada prestasi proses pertanyaan berbanding dengan pokok M yang berjalan dalam perisian. Prestasi keseluruhan pemecut perkakasan untuk beberapa senario juga menunjukkan kelajuan berbanding dengan permintaan cepat berasaskan perisian.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	-	Artificial intelligent
ASIC	-	Application-Specific Integrated Circuit
CPU	-	Central Processing Unit
CU	-	Control Unit
DU	-	Datapath Unit
FIFO	-	First-in-First-Out
FPGA	-	Field-Programmable Gate Array
HDL	-	Hardware Description Language
IOT	-	Internet of Things
IP	-	Intellectual Property
k-NN	-	k-Nearest Neighbor
SMP	-	Symmetric multiprocessing

CHAPTER 1

INTRODUCTION

1.1 Problem Background

Big data, artificial intelligent (AI) and internet of things (IOT) had become the hot areas in current trend and lead to the rapid growth of data size [5]. There are many sources that predict the growth of data toward 2020. However, most of the predictions are in broad agreement that the size of the digital data will become twice on every two years which will result in 50 times on data growth from 2010 to 2020 [6]. According to a Cisco forecast, total data center traffic is projected to hit 15.3 zettabytes, or 15.3×10^21 GB, by the end of 2020 [7]. In addition, overall growth rate of human generated data is approximately 10 times faster than traditional business data, and machine data is increasing much more rapidly at 50 times of current growth rate. One of the perdition on data growth is shown in Figure 1.1.

Database could be consist of different types of information. Furthermore, database could having huge amount of similar data under one categorize. However, there is only certain data is needed for each kind of event and the remaining data might be redundant or non-related. Hence, linear query which is the basic query method that searching throughout the entire database and looking for the matching information has

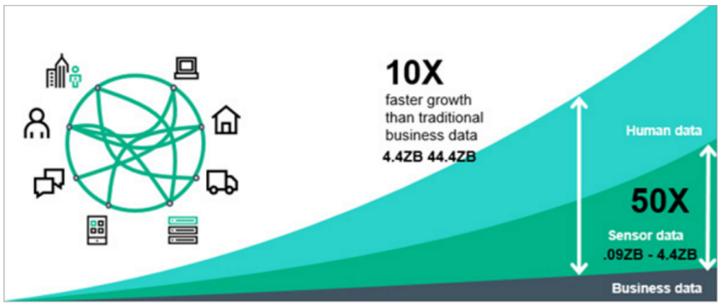


Figure 1.1: Prediction of data growth rate toward 2020 [1]

been widely used to pick up the wanted data. With the exponential grow of data size along with current trend, linear query is no longer sufficient for current situation as the searching will took longer time [8]. As the processing on redundant data will introduce unnecessary run time during the data processing. In addition, the performance of the data processing will be further reduced due to the limitation on data retrieval rate of current disk type where accessing unnecessary data entries will consume the bandwidth of data retrieval rate and require longer access time of data from memory. Therefore, the need of fast query for the database entries has been magnified.

There are multiple query algorithms such as M-tree have been developed by researcher in software which is able to reduce the query time significantly. In query algorithm, machine learning has played an important role. Machine learning is a method of data analysis that using algorithms to learn from data and build the analytical model automatically without being explicitly programmed [9]. The basic concept of the query algorithm is trying to narrow down the searching area and thus improve the query time. While the database size is continue to growth, the difficulty is also increasing. Therefore, acceleration for fast query is needed for incoming demand.

In recent years, field-programmable gate array (FPGA) has been widely used. Field-programmable gate array (FPGA) is an integrated circuit that designed to be configured by a user after manufacturing. FPGA became the platform that allow the user to design, develop, and test the prototype of integrated circuit. Hence, there is possible to archive hardware acceleration. Hardware acceleration is the use of hardware to perform some functions more efficiently than running in software on a general-purpose CPU. One of the example is Application-Specific Integrated Circuit (ASIC) which is an integrate circuit that being designed to perform specific application and able to provide better performance compare to software.

1.2 Problem Statement

Database query algorithm has been developed to reduce the query time. However, the total query time is linearly proportional to database size [10]. The larger the size of database, the longer the query time. In state-of-art, the query algorithm is mostly rely on the tree data structure which is able to eliminate the searching on unnecessary data and improve the query time. However, the level of tree is increasing when the database size increase and causing the difficulty of query increase. In practical, this is unable to be change. Thus, acceleration is needed to speed up the query process.

Besides, the total query time also proportional to the query size [11]. As the number of queries increase, the query time increase. During the event that required to retrieve multiple data such as the data within a range or data with similarity, the query process will invoke the search on the all the data that match the searching criteria which increase the query time. This also the event that is unchangeable, so the acceleration is needed.

Furthermore, performance of query using software is relied on computation unit [12]. In state-of-art, query algorithms are developed in software and software is relying on computation unit to perform the tasks. Thus, device with good computation unit is needed to observe good performance while performing query using software.

1.3 Objective and Scope

In this project, a hardware accelerator is being design and developed to perform the fast query of database. The M-tree query algorithm has been studied and translated into hardware architecture. Besides, the performance of the designed hardware accelerator has been evaluate and comparison with software is done.

The objective of this work is to develop a hardware based accelerator for database queries using M-tree range search algorithm to accelerate the query process. The designed hardware accelerator is needed to provide the speedup compare to the software while performing the query.

Besides, the work is also developed to accelerate the query of multiple data. The designed hardware accelerator is needed to have the ability to perform the multiple query in once. Furthermore, the speedup of the multiple query is required.

In addition, this works is proposed to eliminated the dependency on computation unit for the query process. The hardware accelerator is designed to be a separate hardware unit such that the query process no need to rely on a general-purpose CPU.

The scope of this project is focusing on the architecture design of hardware accelerator. The hardware accelerator is implemented using System Verilog Hardware Description Language (HDL) for FPGA platform. The proposed work will focus on fast query for database only. Besides, the proposed work is developed using the range search algorithm based on M-tree Queries. In addition, the proposed work is limit the node capacity of the M-tree data structure to two. Moreover, One dimensional fixed point numerical data set is used in this project.

1.4 Organization

The report is organized with 5 chapters. Chapter 1 provide the introduction of the project including background, problem statement, objective, and scope. Chapter 2 reviews the related literature review on state-of-the-art of query algorithm and other related works. Chapter 3 describe the methodology for the overview of proposed work, hardware accelerator architecture, query operation, and experimental setup. Chapter 4 elaborates the simulation results of the proposed work, comparison to the existing work, and discussion. Chapter 5 concludes the project accomplishment and discuss on the future works.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction to Fast Query

Fast query is part of fast indexing which is used for the purpose of improving the query time. In order to perform fast query, data indexing needed to be done in advance. Data indexing is the process which create the index for the data based on the similarity and the indexed data will be the input for fast query. Fast query is the process that perform the query of information in a narrow range of data set without accessing unnecessary data which reduce the query time significantly.

Most of the fast query algorithm rely on the tree data structure such that fast query is able to pruned the sub-tree whenever the searching criteria is not met. By pruning the sub-tree, unrelated data will be excluded from the search, and thus the searching area could be narrow down. By narrow down the searching area, the query efficiency is improved as the searching only happen on minimal entries. In addition, this technique eliminate the accessing to the unnecessary data where there is no extra accessing time being introduced.

There are multiple type of fast indexing algorithm has been developed such as B-tree, R-tree, and M-tree. Different fast indexing algorithm using different method to perform the indexing and the query M-tree has been chosen to be using in this project as M-tree offer better performance over others fast indexing algorithm. Besides, M-tree is much convenient will implementing in the hardware as the algorithm is based on the metric space.

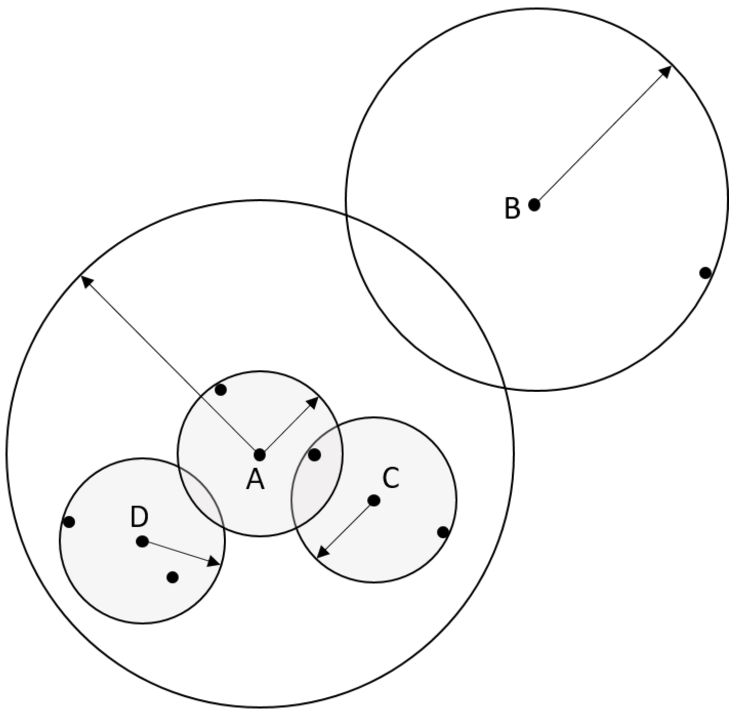


Figure 2.1: Euler diagram of an example data set [2]

2.2 M-tree

M-trees are tree data structures for indexing multi-dimensional information which is developed in software by Ciaccia, P., Patella, M. and Zezula. M-tree is being constructed based on metric space and is relies on the triangle inequality. Metric space is the distance function that define the distance between each pair of elements. Triangle inequality is a theory state that the the sum of the lengths of any two sides of any triangle must be greater or equal to the length of another side. By using these two methods, the database could be expressed in Euler diagram based on the distance of each element as shown in Figure 2.1. From the Euler diagram in Figure 2.1, there are parents sets and sub-sets. Then, M-tree could be built by sorting the sets into a tree structure as shown in Figure 2.2.

As the data is being indexed by sets which having a covering radius, M-tree offer efficient range and k-nearest neighbor (k-NN) queries. Besides, M-tree is able to handle multi-dimensional data because the algorithm is having independent distance function where the distance function could be change to support different dimensional data set without affecting the flow of algorithm.

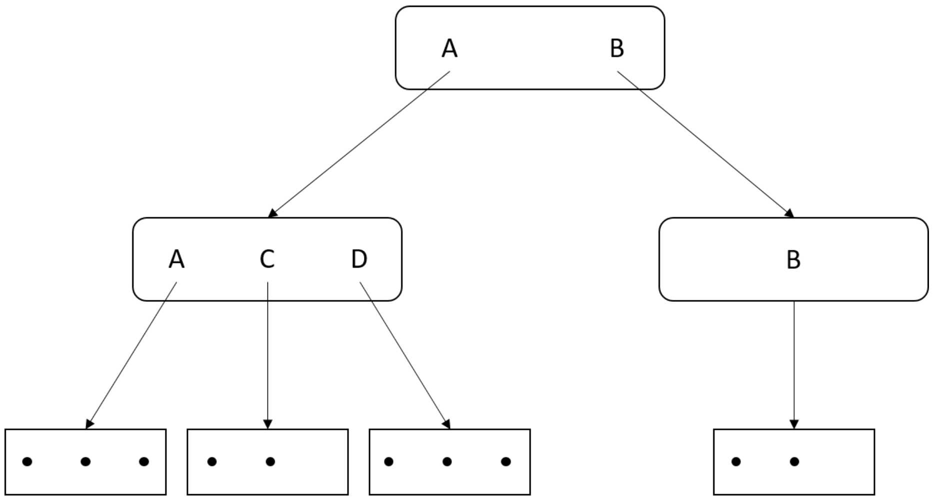


Figure 2.2: Constructed M-tree from example data set [2]

2.2.1 Data structure

M-tree consist of four main components which are non-leaf nodes, leaf nodes, routing object and object. Non-leaf nodes is a set of routing object and leaf nodes is a set of object. Routing object is having sub-components including feature value of routing object, covering radius, distance from parent object, and pointer to covering tree. Object is having the sub-components including feature value of object, object identifier and distance from parent object.

The example of M-tree data structure is shown in Figure 2.3. All the routing objects on the top level of tree is forming a non-leaf node. Then, each of the routing object is pointing to the non-leaf nodes in next level that consist of the routing objects under it's covering radius. The routing objects will continue pointing to the nodes in next level until the pointer is point at the leaf nodes which is the set of objects. Finally, the objects are pointing to the object identifier which is the actual content of the object.

For each routing object, the feature value of routing object, covering radius, distance from parent object, and the pointer are being stored in the data structure. This is same for the object where the object identifier and the distance to parent are being stored in the data structure. With the information that stored in the data structure, the query process could be performed.

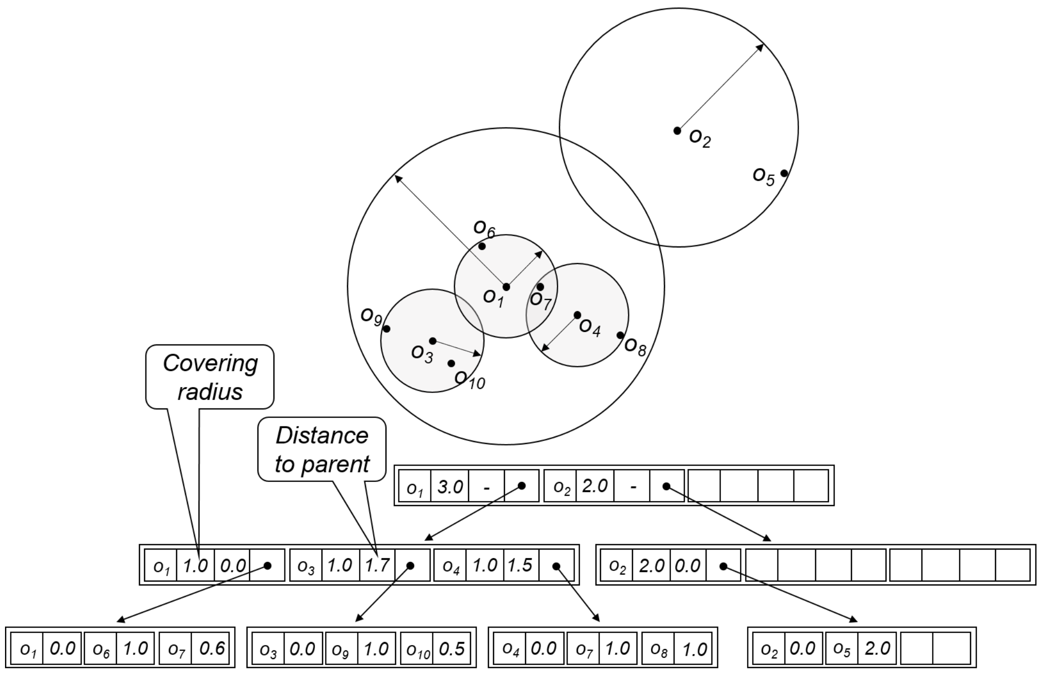


Figure 2.3: M-Tree data structure of example data set [3]

2.2.2 Range Queries algorithm

A range query is perform the search and return all the items within the specified range. The M-tree range query algorithm is working in the way that prune all the entries that does not match the search criteria and only perform the search for all non-pruned entries. In order to prune the entries, triangle inequality is used to perform the check. According to triangle inequality, Equation (2.1) could be written.

$$d(O_P, Q) \leq d(O_r, O_P) + d(O_r, Q) \tag{2.1}$$

where O_P is parent object, O_r is routing object, and Q is query object. Then, the equation could be re-write into Equation (2.2).

$$|d(O_P, Q) - d(O_r, O_P)| \leq d(O_r, Q) \tag{2.2}$$

Equation (2.2) shows that the distance between the routing object and the query object is greater or equal to the differences of the distance between parent object and query object and the distance between routing object and parent object. By considering the covering radius, the relationship between the routing object and query object will be

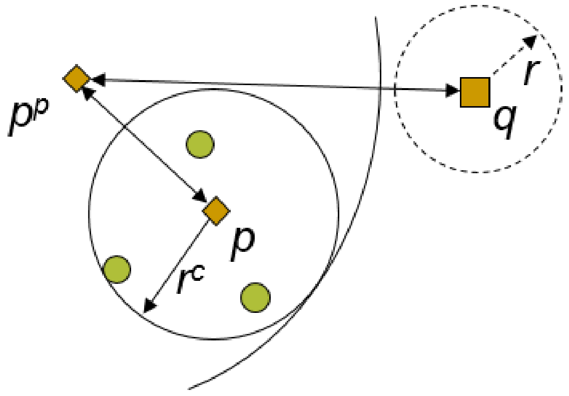


Figure 2.4: Prune condition for non-leaf nodes [3]

achieved only if the minimum requirement is met such that the sum of both covering radius is equal to the distance between the routing object and query object. Thus, the searching criteria between the routing object and query object could be written in Equation (2.3).

Therefore, M-tree range query is using the Equation (2.3) to check the validity for the entries in non-leaf node. In this event, M-tree will prune all the sub-tree that does not fulfill this condition. Figure 2.4 illustrated the situation that the routing object does not fulfill the condition where the routing object could be safely pruned from the search.

$$|d(O_P, Q) - d(O_r, O_P)| \leq r(Q) + r(O_r) \tag{2.3}$$

In addition, The relationship between the routing object and the query object is achieved only if the sum of both covering radius is greater or equal the distance between the object and routing object. Thus, the valid searching criteria for the routing object could be expressed in Equation (2.4). Figure 2.5 illustrated the situation that the object does not fulfill the condition where the object could be safely pruned from the search.

Hence, the condition checking for the non-leaf node is done in two stages for all the routing objects in the node. In first stage, Equation (2.3) is used to perform quick checking and prune the unrelated sub-tree. Then, Equation (2.4) is used to perform detail checking in second stage.

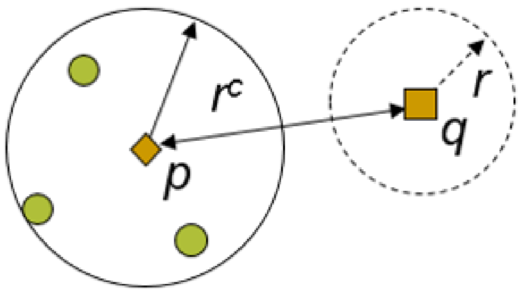


Figure 2.5: Prune condition for leaf nodes [3]

$$d(O_r, Q) \leq r(Q) + r(O_r) \tag{2.4}$$

Once the routing object is fulfill the condition and the range search is arrived at the leaf nodes of the M-tree which content the objects, there is require another checking to validate whether the object is a valid query object.

Since there is no covering radius for the object, the equation of the condition checking could be re-write into Equation (2.5) and Equation (2.6). Therefore, the condition checking for the leaf node is similar to the non-leaf node but using the Equation (2.5) and Equation (2.6).

$$|d(O_P, Q) - d(O_r, O_P)| \leq r(Q) \tag{2.5}$$

$$d(O_r, Q) \leq r(Q) \tag{2.6}$$

The pseudo code of M-tree range queries algorithm could be written as shown in Figure 2.6 where O_P, O_r, and O_j are parent object of node, routing object, and object respectively. The inputs of the M-tree range queries are node of M-tree (N), query object (Q), and search radius (r(Q)). Then, the output of the M-tree range queries is all the objects that fulfill the searching criteria.

```
{
  let $O_p$ be the parent object of node $N$;

  if $N$ is not a leaf then {
    for each entry($O_r$) in $N$ do {
          if $|d(O_p,Q) - d(O_r,O_p)| \leq r(Q) + r(O_r)$ then {
            Compute $d(O_r,Q)$;
            if $d(O_r,Q) \leq r(Q) + r(O_r)$ then
              RangeSearch(*ptr($T(O_r)$),$Q$,$r(Q)$);
          }
    }
  }
  else {
    for each entry($O_j$) in $N$ do {
          if $|d(O_p,Q) - d(O_j,O_p)| \leq r(Q)$ then {
            Compute $d(O_j,Q)$;
            if $d(O_j,Q) \leq r(Q)$ then
              add $oid(O_j)$ to the result;
          }
    }
  }
}
```

Figure 2.6: Pseudo-code of M-tree range queries algorithm [4]

In the pseudo code, there is separate into two parts which are the check for non-leaf nodes and leaf node. For non-leaf node, all entries in the node will being checked with the condition from Equation (2.3) and Equation (2.4). If both of the condition are fulfilled, another RangeSearch is being triggered with the pointer to the next node. For the leaf node, all entries in the node will being checked with the condition from Equation (2.5) and Equation (2.6). If both of the condition are fulfilled, the object identifier which is the actual content of the object is returned as the result.

2.3 Related Works

In order to have better data retrieval rate for current needs, fast query had attracted researcher interest. Due to the different data type and complexity, many fast indexing algorithm has been developed such as B-tree [13], R-tree [14], and M-tree [4]. Different fast indexing algorithm is using different method to constructed the data tree and different queries algorithm to perform the query. This project is focusing on M-tree algorithm only. Hence, the related work on M-tree has been reviewed.

2.3.1 Performance of M-tree queries

The performance of the query process is the key to reduce the query time. There is a many works has been done by researcher to evaluate the performance of the M-tree query algorithm. The performance evaluation is usually done by comparing with others query algorithm. In most of the work, there is showing that M-tree having the better performance in query process compare to other query algorithm. In [4] stated that M-tree is always efficient than R-tree even without optimization. Besides, [15] stated that M-tree is having strong pruning power which allow fewer distance computation. Hence, the query time is being reduced as the lesser distance computation is performed. Furthermore, M-tree is efficient in distance computation which allow saving up to 40% distance computation time [4].

M-tree is able to handle multi-dimensional while preserving the query performance. There is the fact that the cost of distance computation increased as the the number of dimensions increased [16]. The M-tree query algorithm is using metric-based index which had efficiently reduce the I/O cost and number of distance computation[16]. Thus, M-tree is able to perform well in multidimensional because of the reduced distance computation.

M-tree is also efficient in range queries. Since M-Tree algorithm only requires index access, the overall performance for the range query is good and remains fast even as the query size increases [17]. Besides, the sensitivity of the range queries is higher in M-tree when the data set is not uniform [4]. In [17], the paper shows that the variance of the M-Tree performance is much smaller. Therefore, M-Tree queries performs the best in low density areas. In addition, M-tree is able to have better performance for larger range queries which has been demonstrated in [17].

However, the disadvantages of M-tree is that M-tree required higher CPU costs compare to R-tree [4]. Besides, M-tree is required longer time to constructed the tree compare to R-tree [16]. Futhermore, the total query time is longer as the size of data set increase [18].

2.3.2 Speedup for M-tree queries

Along with the exponential grow of database size, current query algorithm will no longer sufficient for future need due to the statement of the query time is proportional to the database size [10]. Thus, speeding up the query process has became one of the research area. There are several works has been done to speedup the current M-tree algorithm. However, the existing works on speedup for M-tree are done in software only. The existing works on speedup of M-tree has been done in two ways which are parallelism technique and tree optimization.

The parallelism technique is running the M-tree algorithm in parallel and the speedup could be achieved. The concept of the parallelism technique is eliminating all the queuing process by running all process at the same time in parallel such that the queue time could be removed. However, the parallelism technique is required supported devices in order to achieve the speedup. In [19], shared memory parallelism is used which required a Symmetric multiprocessing (SMP) machine to perform parallelism processing. Besides, a CPU with parallelism ability is required to perform parallel process [20]. However, the speedup offered by using parallelism is not significant even more resources is required. In the work proposed in [19], there is able to achieved 1.5~1.9 times speedup on the query process.

The another speedup is tree optimization. This technique is optimizing the tree by reduce the complexity. The tree optimization is usually being done during the construction of tree. The optimization algorithms are used to achieve the purpose of reducing the volume of leaf nodes. There are several optimization algorithms has been developed such as slim-down algorithm [21] and space partitioning [18]. The slim-down algorithm is using a post-processing technique to redistributed the ground objects between leaf-nodes. Space partitioning is partition the sub-sets of the M-tree. By using the tree optimization, the query process could be speedup by approximately 100% with the reduced volume [22, 18].

However, there is not work has been done to speedup the M-tree algorithm using hardware accelerator. This is possible due to the existing M-tree code was relied on the object structure offer by object oriented programming which is much more easy to coded in software. Furthermore, most of current applications that using M-tree are running in software.

2.4 Chapter Summary

In summary, the M-tree query algorithm has been discussed including the basic concept, data structure, and operation. Besides, the analytical prove of the M-tree query algorithm is explained. In addition, the pseudo code of M-tree range queries has been discussed. The related works on the M-tree algorithm has been discussed in this section. The related work on the performance of M-tree algorithm has been reviewed. Furthermore, the existing work on speedup for M-tree queries algorithm also being discussed.

CHAPTER 3

METHODOLOGY

3.1 Proposed work

The proposed work is implementing the fast query into a hardware accelerator. The top level view of the hardware accelerator is shown in Figure 3.1. The hardware accelerator will required two inputs including the data set in M-tree data structure, queries and queries range. Then, the output of the hardware accelerator is the matched results.

The M-tree query algorithm is being studied and translated into System Verilog. In order to develop the hardware accelerator flawlessly, pre-work has been done including ASM chart and functional block diagram. After that, the hardware accelerator is being coded in Xilinx's Vivado Design Suite uisng System Verilog Hardware Description Language (HDL). The entire system of hardware accelerator including the memory and FIFO modules are being coded without using the Intellectual property (IP) provided by Xilinx. The purpose of not using the IP provided by Xilinx is to develop the hardware accelerator in generic such that there is able to implement in other tools such as Altera's Quartus.

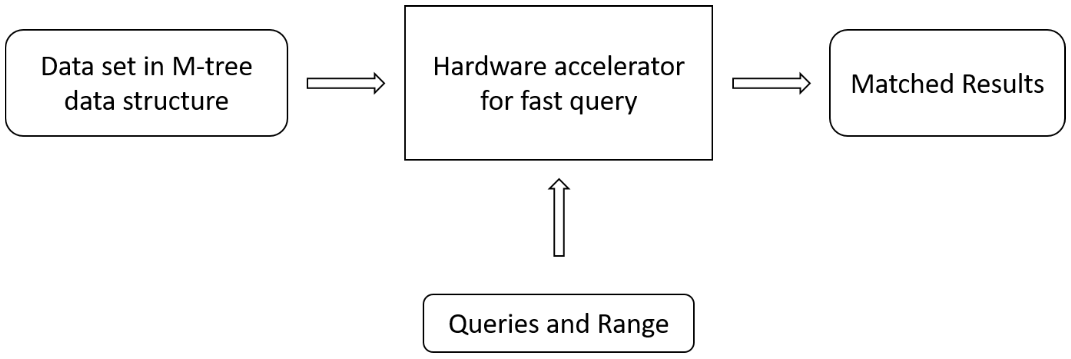

Data set in M-tree data structure
Hardware accelerator for fast query
Matched Results
Queries and Range

Figure 3.1: Top level view of the hardware accelerator

3.1.1 ASM Chart

Before designing the hardware accelerator architecture, the ASM chart has been drawn to provide a overview of the system flow as shown in Figure 3.2. The ASM chart shows the state of the control unit for the entire query flow. Besides, the required internal signal to control the data unit has been listed for each state in the ASM chart.

3.1.2 Functional block diagram

The functional block diagram of the hardware accelerator has been drawn as shown in the Figure 3.3. The main components of the hardware accelerate is the control unit (CU) and datapath unit (DU). Control unit is the unit to control the datapath to perform the tasks. The datapath unit is the unit that perform the operation of the tasks. The range search unit is located inside the datapath unit which perform the condition checking for the query.

According to the M-tree structure, there is needed to have 3 memory modules to store information of nodes, routing objects, and objects. M-tree query algorithm is calling the function while inside the function itself to continue the search for the next non-leaf node until the reaching the leaf node. This method is difficult to implement exactly the same into the hardware. Thus, the alternative way is using the queuing mechanism where First-in-First-Out (FIFO) module is needed.

3.1.3 Data structure

For the M-tree in software which is written using object oriented programming, the data of nodes, routing objects and objects are store as a object. In order to using M-tree algorithm in hardware, the data structure in the hardware accelerator is being designed to similar to the data structure shown in 2.3. The address of the nodes, routing objects and the object is being used as the pointer. With the node capacity limit of two, the node data is storing the address of two routing objects as shown in Figure 3.4. Besides, the data of routing object is consist of object value, covering radius, distance to parent, and the address of node as shown in Figure 3.5. The object memory is storing the actual object data as shown in 3.6.

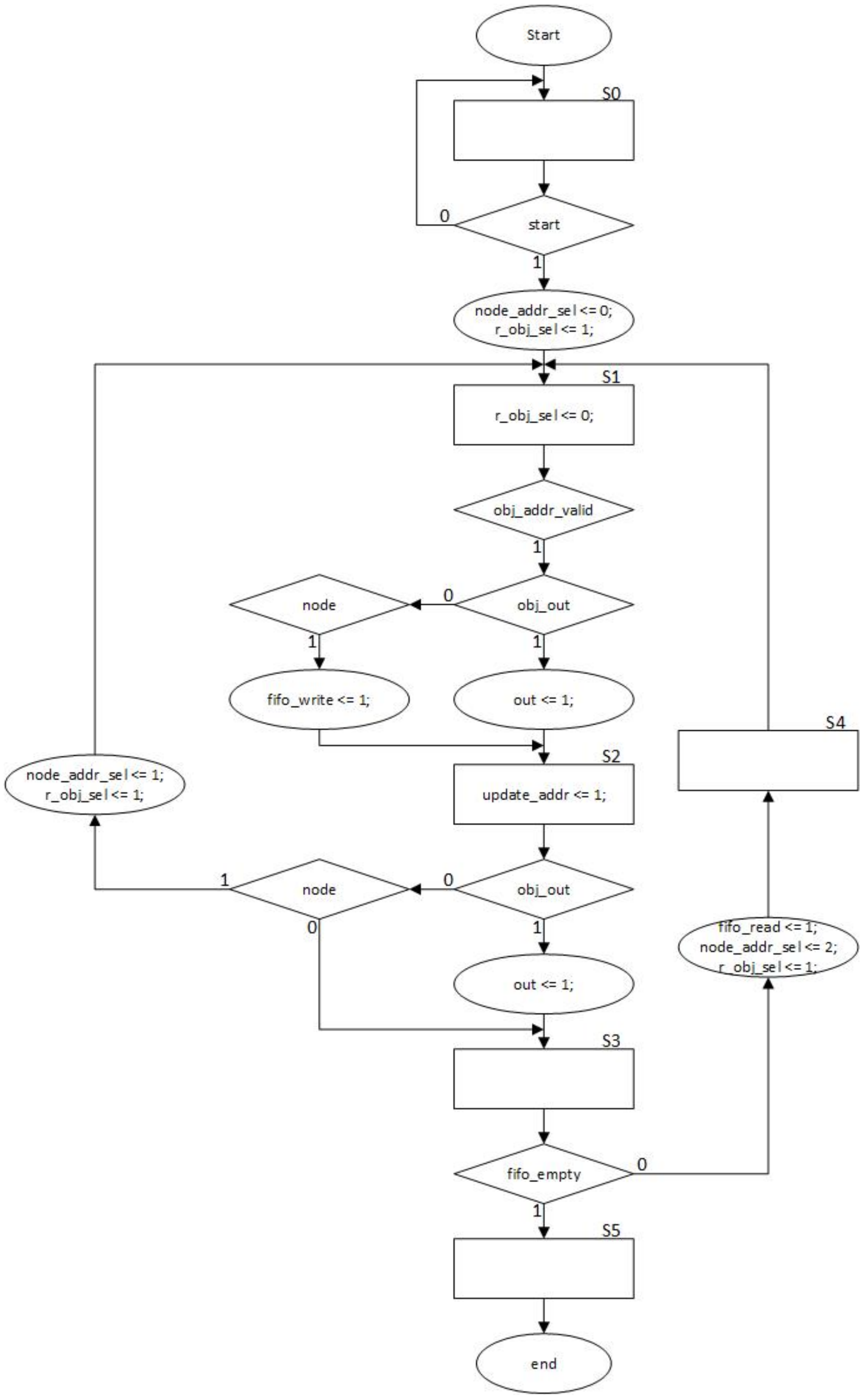


Figure 3.2: ASM chart

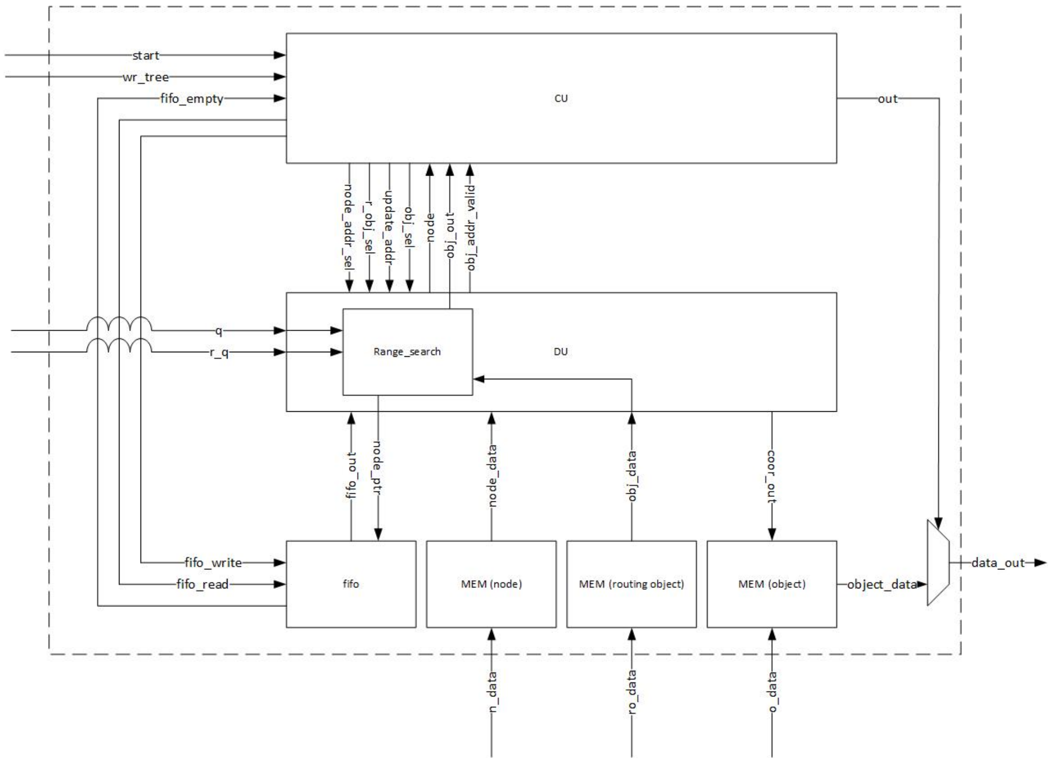


Figure 3.3: Functional block diagram of hardware accelerator

MSB LSB

Address of routing object 2	Address of routing object 1

Figure 3.4: Data of nodes in memory module

MSB LSB

Object	Covering radius	Distance to parent	Address of node

Figure 3.5: Data of routing object in memory module

MSB LSB

Actual object data

Figure 3.6: Data of object in memory module

3.1.4 Operation

Before the query start, the data set in the form of M-tree data structure is needed to be input into the system memory. By assert the wr_tree signal, CU will trigger the write mode and will write the data from the bus n_data, ro_data, and o_data into the memory modules. Once the M-tree data set is being store into the memory, the system is ready to perform the query.

To perform the query, data need to be provide to the bus q and r_q which is the query object and the query object range respectively. The query process will be start is the start signal is assert. The hardware accelerator will start the searching from the top level of M-tree. First node address will be passing to the DU and the node data will be read from the node memory module. Since each node data is consists of two routing object address, the search is done in two states. For the first routing object address, the DU read and passing the first routing object data to the range search and perform the condition checking. Address of the node will be store into the FIFO if the routing object is meeting the searching criteria. At next state, DU will read the second routing object data. In this state, the node will become the next search if the node address is successfully returned by the range search module. During the two searching state, the range search module will assert obj_out signal to the CU if any object is found and matching to the searching criteria. When CU receive the obj_out signal from range search module, an out signal is asserted and the hardware accelerator will output the actual object data. This two states will continue until the the second state does not return any node address. Then, the system will check on the FIFO buffer. If the FIFO is not empty and having the node address in queue, the system will read the node address and repeating the two search state until the FIFO buffer is empty. Finally the query flow is end and a done signal is asserted.

3.1.5 Synthesized design

The designed hardware accelerator has been synthesized using Xilinx's Vivado Design Suite. The synthesis is done based on the evaluation board with the model name of ZYNQ-7 ZC702 which is developed by Xilinx. The tool able to generate and view the elaborated design which is the netlist view of the design. Besides, the details reports are also generated such as the utilization and the power consumption.

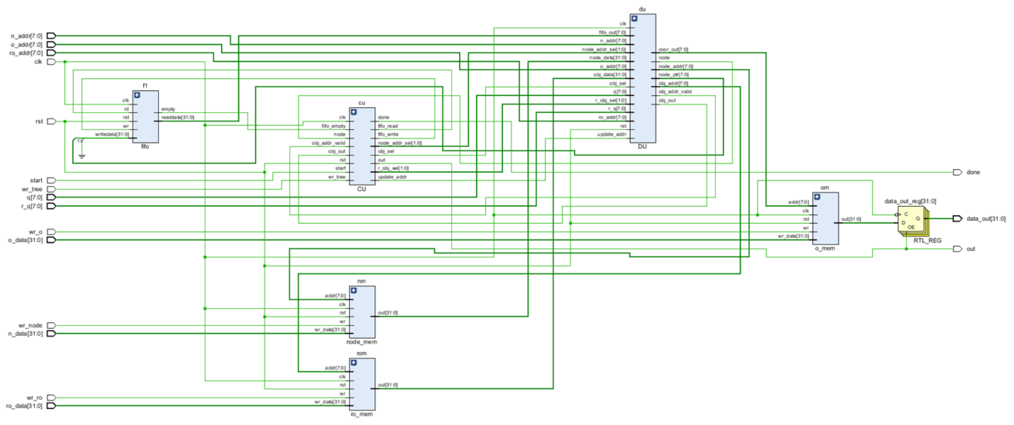
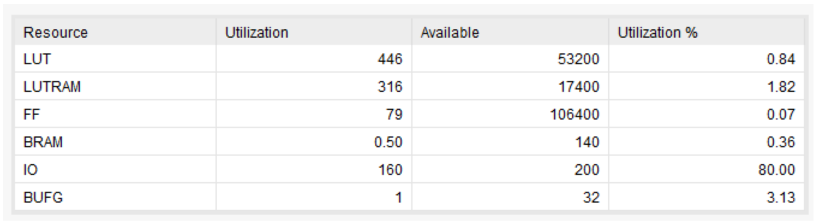


Figure 3.7: Elaborated design



Resource	Utilization	Available	Utilization %
LUT	446	53200	0.84
LUTRAM	316	17400	1.82
FF	79	106400	0.07
BRAM	0.50	140	0.36
IO	160	200	80.00
BUFG	1	32	3.13

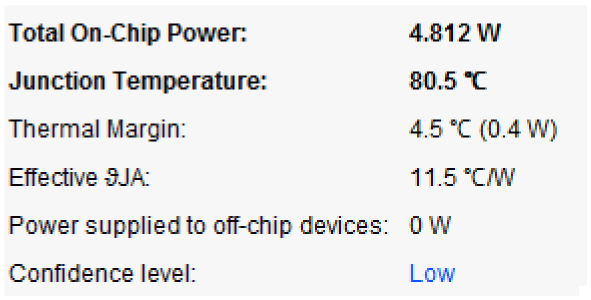
Figure 3.8: Utilization summary report

3.1.5.1 Elaborated design

An elaborated design of the hardware accelerator has been generated as shown in the Figure 3.7. The elaborated design is generated by Xilinx's Vivado design Suite which is based on the written System Verilog code. The generated elaborated design is desired as it is similar to the functional block diagram which has been drawn in Figure 3.3.

3.1.5.2 Utilization

The summary of the utilization of the designed hardware accelerator is shown in Figure 3.8. From the report, there is only little resources are being used from the evaluation board expect the IO. The utilization of LUT, LUTRAM, FF, BRAM, and BUFG are 0.84%, 1.82%, 0.07%, 0.36%, and 3.13% respectively.The IO utilization is large due to the input bus of memory data which are n_data, ro_data, and o_data. This could be further optimize by changing the reading method to bit by bit. However, the overall utilization of the designed hardware accelerator is good.



Total On-Chip Power:	**4.812 W**
Junction Temperature:	**80.5 °C**
Thermal Margin:	4.5 °C (0.4 W)
Effective ϑJA:	11.5 °C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Figure 3.9: Power report

3.1.5.3 Power consumption

The power report also been generated by the Xilinx's Vivado Design Suite after synthesis as shown in Figure 3.9. The estimated total on-chip power is 4.812 W where the power consumption low. Hence, this is possible to implement the hardware accelerator on the small device that require low power. Besides, the estimated junction temperate is 80.5 °C which is the highest operating temperate that could introduced by the device.

3.2 Experimental setup

The hardware accelerator is implemented using the tool named Vivado Design Suite by Xilinx. Xilinx's Vivado Design Suite is a powerful tool that allow the user to design, synthesize, and simulate the hardware design. Besides, the design of the hardware accelerator is is targeted on FPGA platform. This is because FPGA allow the user to design, develop, and test the prototype of integrated circuit without fabricate the chip.

The hardware accelerator is coded using System Verilog Hardware Description Language (HDL). Besides, A test bench has been coded in System Verilog to validate the functionality of the designed hardware accelerator. The designed hardware accelerator is being synthesize and simulated using Xilinx's Vivado Design Suite. The simulation is done using the test bench and the performance of the hardware accelerator is being evaluate. Since different devices offer different clock speed, so the measurement will be happen on the number of clock cycles needed to perform the query. Hence, the total number of clock cycles for the hardware accelerator to perform a query is observed to evaluate the performance. In addition, the total query time could be calculated by multiply the clock speed with the total number of clock cycles.

In order to test the designed hardware accelerator, one dimensional fixed point numerical data set is used. The data set is pre-proccessed into M-tree data structure using software and then being used as the input for the designed hardware accelerator.

3.3 Chapter Summary

In summary, this chapter described the research methodology for the proposed work. The work done while develop the hardware architecture has been shown. Besides, overview, flow, and operation of the proposed solution is discussed. The implemented design has been synthesized and the reports are shown and discussed. Besides, the experimental setup such as the tools, platforms, input, measurement metric, and the validation setup also covered in this chapter.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Results of proposed work

The developed hardware accelerator is being simulated using Xilinx's Vivado Design Suite. The test bench has been coded to perform the simulation. In the test bench the clock speed is set to 100 MHz which having the period of 10ns.

4.1.1 Simulation results

The simulation is done with different query condition including the single query and query with range. A simple data set is used in the simulation which is consist of 1,2,3,4,5,6,7,8,21,22,23,24,25,26,27. The simulated results for query with range is shown in Figure 4.1. From the Figure 4.1, the query object is 5 with the range of 1 and the hardware accelerator is successfully output the correct query data that met the criteria. Figure 4.2 shows the simulated result with single query. The query object is 22 in Figure 4.2 and the hardware accelerator is able to return a correct query data. Hence, the functionality of the hardware accelerator is correct as expected results are obtained. Besides, the simulation also shown that the hardware accelerator is able to process one routing object with 1 clock cycle only.

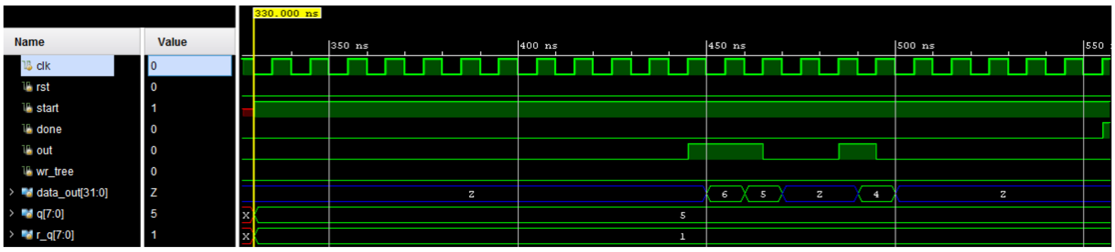


Figure 4.1: Simulation results for query with range

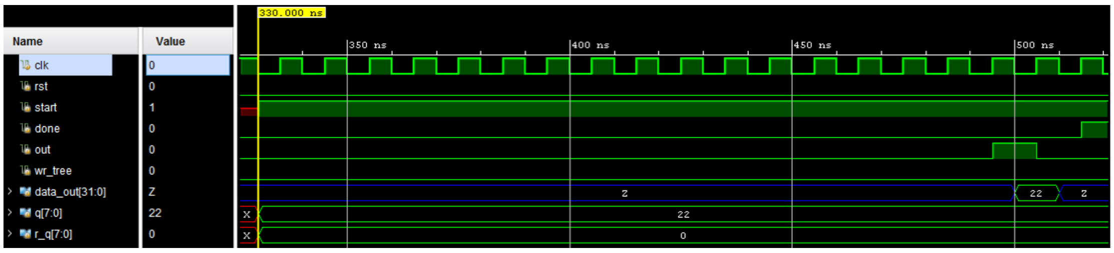


Figure 4.2: Simulation result for single query

4.1.2 Performance evaluation

The performance of the hardware accelerator is being evaluated. As the time scale for different devices will be different, number of clock cycle is used for performance evaluation. The graph of number of query data vs number of clock cycle has been plotted as shown in Figure 4.3. From the Figure 4.3, there could be observe that the total query time is increasing as the number of the query data increase. This is showing the hypothesis that the total query time is proportional to the number of query.

On the other side, the graph of level of tree vs number of clock cycle has been plotted as shown in Figure 4.4. Different level of tree also represent the different database size where larger the database size, larger the level of tree. In Figure 4.4, there is also observed that the total query time is increasing when the level of tree is larger. This is showing the hypothesis that the total query time is linearly proportional to the database size.

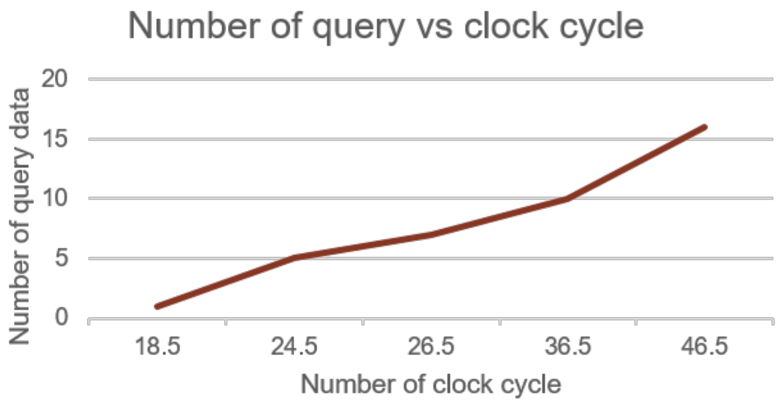


Figure 4.3: Number of query data vs number of clock cycle

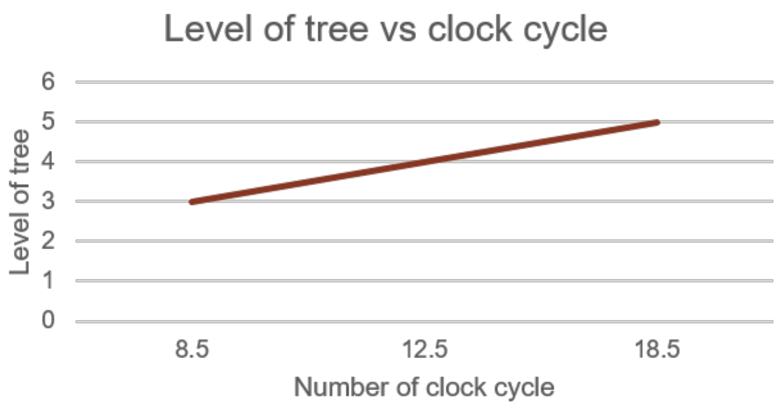
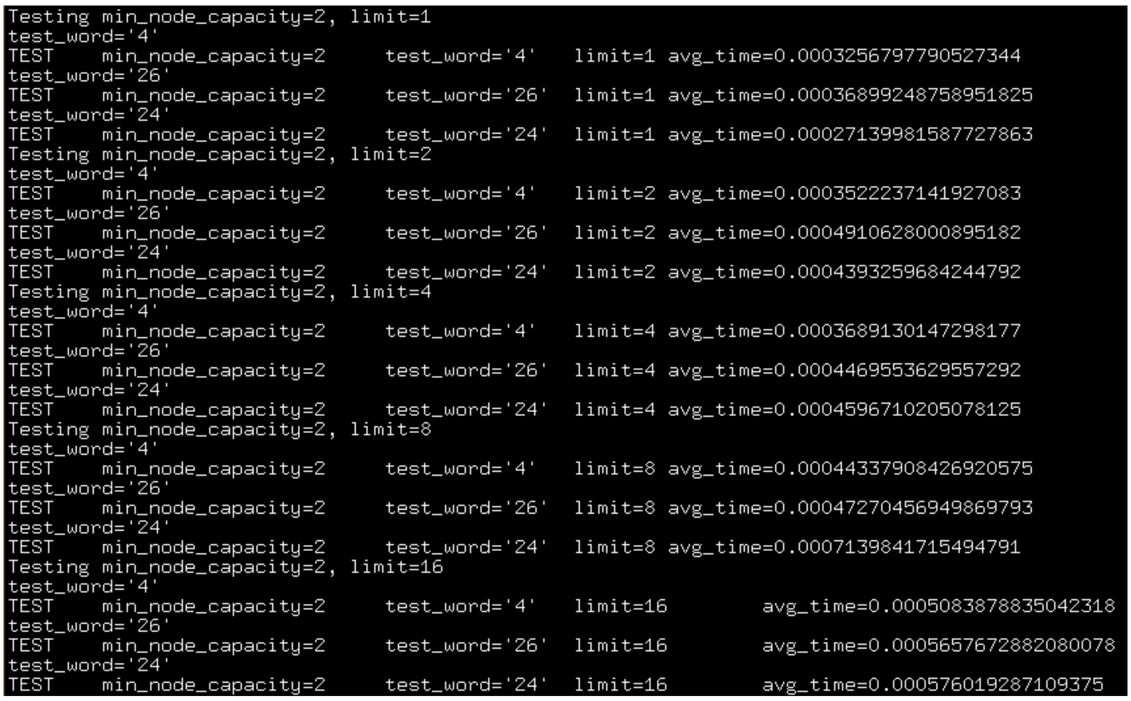


Figure 4.4: Level of tree vs number of clock cycle

4.2 Results Comparison

In order to verify value of the designed hardware acceleration, the comparison with the M-tree in software is done. The M-tree in software is written in python and developed by Paolo Ciaccia et al. which has been presented in their paper [23]. The M-tree software is running in a work station system equipped with Intel(R) Xeon(R) CPU E5-2620 v3 with the clock speed of 2.40GHz. Same data set has been used as the input for the M-Tree in software to achieve apple-to-apple comparison. In addition, the node capacity is limited to two such that it is similar to the designed hardware accelerator. Figure 4.5 shows the results of the M-Tree running in software for different number of query data. Three test cases are carried out by the M-tree in software and total query time for each test case is displayed. Besides, the M-tree software also doing the same for different level of tree.

The results are being tabulated in Table 4.1 and Table 4.2. The speedup of the hardware accelerator compare to the software also being calculated. The speedup of the query is more than 1000 times. The speedup is significant which could be due to several reason. The possible reason is that the general purpose CPU is loaded with OS and running multiple background programs that occupied the CPU bandwidth. By checking the system's load information as shown in Figure 4.6, the CPU is being loaded at 82%. This situation has best demostrated one of the problem statement of this project where performance of query using software is relied on computation unit [12]. Besides, the object oriented programming is relatively slow as its consuming more resource [24]. In addition, the hardware accelerator is simulated in a ideal condition with the clock speed of 100 MHz.



```
Testing min_node_capacity=2, limit=1
test_word='4'
TEST    min_node_capacity=2     test_word='4'   limit=1 avg_time=0.0003256797790527344
test_word='26'
TEST    min_node_capacity=2     test_word='26'  limit=1 avg_time=0.00036899248758951825
test_word='24'
TEST    min_node_capacity=2     test_word='24'  limit=1 avg_time=0.00027139981587727863
Testing min_node_capacity=2, limit=2
test_word='4'
TEST    min_node_capacity=2     test_word='4'   limit=2 avg_time=0.0003522237141927083
test_word='26'
TEST    min_node_capacity=2     test_word='26'  limit=2 avg_time=0.0004910628000895182
test_word='24'
TEST    min_node_capacity=2     test_word='24'  limit=2 avg_time=0.0004393259684244792
Testing min_node_capacity=2, limit=4
test_word='4'
TEST    min_node_capacity=2     test_word='4'   limit=4 avg_time=0.0003689130147298177
test_word='26'
TEST    min_node_capacity=2     test_word='26'  limit=4 avg_time=0.0004469553629557292
test_word='24'
TEST    min_node_capacity=2     test_word='24'  limit=4 avg_time=0.0004596710205078125
Testing min_node_capacity=2, limit=8
test_word='4'
TEST    min_node_capacity=2     test_word='4'   limit=8 avg_time=0.00044337908426920575
test_word='26'
TEST    min_node_capacity=2     test_word='26'  limit=8 avg_time=0.00047270456949869793
test_word='24'
TEST    min_node_capacity=2     test_word='24'  limit=8 avg_time=0.0007139841715494791
Testing min_node_capacity=2, limit=16
test_word='4'
TEST    min_node_capacity=2     test_word='4'   limit=16        avg_time=0.0005083878835042318
test_word='26'
TEST    min_node_capacity=2     test_word='26'  limit=16        avg_time=0.0005657672882080078
test_word='24'
TEST    min_node_capacity=2     test_word='24'  limit=16        avg_time=0.000576019287109375
```

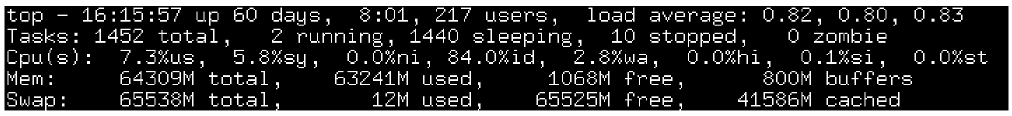
Figure 4.5: Results of the M-tree query using software

Number of query data	M-tree in Software	M-tree in hardware accelerator	Speedup
1	0.326ms	185ns	1762x
4	0.369ms	245ns	1506x
8	0.443ms	315ns	1406x
16	0.508ms	465ns	1092x

Table 4.1: Tabulated comparison results for number of query data

Level of tree	M-tree in Software	M-tree in hardware accelerator	Speedup
3	0.196ms	85ns	2305x
4	0.233ms	125ns	1864x
5	0.481ms	185ns	2600x

Table 4.2: Tabulated comparison results for level of tree



```
top - 16:15:57 up 60 days,  8:01, 217 users,  load average: 0.82, 0.80, 0.83
Tasks: 1452 total,   2 running, 1440 sleeping,  10 stopped,   0 zombie
Cpu(s):  7.3%us,  5.8%sy,  0.0%ni, 84.0%id,  2.8%wa,  0.0%hi,  0.1%si,  0.0%st
Mem:     64309M total,   63241M used,    1068M free,      800M buffers
Swap:    65538M total,      12M used,   65525M free,    41586M cached
```

Figure 4.6: Summary of system’s load inforamtion

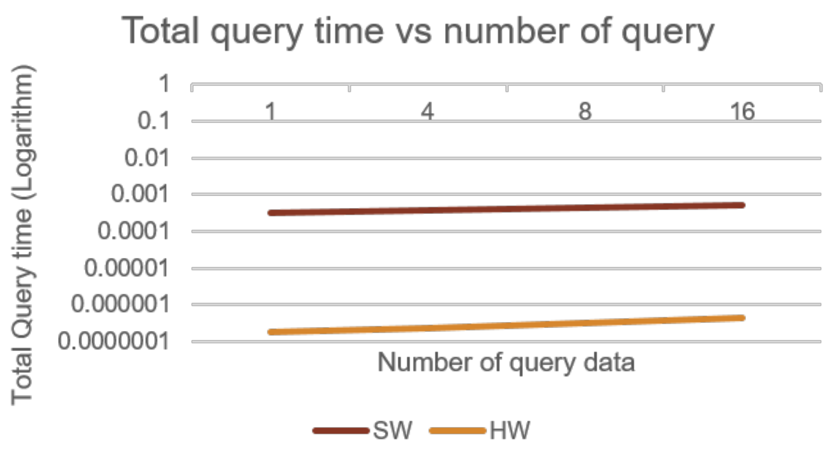


Figure 4.7: Comparison of HW and SW for number of query data

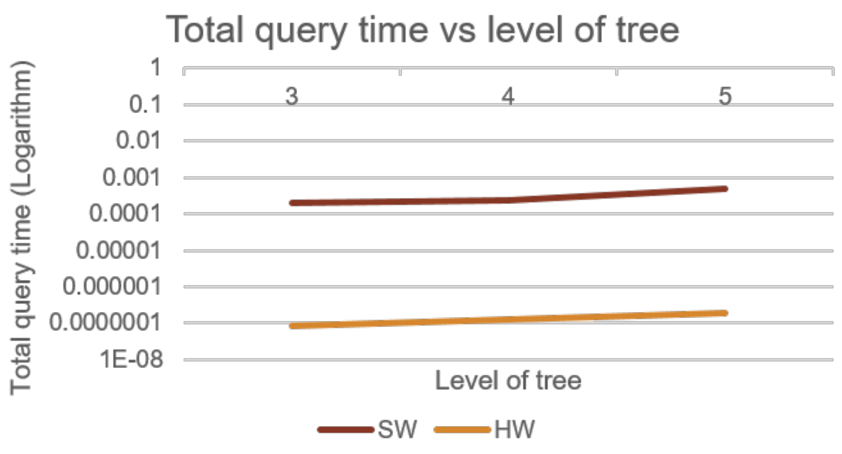


Figure 4.8: Comparison of HW and SW for level of tree

To have better view of the speedup offer by the hardware accelerator compare to software, a comparison graph has been plotted as shown in Figure 4.7 and Figure 4.8. Figure 4.7 shows the differences of M-tree running in hardware ans software for total query time against number of query. Figure 4.8 shows the differences of M-tree running in hardware ans software for total query time against level of tree. In both plots, there could be observed that hardware accelerator is having better performance compare to the software where the total query time is shorter. Thus, the proposed hardware accelerator is successfully provide the acceleration on the performance toward the software.

4.3 Chapter Summary

In summary, this chapter discussed on results obtained from the simulation. The functionality of the hardware accelerator has been verified which is able to perform the correct operation. Besides, the performance of the hardware accelerator has been evaluated. In order to obtain the speedup, comparison between existing works has been done. The hardware accelerator is able to provide significant speedup compare to M-tree algorithm running in software. The discussion on the speedup also being done in this chapter.

CHAPTER 5

CONCLUSION

5.1 Project accomplishment of objectives

Fast query is important for current era as data retrieval rate could affect the performance of devices. With the rapid growth of database size, the performance of the devices will be impacted. Thus, hardware accelerator for fast query is a way to maintain the performance of the devices for current trend. In this project, a hardware based accelerator for fast database query has been proposed. The proposed work has demonstrated that query process could be accelerate by hardware accelerator. The acceleration are observed for both situation including the Besides, the functionality of the hardware accelerator has been verified that could perform the same with the software. Furthermore, the hardware accelerator could be implement for varies devices as it is a separate hardware unit which does not rely on a general-purpose CPU. Hence, the hardware accelerator is well suit for the IOT devices. In conclusion, the objectives of the project are achieved.

5.2 Future Works

There is several works could be done in the future to improve the current proposed hardware accelerator. One of the future works is the improvement on the hardware accelerator to allow customization of node capacity. In current proposed hardware accelerator, the node capacity has been limited to two which will only have two pointer for each node. This will causing the the level of tree become larger if the database size is increased. By allow the ability to customize the node capacity, the M-tree could be wider and much optimized for large database.

The second future work is the improvement for the hardware accelerator to support for multi-dimensional data set. The data set used in the proposed work is fixed to one dimensional fix point data set. In order to fully utilize the feature of M-tree in handling muti-dimensional data, different distance function could be develop and implement into the hardware accelerator. By having different distance function unit in the hardware accelerator, user could use control signal to switch between different distance functions to select the distance function based on the needs.

Besides, another future works is implementing the M-tree build into the hardware accelerator. In current project, the scope is focusing on the fast query such that there is require a ready built M-tree data as input. By implement the M-tree build function into the hardware accelerator, there is allow the input of raw data into the hardware accelerator.

REFERENCES

1. insideBIGDATA. *The Intelligent Use of Big Data on an Industrial Scale*. Technical report. 2017.

2. Guhlemann, S., Petersohn, U. and Meyer-Wegener, K. Reducing the Distance Calculations when Searching an M-Tree. *Datenbank-Spektrum*, 2017. 17(2): 155–167. ISSN 1610-1995. doi:10.1007/s13222-017-0258-5. URL `https://doi.org/10.1007/s13222-017-0258-5`.

3. Zezula, P., Amato, G., Dohnal, V. and Batko, M. *Similarity Search: The Metric Space Approach*. Advances in Database Systems. Springer US. 2010. ISBN 9781441939722. URL `https://books.google.com.sg/books?id=6AdZcgAACAAJ`.

4. Ciaccia, P., Patella, M., Rabitti, F. and Zezula, P. Indexing Metric Spaces with M-tree. *PROC. QUINTO CONVEGNO NAZIONALE SEBD*. 1997. 67–86.

5. Perrone, M. P. Keynote speaker I: Big data transforming industries. *2015 11th International Conference on Innovations in Information Technology (IIT)*. 2015. XXVII–XXVII. doi:10.1109/INNOVATIONS.2015.7381493.

6. Team, E. The Exponential Growth of Data, 2017. URL `https://insidebigdata.com/2017/02/16/the-exponential-growth-of-data`.

7. Cisco. *Global Cloud Index: Forecast and Methodology, 2015-2020*. Technical report. 2017.

8. Nguyen, X.-T., Nguyen, H.-T. and Pham, C.-K. An FPGA approach for fast bitmap indexing. *IEICE Electronics Express*, 2016. 13(4): 20160006–20160006. doi:10.1587/elex.13.20160006.

9. Samuel, A. L. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, 1959. 3(3): 210–229. ISSN 0018-8646. doi:10.1147/rd.33.0210.

10. Ozturk, O. and Ferhatosmanoglu, H. Effective indexing and filtering for similarity search in large biosequence databases. *Third IEEE Symposium on*

Bioinformatics and Bioengineering, 2003. Proceedings. 2003. 359–366. doi:10.1109/BIBE.2003.1188974.

11. Ghosh, E., Ohrimenko, O. and Tamassia, R. Zero-Knowledge Authenticated Order Queries and Order Statistics on a List. Malkin, T., Kolesnikov, V., Lewko, A. B. and Polychronakis, M., eds. *Applied Cryptography and Network Security*. Cham: Springer International Publishing. 2015. ISBN 978-3-319-28166-7. 149–171.

12. Nakanishi, K., Hochin, T. and Nomiya, H. Evaluation of Parallel Multi-Dimensional Indexing System for Big Data Analysis. *2016 4th Intl Conf on Applied Computing and Information Technology/3rd Intl Conf on Computational Science/Intelligence and Applied Informatics/1st Intl Conf on Big Data, Cloud Computing, Data Science Engineering (ACIT-CSII-BCD)*. 2016. 105–110. doi:10.1109/ACIT-CSII-BCD.2016.031.

13. Graefe, G. and Kuno, H. Modern B-tree techniques. *2011 IEEE 27th International Conference on Data Engineering*. 2011. ISSN 1063-6382. 1370–1373. doi:10.1109/ICDE.2011.5767956.

14. Li, H., Ju, S. and Chen, W. Design and Implementation of Generalized R-Tree. *2008 International Symposium on Computer Science and Computational Technology*. 2008, vol. 1. 777–781. doi:10.1109/ISCSCT.2008.317.

15. Schubert, E., Zimek, A. and Kriegel, H.-P. Geodetic Distance Queries on R-Trees for Indexing Geographic Data. Nascimento, M. A., Sellis, T., Cheng, R., Sander, J., Zheng, Y., Kriegel, H.-P., Renz, M. and Sengstock, C., eds. *Advances in Spatial and Temporal Databases*. Berlin, Heidelberg: Springer Berlin Heidelberg. 2013. ISBN 978-3-642-40235-7. 146–164.

16. Viet, H. H. and Anh, D. T. M-tree as an index structure for time series data. *2013 International Conference on Computing, Management and Telecommunications (ComManTel)*. 2013. 146–151. doi:10.1109/ComManTel.2013.6482381.

17. Shaw, K., Ioup, E., Sample, J., Abdelguerfi, M. and Tabone, O. Efficient Approximation of Spatial Network Queries using the M-Tree with Road Network Embedding. *19th International Conference on Scientific and Statistical Database Management (SSDBM 2007)*. 2007. ISSN 1551-6393. 11–11. doi:10.1109/SSDBM.2007.11.

18. Zhou, X., Wang, G., Yu, J. X. and Yu, G. M+-tree: A New Dynamical Multidimensional Index for Metric Spaces. *Proceedings of the 14th Australasian Database Conference - Volume 17*. Darlinghurst, Australia,

Australia: Australian Computer Society, Inc. 2003, ADC ’03. ISBN 0-909-92595-X. 161–168. URL http://dl.acm.org/citation.cfm?id=820085.820118.

19. Qiu, C., Lu, Y., Gao, P., Wang, J. and Lv, R. A Shared Memory Parallel k-NN Query Algorithm for M-tree. *2009 International Conference on Management and Service Science*. 2009. 1–4. doi:10.1109/ICMSS.2009.5305620.

20. Zezula, P., Savino, P., Rabitti, F., Amato, G. and Ciaccia, P. Processing M-trees with parallel resources. *Proceedings Eighth International Workshop on Research Issues in Data Engineering. Continuous-Media Databases and Applications*. 1998. 147–154. doi:10.1109/RIDE.1998.658289.

21. Traina, C., Traina, A., Seeger, B. and Faloutsos, C. Slim-Trees: High Performance Metric Trees Minimizing Overlap between Nodes. Zaniolo, C., Lockemann, P. C., Scholl, M. H. and Grust, T., eds. *Advances in Database Technology — EDBT 2000*. Berlin, Heidelberg: Springer Berlin Heidelberg. 2000. ISBN 978-3-540-46439-6. 51–65.

22. Skopal, T., Pokorný, J., Krátký, M. and Snášel, V. Revisiting M-Tree Building Principles. Kalinichenko, L., Manthey, R., Thalheim, B. and Wloka, U., eds. *Advances in Databases and Information Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg. 2003. ISBN 978-3-540-39403-7. 148–162.

23. Ciaccia, P., Patella, M. and Zezula, P. M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. *Proceedings of the 23rd International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. 1997, VLDB ’97. ISBN 1-55860-470-7. 426–435.

24. Popyack, J. L. Introduction to Computer Science Object Oriented Programming: Disadvantages of OOP. https://www.cs.drexel.edu/~introcs/Fa12/notes/06.1_OOP/Disadvantages.html?CurrentSlide=2, 2012.