Models of computation: A numeric analysis and performance evaluation

Ricardo B. Verschueren Bcs (Hons)

University of Gloucestershire

Department of Computer science and Information Technology

Faculty of Media, Art and Technology

For the fulfilment of a Masters by research degree

August 2014

Abstract

This research seeks to better understand what drives performance in computation. To develop this understanding the researcher investigates the literature on computational performance within the classical and quantum paradigm for both binary and multi-value logic. Based on the findings of the literature the researcher evaluates through an experiment of addition what drives performance and how performance can be improved. For the evaluation of this research, a realist research paradigm employs two research methods. The first is an automaton model of computation to model each of the computing paradigms and computational logic. The second is computational complexity theory for measuring the performance of addition. Through this evaluation the researcher seeks to gain a better understanding of what drives computational performance and how addition can be performed more efficiently.

The results of the research lead the researcher to conclude that modernisation of machinery caused the birth start of automated computing and the binary number system in computers. As this research indicated that computation through increasing the radix can improve performance of computation for addition. Based on reported findings in the science of quantum mechanics research, it would be possible to implement a model of computation with increased radix. Through embracing state discrimination/ distinguishability this research calls to review the current quantum computing paradigm based on state duality.

Declaration

I declare that the work in this thesis was carried out in accordance with the regulations of the University of Gloucestershire and is original except where indicated by specific reference in the text. No part of the thesis has been submitted as part of any other academic award. The thesis has not been presented to any other education institution in the United Kingdom or overseas.

Any views expressed in the thesis are those of the author and in no way represent those of the University.

Ricardo Verschueren

2014

Acknowledgement

In the name of the lord the greatest, most merciful and sustainer of the heavens and earth this research was completed. With the hope that it may be recognised on the day of judgement as not all my time could be dedicated to prayer. I ask for this research to be accepted as an act of worship.

In sincerity this research was completed as an act of worship to which I have been true over a period of four years. With intent of accomplishing every detail worshipfully, and therefore urge readers to read this research as though its an act of worship to gain the full extent of the research.

The Author

Ricardo B. Verschueren, for Masters by Research degree in Computer Science, presented for submission on August 2014, at The University of Gloucestershire.

Title: Models of computation: A numeric analysis and performance evaluation

First supervisor: Dr. Hapeshi

Second supervisor: Dr. Jayal

Contents

List of illustrations

List of tables

Glossary

Abacus	Instrument through which arithmetic can be performed.
Abstraction	Within the context to the research abstractions refers to the model of computation as an abstraction of how computer function.
Academics	Academic in this thesis refer to scholars that have contributed to the body of knowledge.
Adder	An electronic circuit through which addition can be performed.
Adiabatic Computing	Computer circuit through which the amount of inputs are equal to the outputs.
Algorithm	A set of computer instructions through which an computational task can be achieved.
Approaches to Addition	Within the context of this research a general term is used for the different types of adders.
Architecture	Description of how the components within a computational system inter-relate to each other.
Automata Model of Computation	Model of computation used within this research to describe a computation.
Binary	Number base system based on one and zero.
Bit	Most fundamental unite within a classical computational system.
Boolean Logic	Computational logic with classical computing.
Carry	Term used for adders when an overflow of the number base

	system occurs.
Circuit	Model of computation through which a computation is implemented.
Classical Computation	Computation based on classical physics.
Computation	Process of performing a series of calculations through machine instructions with the purpose of attaining an outcome.
Computational Complexity Theory	Method for classifying the amount of resources required by a model of computation to perform a computational task.
Computational Logic	The logic I.E. Boolean logic, used by a computation.
Computational Model	Abstraction of how a machine performs computation.
Computational Performance	The rate at which a computation is completed by a predefined model of computation measure in unites of time.
Cryptography	Computational task that encodes information.
Decimal	Number base system based on ten values.
Decoherence	Quantum phenomena through which is the state of Qbits change by interaction with the environment.
Deduction	A form of logic through which reason is used to form conclusions.
Duality	Quantum phenomena through which two states can be identified within a single Qbit.
Entanglement	Quantum phenomena through which multiple Qbits are joined to form a complex state.
Epistemology	Description of what is considered knowledge within this research and how knowledge is ascertained.
Everettian	Quantum mechanics interpretation.

Exponential	Computational complexity class describing growth rate of computational instructions.
Fourier Transform	Mapping of a function as a signal which is used to perform a computation.
Halting Problem	Computer science theory that tries to determine whether an algorithm will come to an end.
Lambda Calculus	Computational model based on calculus.
Linear	Computational complexity class describing growth rate of computational instructions.
Logarithmic	Computational complexity class describing growth rate of computational instructions.
Logic Gates	Computation operations based on Boolean logic.
Many Worlds Interpretation	Quantum mechanics interpretation.
Quantum Measurement	Quantum science of determining the state of a Qbit.
Methodology	Description of methods used to perform research.
Modulo	Arithmetic operation of division resulting in the remainder
Multi Value Logic	For of logic based on a higher number based.
Multipliers	Computational circuit for performing multiplication.
Ontology	Description of the domain of discourse under investigation.
Paradigm Shift	Change in the mainstream view of the research that affects the underlying fundamental principles.
Parallelism	Computing thesis stating that multiple computational tasks can be performed concurrently.

Polynomial	Computational complexity class describing growth rate of computational instructions.
Probabilistic	Algorithm through which the successful completion of a computational task is based on probability
Quantum Computing	Computational paradigm based on quantum physics.
Quantum Phenomena	Particular characteristic unique to quantum physics.
Quaternary Logic	Computational logic with number base four.
Qubits	Most fundamental unite within a quantum computational system.
State Discrimination/ Distinguishability	Quantum science that seeks to find how many different states can be identified within a single Qbit.
Ternary Logic	Computational logic with number base three.
Truth Tables	Listing of all possible combination of a logic gate.
Turing Model	Universal model of computation.

Chapter 1 Introduction

Within academic research scholars have studied algorithms with the purpose of defining a finite set of instructions that can solve a particular problem. As a subset of this research, the problems found are unsolvable while others may be solvable in theory. In practice the algorithm is not feasible as the level of computational complexity is too high. The algorithms that are solvable in theory are classified according to computational complexity theory as hard to solve or of high complexity. Within this research the fundamental principles that underlie a hard to compute problem is investigated. This is with the purpose of being able to solve such algorithms both in practice and in theory.

This research investigates the fundamental principles of computation through understanding the underlying model of computation. The revision of models of computation is performed, paradigms and different types of logic. By understanding how each of the different models of computation function and perform computation, the research aims to determine what the driving force of computational performance. By comparing and contrasting different models of computation the computational performance difference should be visible. Based on those findings, with regard solving high complexity algorithms in practice can be speculated.

The basic arithmetic is not of high complexity and easily can be performed by computers. Basic arithmetic is the most often performed operation for a general purpose computer, in particular addition. This is because through addition other arithmetic operations can be derived through ‘two’s complement’ computation can do subtraction, and multiple additions lead to multiplication. As basic arithmetic is such a fundamental and integral part of how a computer operates. This research uses addition to evaluate the performance of the different models of computation identified.

Through each of the models of computation the researcher aims to develop an understanding of how computations are performed. Through a set of pre-defined rules computational arithmetic can be performed as humans, who just perform this arithmetic according to the rules thought to pupils at school. On the other hand computers tend to follow a very different set of rules. Models of computation are used to evaluate these rules by which a computer do a computation. It is easier to understand if the desired outcome is archived.

Models of computation are diverse and must be distinguished from computational models. Although used for same purpose, modelling, the difference is that a model of computation is dedicated to modelling computation. Because computational models compute models of different types for example: the strength of a particular chemical composition under specific pressures. As models of computation evaluate how a computation is performed by a computer based on a specific rule set. Different types of models of computation can be produced based on a variation of the underlying rule used to do the computation.

Models of computation are tools used to simulate computation such that behaviour can be studied in a closed environment. These models can be used to study several aspects of computation. Two key references to models of computation are used for this research (Fernández, 2009, Savage, 1998). Within the models of computation that these studies show, two groups can be distinguished. These groups are: the models of computation that are used to represent how the machine functions, and models of machine instructions. For this research the focus is on machine based modelling to gain a better understanding of how addition can be performed.

Both types of models have the same aim, to model the computation of a specific problem. However, when using a programming language based model of computation

the exact functioning of the machine is not considered. These can be performed through a set of known machine operations. This means that the performance of programming languages depends on how effective these machine operations are performed. So this research seeks to understand whether the most fundamental operation of addition can be performed more efficiently.

Within computer science the theory of computational complexity is used to determine the efficiency of an algorithm. As there are different types of criteria for measurement, time and space are most common factors. For this research measurement is performed based on the amount of operations. Where fewer operations are required by a model of computation to do an arithmetic computation, higher efficiency is attained which determines the computational complexity.

The notion of being fast is a subjective matter. This is so when evaluating the performance of a model of computation when performing arithmetic (addition). Through computational complexity rates of performance can be found and classified. So the evaluation of computing arithmetic can be objectively evaluated through *models of computation* against the *computational complexity theory*. This allows the researcher to draw aim conclusions with regards to the computational performance of computation.

The key to the research is the computational modelling of arithmetic. Through the model of computation the rules are defined of how each operation is performed. By using a uniform model of computation the paradigms and computational logic can be evaluated. These aim evaluations allow for computational performance comparison through computational complexity theory. The researcher seeks to derive *how computational performance can be improved within the model of computation.*

Chapter 2 Literature review

Introduction

The following chapter lays out the fundamental bricks to understand the research, methodology, analysis and how the findings are concluded. Most of the content is background knowledge and is essential to understand the computing paradigms and computational logic referred to within this research. The first section looks at how models of computation developed. This is followed by a revision of classical computation focusing on how addition is performed through Boolean and multi value logic. The second section of this chapter performs the same function for the quantum computing paradigm. Throughout each of the revisions for the different approaches to addition the circuit model of computation are evaluated.

Historic developments

Ancient methods of computation

Counting and recording of calculations goes back to the most ancient civilisations. The oldest existing tools are tally sticks on which carves are made as recordings of a count (Rizvi et al., 1991). Speculated stipulates use for counting moon occurrences. Tally sticks are marked per five. Addition of one is applied when the event is repeated allowing for the total amount to be summarised by the end. There are no real means for determining how fast a computation is done. Recording of the event depends on when the moon is visible, which is once a day. Calculating the total, grow at a linear rate which when tallied is at a rate of $(N/5) + X$ where X is the remainder.

Not all methods of counting and arithmetic have existing evidence in the form of a tool. This is for example: *finger counting* which is used by many civilisations in the world. Different civilisations use their own methods that depend on culture. Evidence of this

are numerical recordings with different number bases (Nordhaus, 2001b, Simon, 1997). The most common is the decimal one most used representing ten fingers. But the Babylonians are believed to start counting on their hands by representing twelve on one hand, the second hand to multiply by up to five forming the *sexagesimal system*. Decimal number counting is performed at the same rate as the tally stick. However, subtraction can also be done. With the sexagesimal system perform of *multiplication is possible.*

One of the more sophisticated methods of calculation still used today is the *abacus*. Existent in different forms around the world this tool allows for the four basic forms of *arithmetic* (Bae and Hwang, 2013). Most abaci exist based on the *decimal number base* which enables arithmetic to be performed at the rate *log10(n).* This represents ten to the factor of *X*. *X* being defined by the position in the number base. Other abaci use the *bi-quinary number system*. This is like a *tally system* in which bi symbolises two hands of which each has five fingers. The advantage over the decimal abacus is the decimal number system is fewer rods, as only six rods are required in *a bi-quinary system*. This implies that arithmetic can be performed at a rate of *log10(N-4).* This is the same as the decimal abacus less five rods per number position.

In summary, several different methods of *counting* and *arithmetic* existed since ancient times. This ranges from more basic tools such as: *a tally stick,* to *counting on fingers* and *abacus* which has increased capacity over finger counting. *Finger counting* is one of the most fundamental methods that defined the number base is used in the *abacus*. At the same time those methods of *arithmetic* were in use, complex forms of logic were developed. For example *Aristotle*, referred to as the *father of logic* developed *syllogistic logic* and *model logic* (Hurley, 2006). However, at that time no sign is observed of this logic being used for computational purposes.

Computation during the middle ages

During the early Middle Ages mechanical calculators improved from *abaci* for *astronomical* purposes. Based on a *sexagesimal number system* through a set of mechanisms involving gearing to track celestial body movement (Borrelli et al., 2013). Several types of astronomical clocks were built around the world. The most significant in this context is *programmable* and able to play music at predefined times when a celestial event occurred. At a linear computational rate it could determine the location of celestial bodies in the past or the future. With modern *non-linear* mechanics solution the process is sped-up. However the first developments of mechanical calculator did improve to this extent this is a complex task to archive.

Throughout the middle ages the need for performing arithmetic increased for tax collection. The developments in mechanics lead to more robust of gearing mechanisms. This was a significant improvement to the weaker wooden *astronomic clocks* that undergone excessive wear. The main difference is the number base system that represents the *decimal radix* (Ketelaars, 2001). Operating through twisting rods with one gear a number from *0* to *9* is represented with a gear for every position within the decimal number system. This lead to developments in mechanics calculators such that the four arithmetic functions are performable (Kidwell, 1992). However multiplication became faster because there was no need for operation in a sequence of additions. This implied it was *X* times faster than the first calculators where *X* is the multiplicand. A 1909 study demonstrated that a trained clerk is able to do addition six faster than on a piece of paper (Nordhaus, 2001a).

At the start of the industrial age *Babage* started work on the differential engine (Swade and Babbage, 2001). This implied a set of improvements in terms of mathematical ability. Around the same time, *punch cards* which was used in the weaver. To reduced the time to process a consensus from ten to five years and save five million dollars

(Truesdell, 1965). These phenomenal achievements in data management (processing) demonstrated to be the foundation of computer programming by the developments of Babbage his analytical engine (Bromley, 1998). Notably the concept of the punch card caused further developments towards a computer using *binary*.

The developments of *Babbage* forced further research to rethink *how computers would be built in the future*. Through this process the concept of a *'model of computation'* came into existence in the beginning of the twentieth century. The developments of *Boolean logic* inspired through George Boole are central to the design of computational models. As defined through its axioms the possible operations that can be performed (Shao, 2008). The usage of logic in computation distinguishes *classical computation* from *mechanical computing* devices during the middle ages.

Section one: Classical computation

The transition from mechanical computation to electronic computation meant in first instance computing devices becoming smaller. However more importantly, increased functionality can be attained as well as increased performance. Electronic computation adopted *Boolean logic* to operate on binary information. Through a combination of Boolean logic operations it became possible to perform arithmetic. As multiple Boolean logic operations are required to do a single arithmetic operation, its performance is questioned. However, this research seeks to understand *how computers perform arithmetic through Boolean logic*. *What the models of computations are and the components these consist out of as well as the underlying rule set.*

Classical addition

In classical computing bits consist out of *one* and *zero* are used to compute through *Boolean logic*. To develop an understanding of *how classical computers perform addition,* the *classical full adder* is built up from most fundamental components known as *logic gates*. It is reviewed how those logic gates function and can be used to do computation.

Gates

In classical computations a series of computational gates available that are analogous to *Boolean logic*. For this research the relevant logic gates are presented. These logic gates are depicted in pictures *one*, *two* and *three* below and are known as the 'AND' gate, 'OR' gate and 'EXCLUSIVE OR' gate respectively. Through these gates addition is

Figure 1 'AND' Gate

Figure 2 'OR' Gate

Figure 3 'XOR' Gate

perform in classical computers through sequencing the inputs and output through a series of combined logic gates. *Classical logic gates* take two input values that are either zero or one to produce a result. The result computed through a logic gate depends on *Boolean logic* by which the *logic gate abides*. For the three logic gates used in the *classical full adder* a truth table is included to show what the output is for any given two inputs for each of the logic gates.

Truth tables

The *truth tables* show the results for computing with the logic gates using any given combination of input allows to gain an overview of how the results are derived. For the first logic gate, 'AND' logic is applied. This means both inputs need to be *one* for the output to be *one*. In all other cases the *output* will be *zero*. On the contrary the '*OR*' gate has an output of one whenever any of the input combinations has a *one*. The *'EXCLUSIVE OR'* gate is the same as the '*OR'* gate. It has an output whenever the any of the inputs is *one*, except when both inputs are *equal*. In itself those logic gates provide little functionality as they are limited in the amount of operations they can perform. However, through applying a series of logic gates to an input it's possible to do very complex algorithms, for demonstrative purposes these logic gates are used to do *addition*.

Table 1 'AND' truth table

INPUT		**OUTPUT**
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

Table 2 'OR' truth table

INPUT		**OUTPUT**
A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

Table 3 'XOR' truth table

INPUT		**OUTPUT**
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

The simplest model of computation designed to do addition is known as the 'half adder' (Norman, 1960). In the classical computing paradigm, this consists of two logic operation. The *'XOR'* operation that is based on two binary inputs determines the 'SUM' and the *'AND'* operation which calculates whether there is a carry. The 'half adder' can be extended into what is known as the 'full adder'. The difference from the half adder is that it accounts for a carry input. This implies, more operations must be performed as there would be three and four inputs in the *classical* (Zhuang and Wu, 1992). This adder represents the simplest method of performing addition electronically with Boolean logic. Although efficient it is not optimal through its abstraction.

Full classical adder

Figure 4 depicts a full classical adder. The classical adder enables *three inputs* and *two outputs*. Inputs *A* and *B* are the values being added together and *C* in is a carry value from an earlier operation. The outputs are *S* and Cout. *S* holds the result for the addition of *A* and *B* while *Cout* is the carry value for addition. The *classical adder* exists out of two 'Exclusive OR' gates two 'AND' gates and a single 'OR' gate. The 'Exclusive OR' gates are used to determine whether the addition of the two input value will result in one

Through the 'Exclusive OR' gate the result will only be one when either of the two input values is one. No addition occurs when the inputs are both zero or both one. This gate is applied to input *A* and *B* as well as to the *output* and the *earlier carry* which will determine the *sum*. In the event that both input values to either of the two 'Exclusive OR' gates was *one* the carry are to be computed. so two 'AND' gates are included which will output *one* if both inputs are *one*. Through the final 'OR' gate both 'AND' gates reduced to *one* output determining whether the computation yielded a carry. The 'OR' gate outputs a *one* if either of the two 'AND' gates outputs *one*.

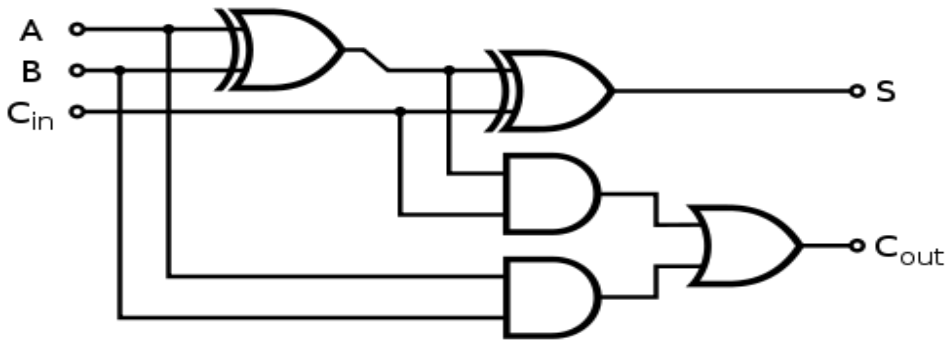

Figure 4: Full classical adder

The electronic methods of computation provided the advantage of being faster than mechanical computational devices. This was through achievement of full automatisation. On the other hand electronic circuits added a layer of complexity through its logic abstraction. This meant in fact it's slower than earlier methods of computation identified. Through this abstraction researcher were able to search for different approaches of performing addition with the purpose of optimising performance.

Extended adders

The simplest extension of the full adder to perform addition on numbers comprises of multiple bits and is known as the 'ripple carry adder' (Knauer, 1989). This is achieved through joining multiple 'full adders'. This is possible because as stated above, the 'full adder' has a carry input and is able to propagate a carry bit to the next 'full adder'. For the classical computing paradigm demonstration showed the 'ripple carry adder' to be an inefficient approach to performing addition (Knauer, 1989). The 'ripple carry adder' is inefficient because in the worst case it is required to propagate a carry for every binary addition performed. This increases the amount of operations significantly.

A further development to improve the performance of addition is the 'carry look ahead adder' (Doran, 1988). This is achieved through the usage of an additional model known as a 'carry look ahead unit', which reduces the amount of gate operations required to

propagate the carry. Some variants are known to have optimal performance under certain circumstances within the classical paradigm. Examples include: 'Manchester carry chain' (Needles, 1990), 'Brent-Kung adder' (Nowick, 1996), 'Kogge-Stone adder' (Knowles, 2001). This are interesting for reference purposes but do not add any value as the performance is insignificant.

Within the classical paradigm several other types of adders are known to exist. One of these types is known as the 'carry by pass adder' or 'carry skip adder' (Kobayashi et al., 2004). This adder makes use of propagation delay to decrease the amount of operations used. Similarly, the 'carry select adder' predetermines the propagation of the carry (Bedrij, 1962). The final alternative adders are the 'carry save adder' (Leininger and Taylor, 1978), and 'conditional sum adder' (Cho, 2003). Each of these has its own advantages and drawbacks in how they deal with performing addition.

Classical multi-value adder

A revision of the literature found there to be a significant amount of research done to achieve higher number base computation through *multi-value logic*, as an alternative to 'binary' and 'Boolean logic'. Application of *multi-value logic* can solve more efficiently *binary* problems, as well as improve circuit design (Dubrova, 1999, Dubrova, 2002). The advantages are summarised as: a reduction in signals; the ability to store two bits of memory instead of one, and improved arithmetic operations (Dubrova et al., 2002). Most common applications of *multi-value logic* are 'ternary' (Gang et al., 2009) and 'quaternary' (Dornajafi et al., 2008, Gaidhani and Kalbande). Furthermore, demonstration showed how addition can be performed through *multi-value logic* (Gonzalez and Mazumder, 1998). So, its application to the performance of *addition* is further investigated in this study.

Similar to the development of adders with binary Boolean logic, a *ternary* (A. Rizvi et al., 1991) and *quaternary* (Mingoto, 2006) *half adder* were created. Likewise these developments extended to 'full adders' with *ternary* (Srivastava and Venkatapathy, 1996) and *quaternary* logic (Thoidis et al., 2001). Notable the research in *multi-value logic* takes a lower level view in how gates are produced. It's very much based on electronic diagrams, rather than the circuit models used in *Binary Boolean logic*. Furthermore, the focus is not on the concept of performance. This may be in part because demonstration showed computation through *multi-value logic* will provide a computational speed-up. Instead, most of these researches are concerned with power consumption and reducing its voltage. As there are no performance metrics. This study evaluates the performance of the *multi-value ternary* and *quaternary full adders* to be compared against the results for the *binary logic full adder* identified in the previous section.

Computational logic

The concept of many value logic is the next step where rather than looking at the possible truth values were the proposition is true or false. Or in the case of modal logic where the degree to which it is so is defined. Many value logic allows to specify other values then true or false that are not the degree of its verity but could be used as such. This logic allows for any number of outcomes to be the result of the proposition (Gottwald, 2005). Academics argued logic to not be depend on one pre-defined axioms defining the outcome and so can't be considered a real logic. This is not necessarily true as it could be possible for the outcome is dependent on logic. The key difference is, in Boolean logic the output being either true or false. Instead some different combinations can be formed based on the amount of inputs specified determining unique outputs. The development of these concepts was around the time when quantum mechanics started to become better understood. It can be argued to be a product of a physics discovery.

Gates

The *ternary logic gates* are the same as *classical logic gates* as they have *two inputs* and *one output*. Even so *ternary logic gates* distinguish themselves from their *classical counterparts* because they operate on *ternary bits*. This is different because opposed to *classical bits* that can hold *one* of *two* values, *ternary bits* can hold *one* of *three* values. Consequently, *ternary logic gates* operate differently on the ternary bit inputs. The key difference here is in the different combinations can be output.

To construct, a *full ternary adder* three *ternary logic gates* are required. These are the 'SUM' logic gate, consensus logic gate and 'ANY' logic gate that are depicted below. The 'SUM' logic gate is used as an alternative to the exclusive 'OR' gate in the classical adder. The 'Exclusive OR' gate computes modulo two and the 'SUM' gate computes modulo three. Similarly the consensus logic gate is used to replace the 'AND' logic gate which performs the inverse function of exclusive by acting only when both inputs are the same. Hence the gate is able to show a consensus between both inputs. The third gate named 'ANY', functions as the 'OR' gate in *classical computing*.

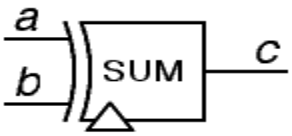

Figure 5 Ternary 'SUM' Gate

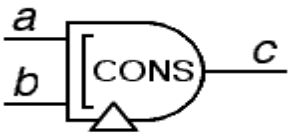

Figure 6 Ternary 'CONSENSUS' Gate

Figure 7 Ternary 'ANY' Gate

Truth tables

To understand how those logic gates function each of their truth tables are depicted below. For the 'SUM' logic gate in table four demonstrations showed how modulo three is attained, when the output reaches *three* the value is reset to *zero*. Considerably, when the inputs are *one* and two, the output is *zero* and when both inputs are *two* the output is

one. This is an effect of *modulo three* restarting counting at *zero* when three is reached. The *second ternary logic gate* is known as the *consensus gate* because its output is determined by whether both inputs are the same. In the case of a consensus between both inputs the output is equal to the input or defaults to *one*. The final logic gate any input to be matched to its output. This means both inputs don't have to be *zero* for the outputs to be *zero*. Except when the inputs are the extreme thresholds the output is *one*. As a general rule for the 'ANY' gate both inputs can be added together and divided by *two* to determine the output.

Table 4 Ternary SUM truth table

SUM	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

Table 5 Ternary CONSENSUS truth table

CONS	0	1	2
0	0	1	1
1	1	1	1
2	1	1	2

Table 6 Ternary ANY truth table

ANY	0	1	2
0	0	0	1
1	0	1	2
2	1	2	2

Full ternary adder

The *full ternary adder* is a circuit through which *ternary bits* can be added together. Figure 8 depicts a full ternary adder which resembles the *binary classical adder*. As the *classical adder* has three inputs, two outputs, and five logic gates. Also it is noticed that to determine the sum output *two* 'SUM' gates are used in a similar way to the *classical adder* which used *two* 'EXCLUSIVE OR' gates. As previously described the 'SUM' gate performs the same functionality that is to do *modulo three*. This means both values will be added together and ignore the *carry*. To perform the carry in the *consensus gate* is used that is like the 'AND' logic gate. The Outputs of the consensus gates are then merged through the 'ANY' logic gate and output the carry for the *addition*. The functionality of the gates and the sequence in which the inputs are passed to the logic gates resembles the *classical full adder.*

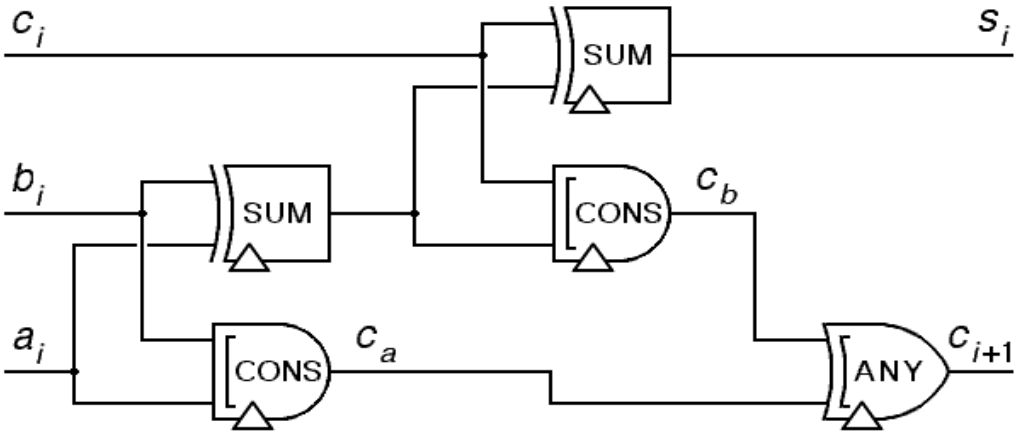

Figure 8: Full ternary adder

Multi-value Ripple carry adder

The 'ripple carry adder' is known to be the simplest extension and is shown to be rather inefficient with *Boolean logic*. Perhaps the reason no implementation of this of *addition* approach in *multi-value logic* exists. Studies of *binary* and *ternary multi-value logic* demonstrated 'binary adders' to be faster, (Vranesic and Hamacher, 2009). This is a contradiction to the first findings in the literature that claimed 'multi-value logic' to be more efficient than the 'binary Boolean logic'. It's possible that for the *task addition* this is not the case.

Extended adders

The amount of variants developed for the 'adder' based on 'multi-value logic' is limited. One of the alternatives is an implementation of a 'ternary select adder' within the classical paradigm. However, there is little concrete indication of performance improvement enabling comparison with other models. Further research must be conducted to demonstrate its advantages (Burgess, 2001).

Classical computation Summary

In the first section of the literature review the classical computing paradigm its approaches to *addition* are evaluated. Starting with the fundamental concepts of *how*

computational logic is utilised to perform addition, followed by the *full adder*. Several of the *large scale adder* implementations are reviewed and evaluated based on the literature findings for performance. Within the *classical computing paradigm* comparison is made with *multi-value logic*. In attempt of comparing the performance results of *addition* for *binary logic* based computation opposed to *multi-value logic* computation. Although the meta-literature suggests *multi-value logic* is more efficient, little evidence is found to back-up this claim (Dubrova, 1999, Dubrova, 2002). With an overview of computation of addition in the *classical paradigm* for *binary* and *multi-value logic*, the quantum paradigm is reviewed in the following section.

Section Two: Quantum computing

Current computing is on the verge of a transition from a *classical* to a *quantum* computing paradigm. This transition is based on discoveries in the science of physics, in which quantum mechanics developed enabling a computational speed-up (Feynman, 1982). This is demonstrated through a series of algorithms (Deutsch and Jozsa, 1992, Grover, 1996, Shor, 1994). However, the effect of quantum speed-up applies to these algorithms, which are very specific tasks of computation such as *integer factorisation* and *searching*. Furthermore, identified computers are dependent on the computational task of *addition*, as all other *arithmetic* depends on; for example, it's found in the literature that Shor's (1994) *algorithm* depends on modular exponentiation, which cannot be performed without *addition* (Berman et al., 2001). Shor's (1994) *algorithm* is demonstrated a *quantum speed-up* (Shor, 1994), but not for the part of *addition*. so, this study seeks to *investigate whether such an improvement in speed is possible through the revision of computational model and approached to addition.*

As described by (Barenco, 1996) *quantum computing* is different from *classical computing* at its most fundamental level. For example: the 'Bit' in the case of a *classical computing* and 'Qubit' in the case of a quantum computer. While the classical

‘Bit’ can hold either the values *one* or *zero*, the ‘Qubit’ is able to hold both in the same time. This scales as more ‘Qubits’ are included. When *two* ‘Qubits’ are used, *four* values between *zero* and *four* can be stored. In the case of *three values* this becomes *eight*. These values are stored in a universe. A further advantage enables computation to be performed on all values at the same time; however at the end of a computation only *one* result can be retrieved. This is known as the ‘*quantum parallelism thesis*’, the key reason *quantum computers are faster than classical computers* (Duwell, 2007).

Quantum computing is the science that seeks to understand how quantum mechanics can be used to perform computation. The research on quantum computing is driven by the potential to *miniaturise computers* and make possible a *computational speed-up*. This all started when a physicist who tried to demystify quantum mechanics through simulating its phenomena on a computer (Feynman, 1982). This is impossible because the rate at which the complexity of quantum mechanics increases is higher than a computer. Based on these findings and speculation, research began to understand *how it could be possible to use quantum mechanics to solve computational problems currently unsolved due to their inherent complexity.*

The research in *quantum mechanics* applied to computer science is *active research* and is so far only partially successful. The *complexity* in developing a *quantum computer* is in the *quantum mechanics,* which is not completely understood. There are several different interpretation to quantum mechanics, some of which have been disproved while other are subject to interpretation (Schmelzer, 2011). The most common view is the ‘Copenhagen interpretation’ that is most accepted for *teaching* purposed (Mermin, 2003), and is accepted by early scientists working on a quantum computer.

However, more recent research suggests the ‘many world interpretation’ is more suitable for quantum computing enabling parallel computations through its multi-

universe (Osnaghi et al., 2009, Wallace, 2002). *Multi-universe view of quantum mechanics* is the underlying theory that explains the *quantum parallel thesis*. The researcher of this study **does not seek** to elaborate on the mysticism of what is uncertainly known for the science of quantum mechanics. But **rather seeks** to exploit *what is known about the states of particles* (Audenaert et al., 2012, Bergou et al., 2012) in *quantum mechanics.* And evaluate whether it's possible to improve computational performance through a *hypothetical model of computation*.

In principle, quantum computing and classical computing are the same. However, in theory quantum computing is shown to be able to solve some computational problems in less time than in classical computing. *Richard Feynman* demonstrated quantum mechanics enabled an increased computational *speed-up,* based on the idea of the *multi-universe interpretation of quantum mechanics* (Deutsch, 1985).

The research into building a computer based on *quantum phenomena* according to the *multi-universe interpretation* as previously described is on-going. There have been claims that a quantum computer has been built. However, significant criticisms of academics who stated this implementation is not valid and does not provide a computational speed-up (Van Dam, 2007). More recently, the University of Bristol developed a *universal quantum photonic chip for educational purposes* (Shadbolt et al., 2011).

The Business school Said of Oxford University in 2005 developed an interesting study during which questioned leading researchers in quantum computing. Based on their findings, a marketing study is completed. This study brings together the research potential of quantum computers and the application for which they can be used in direct relation to market demand of computer use (Corker, 2005). According to this study the following applications are market demanding: 'quantum cryptography, quantum

computer, quantum auctions, quantum gaming, quantum scheduling and optimisation, and quantum meteorology'. Interestingly, they have developed a timeline on which is demonstrated where they expect those technologies to emerge and become available on the market. Notably, wide spread commercial application will take several years to become available; however *quantum encryption from point to point* is implemented and used since 2005. Such applications are not commercially available; it is suspected that they are used for government maters. Small scale prototype developments in quantum computing labs are a clear development towards building quantum computers on a large scale that may become available within the foreseeable future (Lloyds 2008).

Decoherence

While researchers are running ahead trying to determine how quantum computing will revolutionise computing, scientists are still resolving a fundamental problem known as 'Decoherence'. Even though, it's been possible so far to design and implement a *quantum chip*, because of the problem of *scaling u*p the quantity of 'Qubits' involved. This is caused by the problem of 'Decoherence', which occurs when 'Qubits' interact with the environment and 'Decoherence' or undergo a state change (Golubev and Zaikin, 1998).

Nevertheless active research seeks to overcome 'Decoherence' known as: *quantum error correction* (Steane, 1998), *stabiliser codes* (Gottesman, 1997), *entanglement assisted quantum error correction* (Hsieh et al., 2007), and *quantum convolutional codes* (Chau, 1999). For this research it's important to be aware of *Decoherence* and the *implication* is for the performance of computation, as additional equipment is required to prevent 'Decoherence'. Nevertheless, the prevention of 'Decoherence' is outside the scope of this research

Quantum logic

The concept of *quantum logic* developed by (Birkhoff and Von Neumann, 1936). The leading scientist in this field interested in developing the field of logic based on the findings of *quantum mechanics*. Through these developments, the basics of quantum computing are established. However, the developments in *quantum physics* developed over time. One of the key contributors of *quantum theory* is (Mackey and Benjamin, 1967) who had a renaissance effect quantum logic its *algebra*. This became the *standard quantum logic* deviated from the quantum logic in the complete *orthomodular* lattice based on the closed subspace in a *Hilbert space* (Birkhoff and Von Neumann, 1936). The most common form is 'Orthomodular Quantum Logic' (OQL) of which modal interpretation exists.

According to (Greechie, 1981) OQL rectified by 'Hilbert Quantum Logic' (HQL). Furthermore, research conducted to use the concepts of first order logic in quantum logic. (Takeuti, 1981) further developed *quantum logic* in the field set theory through creating equivalent concepts of *Boolean logic*, based on the *algebraic* structure of a complete *orthodular lattice*. At a later stage, started off some serious criticism against the standard logic (Ludwig and Hein, 1985). In favour of the initial concepts developed (Birkhoff and Von Neumann, 1936). And support developed for the concepts in quantum mechanics referring to *pure* and *mixed* states. Subsequently, other forms of quantum logic developed.

The science in quantum mechanics allows for quantum computing is known as *state duality*. This means *two* states can be held simultaneously. This is the fundamental principle behind the *quantum bit* which distinguishes itself from the *classical bit*. The *quantum bit* state determines the information that can be stored. For a *pure state* this means the simple representations of the quantum bits are represented. For example: with a register of *three quantum bits* the value *eight* can be stored. In contrast with *mixed*

states any of the *parallel universes* can be accessed to perform a computation on the register. This is because they can represent any of the possible outcomes to a certain extent based on its statistical properties.

This is the most fundamental concept behind *quantum computing* and the potential of performing hard computational tasks in the c*lassical computing paradigm* solvable. Therefore this research investigates *how quantum addition is performed and whether the research is able to take advantage of quantum states to attain improved performance when computing addition*. Also, the research in quantum computing investigates the amount of *states* that can be identified. This research is known as *quantum state discrimination(Bae and Hwang, 2013)* or *distinguishability* (Borrelli et al., 2013). Based on this research in quantum states, this research *seeks to re-evaluate how quantum computing performs addition*, by using the *different states* identified as *a number base system*. This different view, on how to use the science of quantum mechanics in applied computer science, is compared against conventional computation in the classical and quantum paradigm for both *binary and multi-value logic*.

Quantum addition

The first two *full adders* described so far are within the same computing paradigm known as *classical computing*. They are distinct in the *number base* used. Where the *first one* employed *binary* the *second one* is based on *ternary*. This implied both required a similar form of logic to perform computation. The following full adder distinguishes itself in terms of its computing paradigm. This is because it operated on *quantum bits* rather than on *classical bits*. A *quantum full adder* has a unique set of logic gates. To perform *addition two* distinct logic gates are required in the quantum paradigm.

Quantum gates

The logic gates used in quantum computing are very distinguished from its classical counter parts. This is predominantly due to having to operate on *quantum bits*. Furthermore, the quantum gates used in the *quantum full adder* differ from *classical binary* and *ternary gates* in an equal amount of inputs and outputs used for every logic gate. More important is the concept of *quantum control bits* which are used to operate of *quantum target bits*. In figure 9 a 'CNOT' logic gate is depicted. It consists of *one quantum control bit* and *one quantum target bit*. While the gate in figure 10 the quantum 'TOFFOLI' gate has *two quantum control bits* and *one target bit*. Quantum control bits are used to determine the target bit. The 'CNOT' gate inverts the quantum target bit depending on the state of the quantum control bit. The 'TOFFOLI' gate performs the same function as the 'CNOT' gate, with the difference that the inverting of the target bit depends on both control bits. With different types of bits quantum computing requires logic gates function differently. To gain a better understanding the truth tables of the quantum logic gates need to be examined.

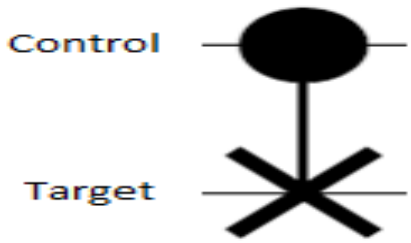

Figure 9 Quantum 'CNOT' Gate

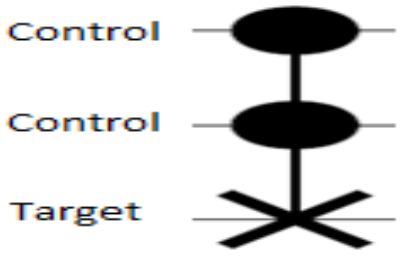

Figure 10: Quantum 'TOFFOLI' Gate

Truth tables

The truth tables for the quantum 'CNOT' and 'TOFFOLI' gate demonstrate the difference with *logic classical gates.* Most convenient with the quantum gates is the *control quantum bit* never changes; the input is always the same as its output. On the contrary the *target quantum bit* is changed in the 'CNOT' gate when the quantum

control bit is *one*, and left to its default value when *zero*. The same logic is applied for the 'TOFFOLI' gate with the difference that both quantum control bits have to be *one* to change the quantum target bit. The logic can be confirmed in truth table 7 and 8 below.

Half quantum adder

The 'quantum half adder' is analogous in the quantum paradigm, which performs in principle the same logical operations (Barbosa, 2006). However, based on the knowledge that the 'Qubit' is different compared to the *classical bit*, quantum gates are used to simulate *Boolean logic*. Most significantly, the quantum adder requires *three* 'Qubits' while the classical adder requires *two* Qubits'. Furthermore, both models use an equal amount of operations to perform computation, which means *addition* is performed at the *same rate*. Similarly to the classical adder the same abstraction is used. Therefore it is questioned how efficient an implementation of addition the quantum adder is?

Table 7 Quantum 'CNOT' truth table

INPUT		OUTPUT	
C	T	C	T
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Table 8 Quantum 'TOFFOLI' truth table

INPUT			OUTPUT		
C	C	T	C	C	T
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

Full quantum adder

With quantum bits and quantum logic gates so fundamentally different from their classical counterparts, it's questioned how computation can be performed. Below a full quantum adder is depicted. It contains more inputs and outputs, but has one less logic gate. Although at first sight the full quantum adder looks very different from the

classical full adder, they hold a great deal in common. Essentially, a different approach is used to attain the same objective of adding two inputs *A* and *B* together. The quantum full adder does this by determining the *sum output* through 'CNOT' logic gates. Thus when either of the inputs, *A* or *B* is *one* the output sum is inverted. In the case of a *carry* it would become *zero,* and when both inputs are *one* the second one cancels out the first one.

Herby the effect of the 'Exclusive OR' logic gate in the classical adder are attained. To determine whether there should by a *carry output K* the 'TOFFOLI' gate is used. A *carry output* if one of the inputs is *one* and the output is *one*. This cannot occur twice because the first 'CNOT' will negate the *carry* input, therefore not allowing the *carry output* to be reset to *zero*. The 'TOFFOLI' gate has a similar effect as 'AND' gates in the classical full adder. The final 'OR' logic gate is not required in the quantum full adder. Its structure through which the two 'TOFFOLI' gates produced a single output, hence only four logic gates are required. Revision of the quantum full adder found the underlying logic behind similar to the classical adder (Cheng and Tseng, 2002).This added to the question of whether they are equal in performance?

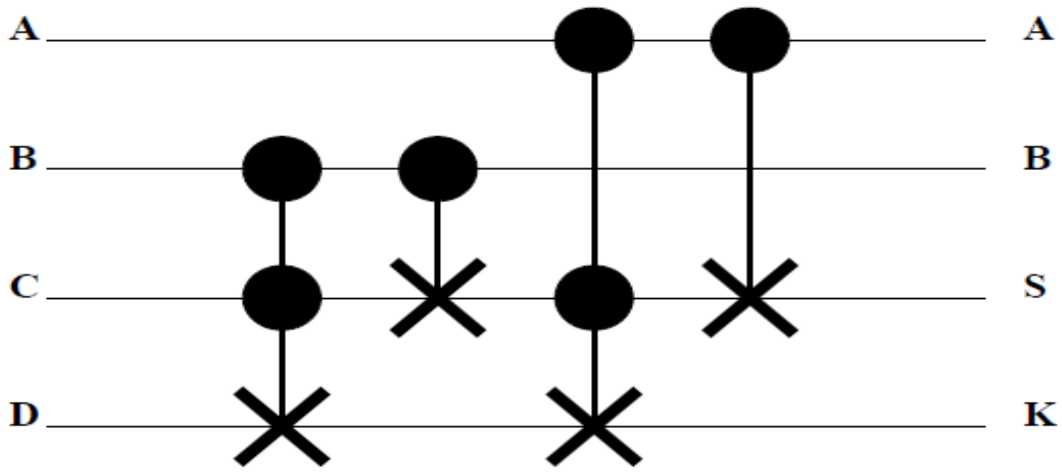

Figure 11: Quantum full adder

Quantum adders

Classical quantum adders

The ‘quantum ripple carry adder’ consists of multiple ‘full adders’, and is demonstrated to give increased efficiency to linear time in the quantum paradigm (Beckman et al., 1996, Cuccaro et al., 2004, Vedral et al., 1996, Gossett, 1998). The ‘quantum carry look ahead adder’, is a significant improvement over its classical counterpart, as it’s reduced to logarithmic depth through application of modular arithmetic (Draper et al., 2006). Other improved implementations are the ‘bypass adder’, which a model based on binary Boolean logic created implementing skip logic (Islam et al., 2010). The ‘quantum carry select adder’ is demonstrated to perform addition in squared linear time (Meter III, 2006).

The most efficient implementations are the ‘carry save adder’ (Gossett, 1998) and the ‘conditional sum adder’ (Meter III, 2006). These adders are based on *modular arithmetic* rather than being based on *Boolean logic* in the quantum paradigm. The ‘carry save adder’ has a reduced number of gates at the cost of increasing the amount of ‘Qubit’ used within the computation. On the other hand, the ‘conditional sum adder’ achieves a more successful result, enabling addition in logarithmic time.

Non Classical quantum adders

A number of *adders* existing only in the quantum paradigm are identified. There is no classical counterpart as these adders are based on quantum phenomena. One example is based on the *quantum Fourier transform* (Draper, 2000). However, performing addition in linear time is less efficient, but is an improvement in terms of the amount of gates used. Another example, based on quantum mechanical properties is the carry look ahead design, is based on quantum measurement (Trisetyarso and Van Meter, 2009).

Through the interaction of a measurement the state is changed in such a way that a computation is performed. This is demonstrated to be more efficient on a larger scale.

For those who have developed a model of computation based on *a binary radix* through either *binary Boolean logic* or *modular arithmetic*. it's found within the *classical paradigm* the computational task of arithmetic is performed at best in linear time (Pai and Chen, 2004). The same is demonstrated in the quantum paradigm through the 'conditional sum adder' (Meter III, 2006). However, demonstration showed in the quantum paradigm a faster implementation achieved (Choi and Van Meter, 2008). This is based on a 'Kd mesh' enabling the reduction first to a squared root of linear time, then cubed, then to the power of four, and so on. Thus, the factor of the square root increases throughout the lifetime of the computational task. According to the structure of the 'Kd mesh', where K stands for the denominator of the root, and *d* stands for the dimensions of the mesh. Similar to using a binary tree enabling a computation in linear time, this data structure allows for reducing the performance of a computational task to a rooted time. These claims are very interesting, however are not validated by other studies.

Quantum multi value logic

Revision of addition in the quantum paradigm leads to several interesting findings which are improved performance and alternative approaches to quantum addition based on quantum mechanics. This research is not extended to quantum multi-value logic within the quantum paradigm. This enables to compare quantum multi-value logic addition against quantum addition and addition within the classical computing paradigm. Therefore quantum multi-value addition is further investigated in terms of its functioning.

Gates

Figures 13 to 18 depict the *elementary quantum ternary functions* of which any *three* can be used in conjunction to form *a quantum ternary logic gate*. The **first** operation leaves the input unchanged. The **second** adds one to the output, the **third** adds two. The **fourth** multiplies the input by two. The **fifth** multiplies the input with two and adds one. The **last** multiplied the input with two and adds two. To form a quantum ternary logic gate any of these six operations are joined together as depicted in figure 12. It's important to consider the order of the operations as the quantum control bit determines which of the three operations will be performed on the input.

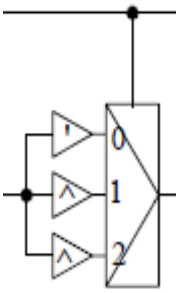

Figure 12 Sample Quantum Ternary Gate

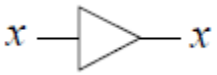

Figure 13 Quantum Ternary Buffer

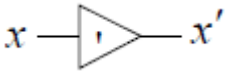

Figure 14 Quantum Ternary Single shift

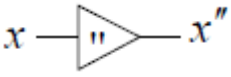

Figure 15 Quantum Ternary Dual shift

Figure 16 Quantum Ternary Self shift

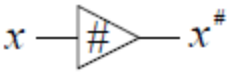

Figure 17 Quantum Ternary Self single shift

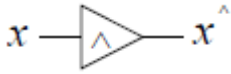

Figure 18 Quantum Ternary Self dual shift

Truth tables

It's easier to understand the exact functioning of each operation through their truth table. As identified with the classical ternary adder, modulo three is applied. This is seen in the single shift operation when *one* is added to the *input*, the output is reset to *three*. For the *last three truth* tables including a *self-shift* the input is first multiplied by *two*. Any multiplications with *zero* have the output of *zero*.if two is multiplied by itself it's reset at *zero* and becomes *one* as depicted below in table 9 - 14.

Table 9 Quantum Ternary **Table 10 Quantum Ternary** **Table 11 Quantum Ternary**

Buffer truth table

Buffer	
X = X	
Input	Output
0	0
1	1
2	2

Table 12 Quantum Ternary

Single shift truth table

Single shift	
X = X + 1	
Input	Output
0	1
1	2
2	0

Table 13 Quantum Ternary

Dual shift truth table

Dual shift	
X = X + 2	
Input	Output
0	2
1	0
2	1

Table 14 Quantum Ternary

Self shift truth table

self shift	
X = 2X	
Input	Output
0	0
1	2
2	1

Self single shift truth table

Self single shift	
X = 2X + 1	
Input	Output
0	1
1	0
2	2

Self dual shift truth table

Self dual shift	
X = 2X + 2	
Input	Output
0	2
1	1
2	0

Quantum ternary adder

The full *classical ternary* adder is an extension of the *full classical adder*. And the *full quantum ternary adder* is a modified version of the *full quantum adder*. In both cases *quantum bi*ts are used, however for the quantum ternary adder *three states* can be distinguished. Therefore different logic gates are required to operate on ternary information. Figure 12 demonstrates a full ternary quantum logic gates sub-exists out *three* of the *six* elementary quantum ternary gate operations. Like quantum computing a control bit that in this case is ternary will determine which of the *three* operations must be performed.

Full ternary quantum adder

Within the quantum paradigm ‘half and full adders’ are identified for both ternary and quaternary logic (Chattopadhyay et al., 2009, Hung et al., 2004, Khan, 2004a, Khan and Perkowski, 2007). While some of the quantum implementations are described through specific quantum technologies. This is not always the case when theoretical designs are made of how an ‘adder’ would operate, based on what is known about quantum mechanics. Nevertheless, at this stage it has not been possible to determine whether a ‘quantum adder’ for either *ternary* or *quaternary multi-value logic* could yield higher performance. But it’s identified that multi-value logic adders require more gates (Jahangir and Das, 2010).

The *full quantum ternary adder* is most distinguished from all previous adders demonstrated. This is because within a single ternary logic gate it’s possible to perform one of three operations based on *the quantum ternary control bit*. This makes the full quantum ternary adder more complex. It consists in total out of ten logic gates which is a double of the classical adder. It is different from the quantum full adder because there are no logic gates with two quantum control bits. Also there are five inputs and output quantum bits. Unlike the previous reviewed full adders, the full quantum ternary adder bears no resemblance in terms of the underlying logic used to perform addition between two quantum ternary values. Its difference with the full classical ternary adder is that its logic gates have control bits. Also it is different from the full quantum adder in that it does not have gates with multiple control bits. Therefore the logic used in the previously investigated full adders cannot be found in full quantum ternary adder. Instead it uses a convoluted method to determine the carry within the addition which makes it very inefficient.

Table 15
Quantum Ternary adder Gate 1

1			
In	**Out**		
	0	1	2
0	1	0	1
1	1	0	1
2	1	0	1

Table 16
Quantum Ternary adder Gate 3

2			
In	**Out**		
	0	1	2
0	1	2	0
1	2	0	1
2	0	1	2

Table 17
Quantum Ternary adder Gate 3

3			
In	**Out**		
	0	1	2
0	1	2	2
1	2	1	1
2	0	0	0

Table 18
Quantum Ternary adder Gate 4

4			
In	**Out**		
	0	1	2
0	2	1	1
1	1	0	2
2	0	2	0

Table 19
Quantum Ternary adder Gate 5

5			
In	**Out**		
	0	1	2
0	0	0	1
1	2	2	0
2	1	1	2

Table 20
Quantum Ternary adder Gate 6

6			
In	**Out**		
	0	1	2
0	0	1	2
1	2	0	0
2	1	2	1

Table 21
Quantum Ternary adder Gate 7

7			
In	**Out**		
	0	1	2
0	2	0	1
1	0	1	2
2	1	2	0

Table 22
Quantum Ternary adder Gate 8

8			
In	**Out**		
	0	1	2
0	0	1	1
1	1	2	0
2	2	0	2

Table 23
Quantum Ternary adder Gate 9

9			
In	**Out**		
	0	1	2
0	1	0	1
1	2	1	0
2	0	2	2

Table 24
Quantum Ternary adder Gate 10

10			
In	**Out**		
	0	1	2
0	0	1	0
1	1	2	1
2	2	0	2

To be able to evaluate the full quantum ternary adder for verification purposes, intermediate truth tables are included in tables 15 to 24. The quantum full ternary adder is depicted in figure 19. For each phase in the diagram a truth table is included with a corresponding number. These truth tables demonstrate for each possible input in the input column, what the output would be based on the control bit at the top of the output column.

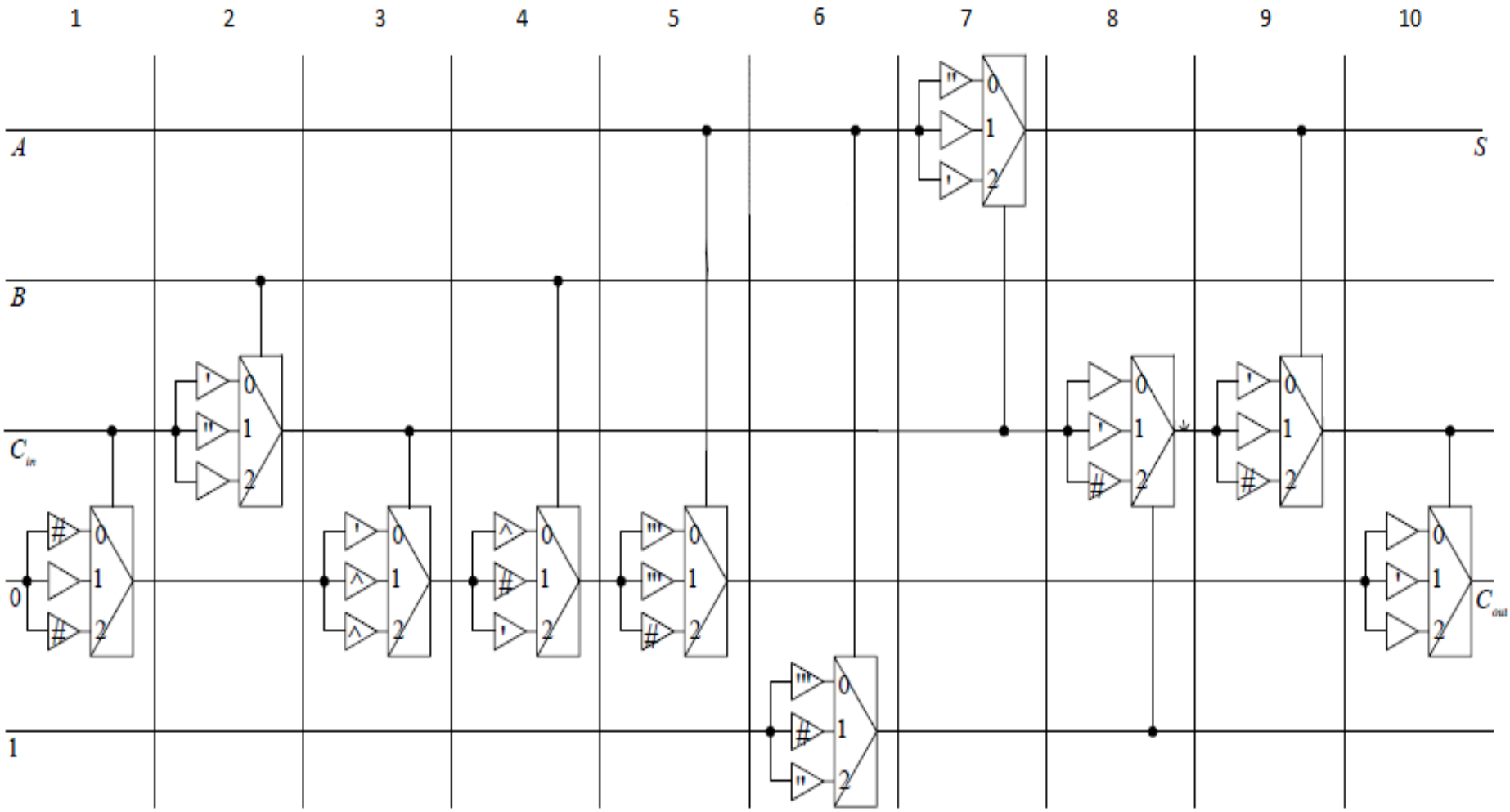

Figure: 19 Full Ternary adder

Literature review summary

Two types of machine based models of computation that are suitable for this research are identified. The *logic model of computation* in the form of *Boolean circuits* is the most often used to model the computational *task addition*. It's used in the *classical paradigm* for binary Boolean computation (Bedrij, 1962, Cho, 2003, Doran, 1988, Knauer, 1989, Knowles, 2001, Kobayashi et al., 2004, Leininger and Taylor, 1978, Needles, 1990, Norman, 1960, Nowick, 1996, Zhuang and Wu, 1992). The *carry save adder* and *conditional sum adder* are the most efficient in the *classical paradigm.* However, performance depends on the magnitude of how large the implementation is *scaled up*.

Implementations of *multi-value logic* within the *classical paradigm* with the purpose of improving the performance for computing addition are plentiful (A. Rizvi et al., 1991, Dornajafi et al., 2008, Dubrova, 1999, Dubrova, 2002, Dubrova et al., 2002, Gonzalez and Mazumder, 1998, Mingoto, 2006, Srivastava and Venkatapathy, 1996, Thoidis et al., 2001, Vranesic and Hamacher, 2009). It's found that for most of the different approaches to addition a multi-value logic implementation is created and published. However the performance benefits are not transparent due to insufficient and unclear reporting.

In the *quantum paradigm* a different approach to implement *Boolean logic* are found. Nevertheless, approaches to the *computation of addition* are inspired on existing implementations in the classical paradigm. An implementation is for most different types (Barbosa, 2006, Beckman et al., 1996, Cheng and Tseng, 2002, Cuccaro et al., 2004, Gossett, 1998, Islam et al., 2010, Meter III, 2006, Trisetyarso and Van Meter, 2009, Vedral et al., 1996) and *binary modular arithmetic* (Draper, 2000). The findings in the quantum paradigm are interesting from the perspective that show significant

performance increases over the classical counterparts, especially, when quantum phenomena is used to perform computation of *addition*.

Muli-value logic within quantum paradigms is under developed. Few studies implement an approach to *computation addition* (Chattopadhyay et al., 2009, Hung et al., 2004, Khan, 2004b, Khan, 2004a, Khan, 2008, Khan and Perkowski, 2007, Oklobdzija et al., 2003). Out of the *few studies existent* it's impossible to deduce what the performance implication are for implementing *multi-value logic addition* in the *quantum paradigm*.

The findings are mixed, in first instance it's unclear what the performance results were. Most of the implementations to the different approaches to addition within each of the computational paradigms for the computational forms of logic specified. Even in the cases where performance results are reported, it's uncertain how comparison can be performed. In some cases different types of performance are measured and reported.

One of the key findings in this literature review is the link between *quantum phenomena* and *computational performance*. It's found that for the *computation of addition* significant performance results are demonstrated. So **this research seeks to** *investigate quantum phenomena in relation to computational performance, with specific focusing on how the concept of state discrimination can be used*. The research in quantum mechanics on state discrimination could enable an alternative approach to its application in the computation of addition. **Through this research**, it can be established whether through quantum state discrimination computation at a higher number base can allow computation of addition at a higher performance. This means that the paradigm shift to quantum computing must be reviewed.

The main obstacle identified within the literature review is the inconsistency between different studies on the topic of *addition* between each of the *paradigms and computational logic*. This lead to finding results of different metric types which are not

readily comparable. Therefore, *this research seeks to take an academic approach to the evaluation of performance. Through the usage of an automaton model, the computation of addition is modelled and measure for performance with computational complexity theory. This should enable objective comparison between different approaches of addition.*

Chapter 3 Methodology

Introduction

The following chapter seeks to outline the mind set the research adopted during the research to allow other researchers to validated and evaluate the findings. The research questions and objectives are reiterated as a reminder of what the purpose of the research is. This will help to understand how the methodology enables this research.

The research methodology consists out of two parts which are the researchers mind set and the practical methods used to attain the research objectives. To outline the mind-set required be adopted when reading this research a research paradigm based on the epistemic stance reason and a realist ontology, the researcher seeks to deduce answers to the research questions. As part of this mind set it's questioned through critical evaluation whether the experiment is subjective. Also the research questions the current paradigm shift in computer science based on research findings in quantum mechanics.

The research experiment is based on two primary research methods known as automaton model of computation and computational complexity. The automaton model of computation is used to model the computation addition for the different paradigms and computational logic which allows for comparison while computational complexity theory is used to determine the performance of each approach to with the purpose of understanding how performance can be increased. As the automaton model of computation is a universal model of computation it's able to demonstrate how an alternative view on the application of quantum mechanics can enable a performance increase in the computation of addition through emulating a higher number base, opposed to binary.

Research questions

1. What are the available computational models and how can these be used to evaluate the computational performance of addition?
2. Find out what the performance implications are for addition in different paradigms and logic, and how can this be measured systematically?
3. Based on the findings of research question two, can the performance of addition be improved?

Research objectives

1. Identify a suitable model of computation for evaluating different approaches to addition.
2. Model the addition in the chosen model for each paradigm and logic and evaluate their performance.
3. Identify what is the contributing factor for performance in computation of addition.

Research paradigm

Within the research community a broad scope of research methods. To provide an overview (Clear, 2004) reiterated three paradigms as: “scientific, interpretivistic, and critical enquiry”. Whilst his purpose is to demonstrate how critical enquiry can be used in computer science research, he states, most research in computer science is within the scientific paradigm. This is true for this study, this is because formal statements are made about the models of computation presented and use a formal method for accessing their computational complexity. This is instead of using methods that look at understanding complicated phenomena (quantum mechanics) which is complex to understand. However, this study is based on such complex phenomena nevertheless remains superficial of it. Furthermore quantitative methods are used in this study

regardless of whether the statement of computational performance holds true for the models of computation. It's the distinction in quantitative methods over qualitative methods that make this study of scientific nature. In favour of the scientific paradigm this study does not consider any sociological implication so the critical paradigm is excluded as well.

The educational field of computer science (scientific paradigm) has diversified and evolved into different theoretical views. (Eden, 2007) was able to categorise those views into three paradigms which can be summarised into rationalists who take a more theoretical position, technocratic for engineering and scientific when natural science is involved. The method used to define in which category the research belongs is determined by why priori or/and posteriori knowledge is obtained. Its actual methods are deductive reasoning, test cases, "formal deduction and scientific experiment" respectively to the three paradigms. By means of reviewing study one and study two with regard to their method of deriving knowledge it's possible to obtain which paradigm they operate in.

This study is considered within the scientific paradigm to hold a rationalist position. This is determined through the fact that the method used to evaluate the computation addition is referred to in an abstraction. Despite, the computation of addition being performed based on the knowledge of quantum mechanics, the researcher does not search to find out his research based on this knowledge, and therefore, it cannot be considered within the scientific paradigm, however in essence quantum computing does fall within this paradigm. Nevertheless this research is not considered to be strictly rationalist because only priori knowledge is sought through deductive logic. This is not so for the technocratic paradigm which is not applied in this study which seeks posteriori knowledge through practical application.

As this study does not base itself on quantum phenomena, but instead uses physical objects in the described model of computation. Nevertheless this research does evaluate how quantum phenomena are used in the application and is critical of it by suggesting an alternative approach. The description of this model of computation is an abstraction of a machine its instructions which is rationalist in nature. This concludes this study to fall within the scientific paradigm as described by (Clear, 2004) and adopts a rationalist theoretical viewpoint.

Epistemology

The research paradigm defines how knowledge is obtained. The concept of how knowing what is true is explained by (Holloway, 1995) to be either a matter of authority, reason, or experience. The authoritive epistemology is divided into two possible forms of authority. These can be omniscient authority and human authority. This study seeks not to justify its findings through authoritive epistemology. Because this study does not obtain the findings form an authorative source. It’s more likely that this epistemology is used in an interpretative paradigm or even in a critical enquiry.

The epistemological stance of experience claims to find what is true through using senses (Holloway, 1995). Out of the wide variety of experience based epistemologies available, of which the most relevant in this context is experimental evidence. This is because experimental evidence can be used within the technocratic paradigm to present posterior knowledge through practical tests. These studies do not engage in practical experiments from which they derive conclusions forming the purpose of the research. Instead, this study refers to a model of computation and presents a method for accessing computational performance. Through the method priory knowledge is derived verifying the postulates of computational performance are right.

Within the epistemological stance reason, it's required to prove truth with deductive logic rules. Having previously identified both studies to fall within this category, this study adopted an epistemology of reason. Through evaluating different approaches to computing addition with a uniform model of computation it's deduced whether there are any differences and reasoned as which of the evaluated approached yields greater performance.

For this research the ultimate objective is to develop an understanding of how performance of computation in particular addition can be improved. Therefore, knowledge derived through this research allows the reader to understand the significance of the computation of addition, different approaches to addition within the classical and quantum paradigm through different forms of computational logic. As the shift to quantum computing is questioned for its approach to how computations is performed, and an alternative approach to the computation of addition is suggested based on quantum mechanics research in state discrimination as a contribution to knowledge.

Ontology

With consideration of what knowledge is in this study, it's required to consider how the world under observation also known as the ontology is defined. It's required to have a thorough understanding of the model of computation, this is an ontologically description. Copeland and Shagrir (2011) evaluated and compared two ontological positions are used to describe models of computation. These two ontological positions are purist and realist which describe the models of computation at different levels. It's significant for the research they present as it has implications for their findings in that under one conception they find different computational ability. They also review Turing's description of his model of computation and find him to fall under the realist conception.

Under the purism conception a model of computation is a mathematical object. It's important to distinguish between the design of what a physical computer would be like and its mathematical abstraction (Copeland and Shagrir, 2011), this is because a mathematical abstraction can multiple different physically implementations. An example of a purist (Shor, 1994). This is because the study of Shor presented a mathematical formulation of a quantum algorithm to perform integer factorisation on a quantum model of computation. The revision of this algorithm hence is expressed mathematically.

This study focuses on the computational task of addition through a model of computation documented in the literature. However the literature found most use a different type of model of computation known as a circuit model. Both the circuit model of computation and the automaton model of computation are within the realist ontology group rather than purist as they do not use a mathematical description. Instead the models of computation us a diagrammatic means of presenting the operations of the approach to addition.

The difference between the circuit model of computation and the automaton model of computation is visible at different levels. The circuit model of computation is a direct representation of the wiring within the hardware used to implement the addition while the automaton model of computation is a machine representation like the circuit model of computation, but differs in its level of abstraction. Therefore the same approach is presented differently.

The current mainstream usage model of computation for addition is the circuit model of computation because it enables intricate optimisation for implementation, while the automaton model of computation is more for academic purposes. This research has adopted the automaton model of computation because the objective of this research is to

evaluate different approaches to addition within different paradigms and with different types of logic.

This research is focused on evaluating the difference in terms of computational performance between computing paradigms. It's important for the model of computation (automation model of computation) used within this research is able to represent each of the computational paradigms. Although the research paradigm of this research is clear about the idea that the research is based on the underlying physics of the computational paradigms. It does not contribute to its development. However the research does evaluate the underlying physics are used to perform the computation of addition more efficiently. This is based on the usage of state discrimination as the primary science of quantum mechanics over the current application of quantum computing based on quantum phenomena of particle duality.

The research therefore evaluated three ontological views of physics and compares those with regards to performance when computing addition. For each of the three views the ontology is based on the most fundamental unite of computation with is the bit, as previously described in classical physics a qubit is either one or zero. Based on particle duality in quantum mechanics under the many world interpretation a quantum bit can represent both one and zero at the same time, through which several researchers have claimed to be able to attain a computational speed-up (Simon, 1997). As an alternative view to the usage of quantum mechanics for the computation of addition the science of state discrimination is suggested through which a finite amount of single states can be represented (Bae and Hwang, 2013). Based on this science an alternative paradigm with a number base equal to the amount of states can be represented opposed to the other binary paradigms.

To compare those three ontological views, it's required to use an automaton model of computation to perform this research. The automaton model of computation brings the computation approach of addition to a higher level of abstraction not attainable by the circuit model of computation as to close a representation of the physical material from which the implementation consists. The level of abstraction provided by the automaton model of computation enables to compare between the different approaches to addition independent of how they would be physically implemented.

Subjectivity

The application of computational complexity theory has a significant set of ontological implication in terms of objectivity and subjectivity. These implications can undermine the epistemological justification of knowledge derived through the application of the methodology. To understand the implication, it's required to understand what constitutes objective measurement of complexity. According to Fioretti (2000) there are two prerequisites to objectivity. The first is to be able to identify the most fundamental required components in the system relevant to the study without the need for in-depth analysis of subcomponents. The second requirement is the links between those components and the action undertaken is defined. For this study a clear explanation is provided of the above stated requirements, which implies an objective measure of complexity is conducted.

It's expected to be the case as according to Fioretti (2000) the discipline of computer science is primarily objective. He states: "the components are objectively given: they are the symbols a computer works with, and they do not arise out of measurement of a physical magnitude". This study is in accordance with this statement, however (Fioretti, 2000) continued by stating "contrary to computer science classical physics does involve a translation of continues and open ended phenomena into symbols". This would imply measures of complexity become subjective. As this study relies to a certain extent on

the concept of quantum phenomena to perform computation which raises the question as to whether the measure of complexity measurement becomes subjective?

Under the condition of a universal consensus within a scientific community that agreed on the components of the system. It's possible to define a computational complexity measurement as objective. With regard to the this study, on-going debate within the research community on the several interpretations of quantum mechanics (Osnaghi et al., 2009) despite the many world interpretation taking a lead within the applied field of computer science. Based on the understanding of this interpretation an objective measure of complexity should are established in this study.

True objectivity within this study remains uncertain. This is based on the fact that the components within the model of computation are agreed upon at the start of the study. This is true for both paradigms and forms of logic in the different approaches to addition. It's when these components predefined that an objective measure of complexity is considered to be subjective to some extent. This means, the measure is objective but the experiment is subjective. Furthermore Fioretti (2000) discussed that being able to meet the requirements to an objective complexity measurement can be done by an algorithm. This means that the measurement of an addition algorithm would have been objective independently of its possible initial flaw suggested by this study.

Fioretti (2000) also stated objective measurement of complexity to retain a subjectivist flavour because it's a machine which is executing the algorithm. This study does have in some sense subjective influence through the model of computation. With regard to the critical evaluation of the approaches to addition and there is no implication towards their finding in terms of accuracy of knowledge found using the computational complicity measurement. It means the researcher is expecting to obtain those findings at the start of the research.

Research methods

Models of computation

This study seeks to evaluate whether computation through a higher number base system would be computationally more efficient than classical computation or even quantum computation based on the multi-universe interpretation of quantum mechanics. To model computation a specific model of computation is required. According to the literature there are two types (Fernández, 2009, Savage, 1998). These are those representing machine instructions and those used to model programming languages; this study is interested in modelling machine instructions. For those models of computation, a further two types are identified, which are logic based on circuit models and automaton models. This research seeks to use universal automation models of computation, so computation between the different paradigms can be accurately compared.

The Turing model of computation is a form of automata. These automata are described through the Chomsky hierarchy, which categorises those models of computation (Chomsky, 1956). The categorisation of those models starts with the universal Turing model of computation at the bottom, which can compute any computational task as long as expressed as a function. Each category in the hierarchy symbolizes how complex a computational task a model of computation can perform. At the highest level of the hierarchy is the finite state model or automaton, as the simplest form of automata. The models of computation can all be visualised as finite state models. As the hierarchy is traversed towards the universal Turing model, it's found that each category is an extended finite model: the Turing model consists of multiple finite models which enables its universality. As the computational task addition is so simple, the finite state

model of computation is used throughout this research to determine computability and computational performance.

Revision of models of computation revealed it impossible to compare each of the ontological views described when performing addition and measuring performance. To establish an objective evaluation a universal model of computation is to be used to represent each of the states that the different ontological views stand for. The automaton model of computation is able to represent each of the different states the three ontological positions can be in and the transitions required as part of the computation that changes the states.

Computational complexity theory

The secondary research method in this research is computational complexity theory. This method is used in conjunction with the automaton model of computation. Through application of a consistent singular method of measurement each of the ontological views presented can be compared in terms of performance. As computational complexity theory is key to the successful evaluation of this research an overview is provided to understand the origin of computational complexity theory, its application and limitations with regards to this research.

Although the old Greeks had a sense of what an efficient algorithm was, this would was limited to the abacus. Computational complexity was something which was unknown to Alan Turing. However the lack of a method to measure computational resources for the Turing model of computation. Out of this requirement computational complexity was born (Fortnow and Homer, 2003). This developed over the years through the addition of classes demonstrating different levels of complexity of which polynomial and exponential are most often used. Within the complexity theory different types of resources such as time and space have been defined as individual complexity classes.

Therefore the time and space complexity per computational task can be specified individually. Each of the complexity classes is specified for the Turing model of computation according to determinism. The understanding between these two complexity classes proved too difficult for to resolve.

The problem of being able to determine an algorithm able to complete a computational task in polynomial time and being able to verify whether the answer is correct provides the answer to the question is the polynomial time complexity class equal to the non-deterministic polynomial complexity class. This problem is an academic interest for which the Clay Mathematics institute currently award an academic prize of one million dollars to who is able to solve the question (Institute, 2000). Rather than the inability of being able to solve the problem of developing an algorithm able to perform a computational task in polynomial time, the usage of an algorithm that is able to verify an answer became significant in solving computational problems within this complexity class (Gill, 1977). This is because by being able to verify an answer allowed usage of probabilistic computational tasks are able to solve some computational problems to some degree of accuracy. By being able to verify the result in polynomial time the computation can be completed multiple times until its correct answer is verified. Through this probabilistic approach it becomes possible to solve non-deterministically hard problems.

Computational complexity theory has proved to be sound through its adaptations to quantum complexity for measuring resource usage (Bernstein and Vazirani, 1997). This complexity class also known as bounded error quantum polynomial time represents the quantum version of the classical probability computational complexity class. The importance for this class comes with the fact two significant algorithms is developed which are able to perform database searching and integer factorisation more efficiently than their classical counterparts. However the complexity of NP and BQP are not

analogues. This means some computational problems which are within the computational boundaries using a quantum model of computation but do not satisfy these criteria when using a classical model of computation. A thorough understanding of computational complexity is required to answer the question: what is the computational performance of a non-binary computational model?

With the purpose of determining how efficiently 'addition' can be computer. The theory of computational complexity is used. Through computational complexity problems can be categorise according to how hard it is to compute them (Papadimitriou, 2003). This can only be determined based on the model of computation used (Fortnow and Homer, 2003). Through computational complexity a set of measures can be applied to represent the taxonomy of the resources required for a predetermined model of computation to perform a computational task. The amount of resources used by the model of computation is expressed as a function. Through this function the required resources used over time can be determined.

In view of the current shift from the classical to the quantum paradigm the measurement of computational performance must be taken into consideration. Within the literature classical computational complexity theory is adapted to quantum computational complexity theory (Watrous, 2008), which includes classes in which it's possible to categorise the complexity of performing computational tasks through a quantum model of computation. This research questions to what extent it's possible to capture any computational speedup through computational complexity theory of the task addition. While most of the limitations of computational complexity theory are discussed in the research methodology chapter, they are addressed below.

Computational complexity is demonstrated to be a coherent theory. This is because throughout the paradigm shift from classical to quantum computing easy adaption took

place. However, a study has demonstrated computational complexity theory does not account for all aspects of a computation (Blakey, 2011). This is an important concept within this research. Furthermore, computational complexity theory is objective in nature and is affected by the subjective considerations of the researcher's experiment (Fioretti, 2000). Therefore, when computational complexity theory is applied, the experiment needs to be well thought out to maintain objectivity. This affects how computational complexity should be applied, and how the performance of addition is measured for the specified models of computation.

Computational complexity uses several different types of metrics. The most common are those bound the computational model by 'time' and/or 'space'. Throughout the literature the circuit model of computation is most often applied. With this model of computation the most used metrics which are frequently cited are gate depth and gate count. This represents the amount of sequential operations required to traverse a complete computation, and the total amount of operation performed throughout the computation. Other metrics included in the quantum paradigm, the amount of 'Qubits' used throughout the computation and the amount of garbage outputs. As a limited number of studies have used the finite state model of computation, these metrics cannot be applied to this research, as they are very specific to the logic model of computation. Therefore the metrics used to measure computational complexity of the finite state model when performing addition is 'time' and 'space'.

Once the computation is evaluated through the universal automation model of computation its performance must be measured. Computational complexity can be used to determine the computational performance of each implementation to compare the performance (Papadimitriou, 2003). However, some considerations must be made during the application of computational complexity. In the first instance is the tests must be applied objectively as possible to define a subjective test as an experiment for

which the result is predetermined (Fioretti, 2000). Further investigation found computational complexity to have limitation (Blakey, 2011). Therefore, it should be considered whether computational complexity does actually demonstrate a computational performance increase through a higher number base system. This could be critical for the outcomes of this research. This is because if it's required to use alternative means of demonstrating the performance difference between the ontological positions then objectivity can be lost.

Research experiment

To evaluate the performance of different approaches to addition an experiment is applied for each approach with one calculation. It's expected to find through performing the same calculation through different approaches the results is determined accordingly.

The experiment simply evaluates the steps that a computation goes through for each of the different approaches to addition evaluated. Three tests are used to evaluate the performance of each approach. Each test evaluates the operations, and test different cases. The three tests are summarised below in binary, decimal and ternary

1. 0101 + 1010 = 5 + 10 = 012 + 101
2. 0101 + 0101 = 5 + 5 = 012 + 012
3. 0111 + 0111 = 7 + 7 = 021 + 021

Each of the three computations is shown below as a different computation in terms of its number base system. This is because for the research experiment that will compare between the classical and quantum research paradigm both binary and ternary computational logic are used. This will reveal whether quantum computation of addition is more efficient than the traditional classical computational paradigm. Also within each of those paradigms the different types of logic are compared in terms of performance.

The research is set with the expectation of multi-value logic enabling higher performance for the computation of addition.

The subsequent comparison within the quantum computing paradigm seeks to evaluate the computational performance between what is binary and a higher number base system. For this experiment number base ten is assumed. The number base would be defined through the amount of states which can be distinguished in a single particle In this case ten states are required for the number base system.

Chapter 4 Analysis

Introduction file

In this chapter the research experiment described in the methodology is implemented. First fundamental logic gates for each paradigm and computational logic are described. Then the adder circuit model of computation is evaluated and abstracted through the automaton model of computation. This allows for a better understanding of the computational performance differences between each paradigm and computational logic for the different approaches to addition.

Evaluation

Based on the revision of the four presented full adders, a comparison is made to determine *the most efficient approach for a computer to perform addition.* The first comparison demonstrated between the full classical and the full quantum adder, are similar in underlying logic. The comparison is performed through an automata model of computation through which both full adders can be compared. This means there is no constraints of hardware that needs to be considered in adders. A subsequent comparison is made between the Binary based adders and the classical Ternary adder to develop an understanding of performance implications, when using adders with different number base systems.

Binary automaton adder

The Binary automaton adder depicted below in figure 20 is an abstraction of both the *classical full adder* and *quantum full adder*. Through abstraction the physical constraints are removed. Each of the adders performs addition through similar logic. So, it's concluded both are equal in computational performance. This means that both adders have an equal number of operations when modelled by the automaton model of computation.

The Binary automaton adder depicted in figure twenty consists out of five states between, which the automaton can switch provided there is a connection. Each connection has a box in which the two values are added recorded. After the forward slash a carry from the previous operation is recorded. Once an operation has occurred, a state transition takes place. Every operation involves a state transition even if that is to the same state.

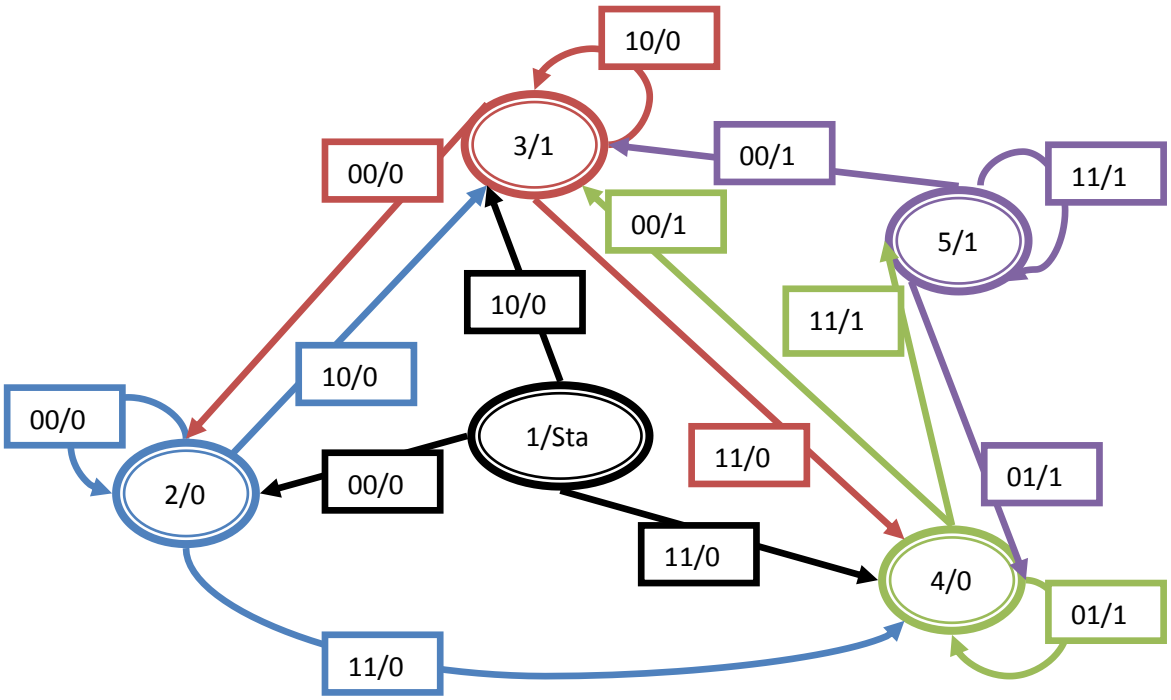

Figure 20 Binary automaton adder

Computation

For the experiment three computations are performed. These computations are based on *three different calculations* that will demonstrate a computation, when there are no carry values; the other two computations involve some *carry values*. Through these experiments demonstrated how efficient these approach is in performing *addition* by the amount of computational steps required.

0101 + 1010 = 5 + 10 = 15

Table 25: Binary automaton computational steps 5 + 10

Computational step	Computational state	Input / carry	Carry	Output	Result
1	1	10/0	0	1	1
2	3	01/0	0	1	3
3	3	10/0	0	1	7
4	3	01/0	0	1	15

0101 + 0101 = 5 + 5 = 10

Table 26 Binary automaton computational steps 5 + 5

Computational step	Computational state	Input	Carry	Output	Result
1	1	11/0	1	0	0
2	4	00/1	0	1	2
3	3	11/0	1	0	2
4	4	00/1	0	1	10

0111 + 0111 = 7 + 7 = 14

Table 27 Binary automaton computational steps 7 + 7

Computational step	Computational state	Input	Carry	Output	Result
1	1	11/0	1	0	0
2	4	11/1	1	1	2
3	5	11/1	1	1	6
4	5	00/1	0	1	14

Ternary automaton adder

The Ternary automaton adder depicted below in figure 21 demonstrates how Ternary addition can be performed without physical constraint. Although more complicated and harder to follow, the ternary automaton adder functions like the Binary automaton adder depicted in figure 20. The difference being, that there are more states and inputs for addition reflect the Ternary number system.

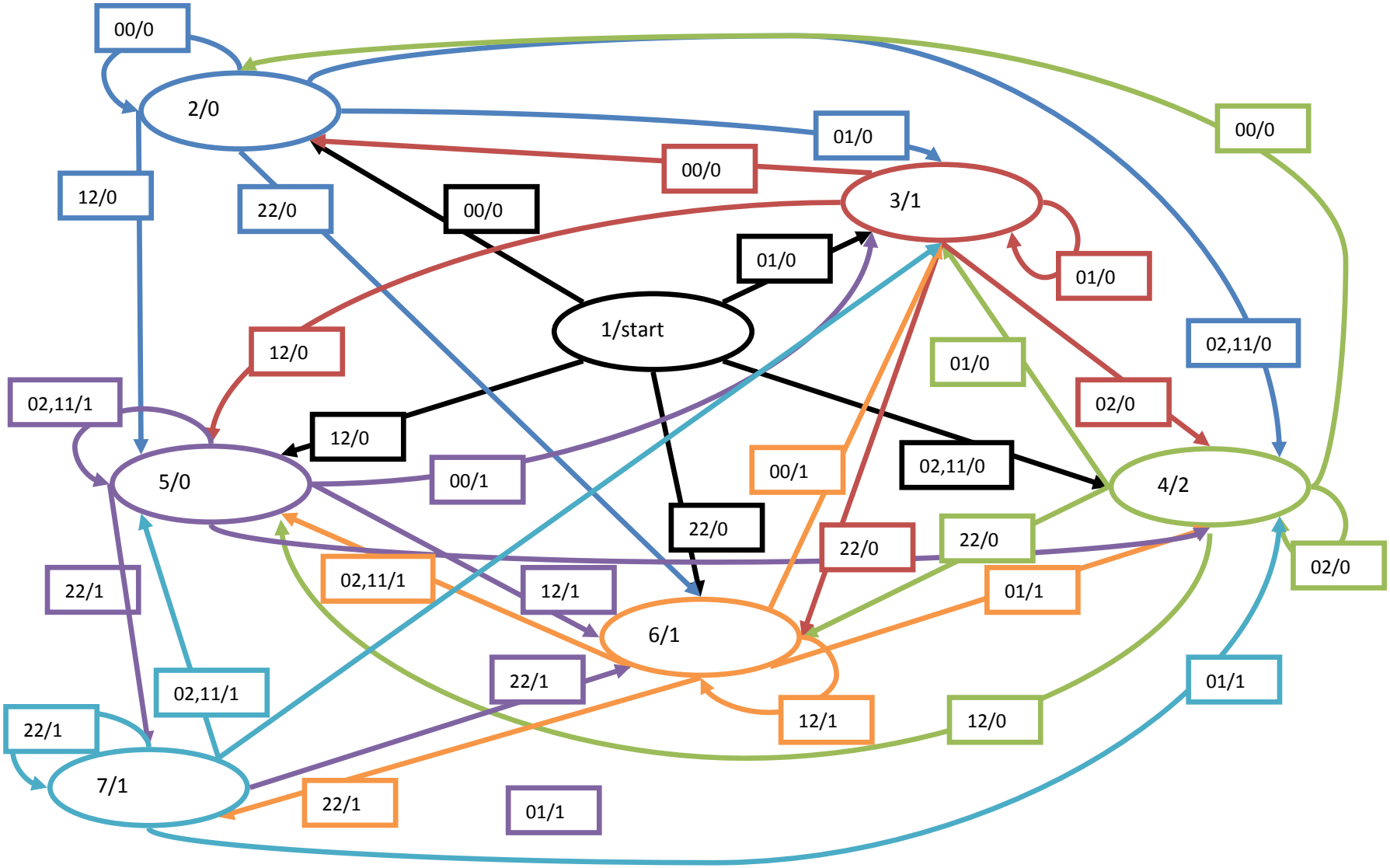

Figure 21 Ternary automaton adder

Computation

The same three test cases are used to evaluate the Ternary adder automaton as for the binary automaton. Using the same type of computational model and the same test cases any significant difference between both the approaches is verifiable. Through this method the significance of performing addition with differences in the radix of the number base system are evaluated.

012 + 101 = 5 + 10 = 15

Table 28 Ternary automaton computational steps 5 + 10

Computational step	Computational state	Input / carry	Carry	Output	Result
1	1	21/0	1	0	0
2	5	10/1	0	2	6
3	3	01/0	0	1	10

012 + 012 = 5 + 5 = 10

Table 29 Ternary automaton computational steps 5 + 5

Computational step	Computational state	Input	Carry	Output	Result
1	1	22/0	1	1	1
2	6	11/1	1	0	1
3	3	00/1	0	1	10

021 + 021 = 7 + 7 = 14

Table 30 Ternary automaton computational steps 7 + 7

Computational step	Computational state	Input	Carry	Output	Result
1	1	11/0	0	2	2
2	4	22/1	1	1	5
3	6	00/1	0	1	14

Decimal adder automaton

Based on the revision of full adders in different paradigms with different number base systems, abstractions are made into automaton models of computation to compare between them. Some improvement of computational efficiency is observed between the *classical full adder* and the *classical Ternary adder*. Further investigation into the increase in number base results in *improved performance of computing addition*. To evaluate whether this holds true, a Decimal adder automaton model of computation is depicted through which a further experiment is conducted using the same test cases.

The Decimal automaton adder functions similarly to the Binary and Ternary automaton adders. For complexity reasons the Decimal automation adder is depicted in part. In total twenty one states are exist, and have a connector between most states, this high amount of connectors is not depictable in figure 22 below. Based on figures 20 and 21 it's easy to understand how the Decimal automaton adder would fit together.

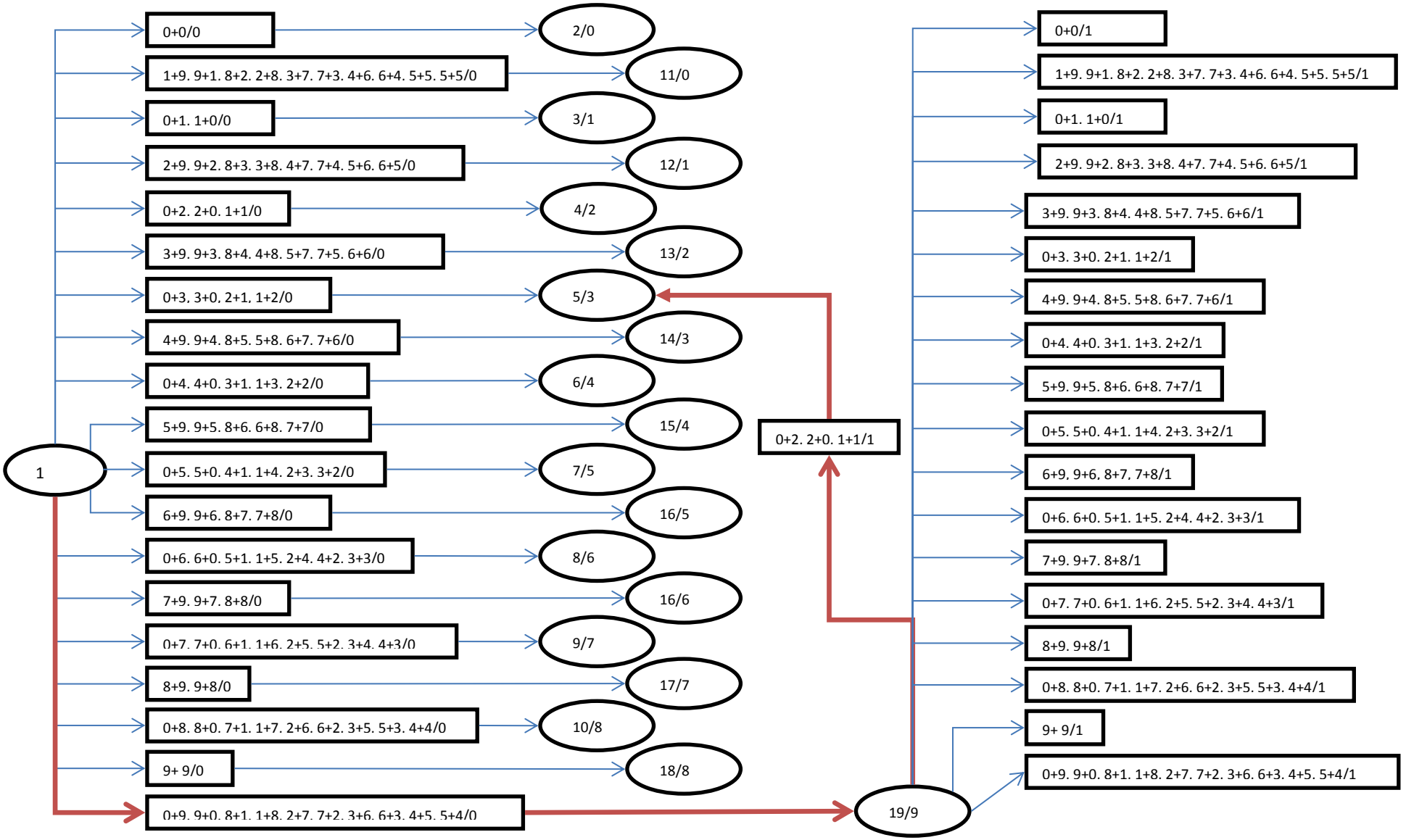

Figure 22 Decimal adder automaton

Computation

For consistency the same three test cases are used in this experiment to measure the performance of the Decimal automaton adder. Through using the same test cases, it's possible to compare the results against the previous computations that where performed using the B*inary* and T*ernary automaton model of computation*. The tables below demonstrate the steps of the computation include the computational step, which is just a count of the amount of operations. Computational state signifies where in the automaton model the computation is. Input and carry defines the data used to perform the computation. Output is the result of the operation and the result is the overall outcome of the computation.

05 + 10 = 15

Table 31 Decimal automaton computational steps 5 + 10

Computational step	Computational state	Input / carry	Carry	Output	Result
1	1	50/0	0	5	5
2	7	01/1	0	1	15

05 + 05 = 10

Table 32 Decimal automaton computational steps 5 + 5

Computational step	Computational state	Input	Carry	Output	Result
1	1	55/0	1	0	0
2	11	00/1	0	1	10

07 + 07 = 14

Table 33 Decimal automaton computational steps 7 + 7

Computational step	Computational state	Input	Carry	Output	Result
1	1	77/0	0	2	2
2	15	00/1	0	1	14

Summary and findings

The evaluation of full adders within this research found several significant findings. These are the *similarities* between the *different types of full adders*, to the extent that the *classical full adder* and *quantum full adder* can be generalised as the *same automaton adder*. Furthermore, *significant are similarities* in the way that the *full classical Ternary* adder is modified based on the fundamental concept of *modulo two into modulo three.* Although it's not possible to prove the same similarities between the *classical Ternary adder and quantum Ternary adder* as with their *classical counterparts*. This is believed to be due to the *constraints of physical realisation*. In principles the *quantum Ternary adder* should be reducible to the same t*ernary automaton adder.*

This lead the research into comparing the adders based on the radix of their number base system. The first comparison between *the Binary automaton adder of computation* and the T*ernary automaton model of computation* presented **a few differences**. This is an increase in *complexity* depicted in figure 21. This has more connectors between the states in the automaton compared to its Binary alternative. When evaluating the actual computation it's found that the Binary automaton requires *four steps* of computation while the Ternary automaton is able to perform the computation in *three steps.* While

this is an improvement, according to computational complexity, it's found that the output is linearly bounded to the input. The B*inary automaton* has *four input* and so *four outputs*, the reason the T*ernary automaton* has three computational steps is because the inputs can be represented in less significant values.

Through the usage of the method computational complexity it's possible to evaluate and compare the computational performance between models of computation based on different number base systems. Significantly, the higher the number base system the lower the number of computational steps required to perform the computation. For the first model based on the Binary number system four steps where required. The second based on Ternary required three steps. And the final model based on Decimal required two steps to compute a simple addition of five and ten.

As the findings indicated a difference for computational steps required in relation to the number base used. It's questioned whether a further experiment could be performed to find out how significant this difference would be on large scale computation. As it's not be possible to illustrate in a diagram a computation with greater values. It's hypothesised how great a value can be computed in each of the number base systems.

The assumption is that the growth rate of output per computational step increases according to the base unit of the radix used within the computational model. This is based on the findings in the experiment above. Below a table is presented in which the growth per number systems is displayed in sequence with the computational steps. The growth rate of output is the radix to the power of the computational step expressed in big O notation.

- Binary → $O(radix(2)^N)$
- Ternary → $O(radix(3)^N)$
- Decimal → $O(radix(10)^N)$

Table 34 Growth rate of output per computational step

Computational step	**Binary**	**Ternary**	**Decimal**
1	2	3	10
2	4	9	100
3	8	27	1000
4	16	81	10000
5	32	243	100000
6	64	729	1000000
7	128	2187	10000000

The forecast of expected outcomes of a large scale implementation for each of the presented models of computation based on increasing number base systems is captured in the graphs below. Figure 23 demonstrates the total amount of output per computational step. While in the second figure 23 the output per computational step is demonstrated as a percentage. This shows that for Binary and Ternary models of computation the gap for step one is 15%, step 2 10%, and step three 5%. Between Ternary and Decimal the gap is 65% for the first step, step 2 is 90% and the tird step is 95%. In all cases the trend is an increasing gap between the outputs of each of the models of computation.

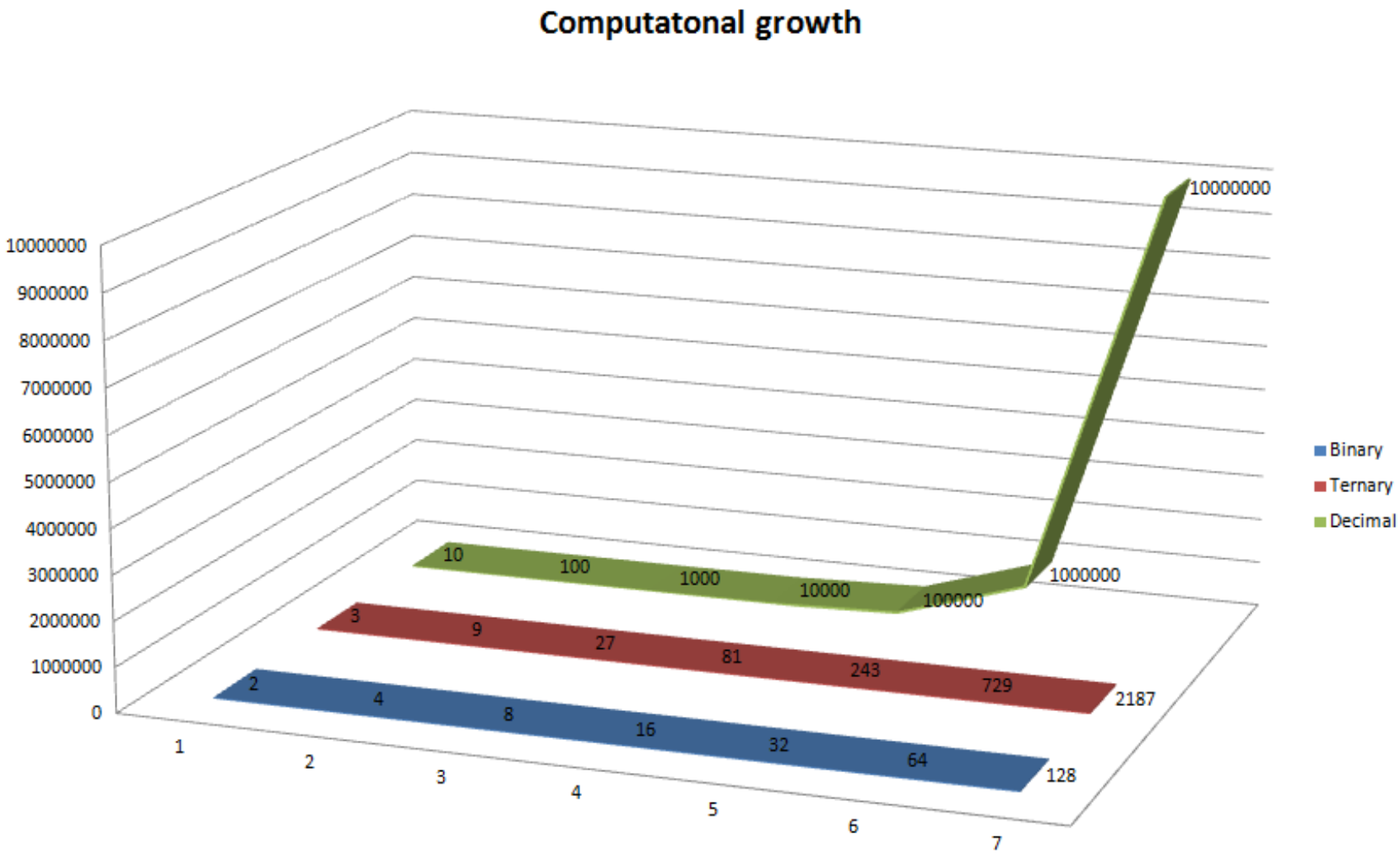

Figure 23 Forecast of growth output

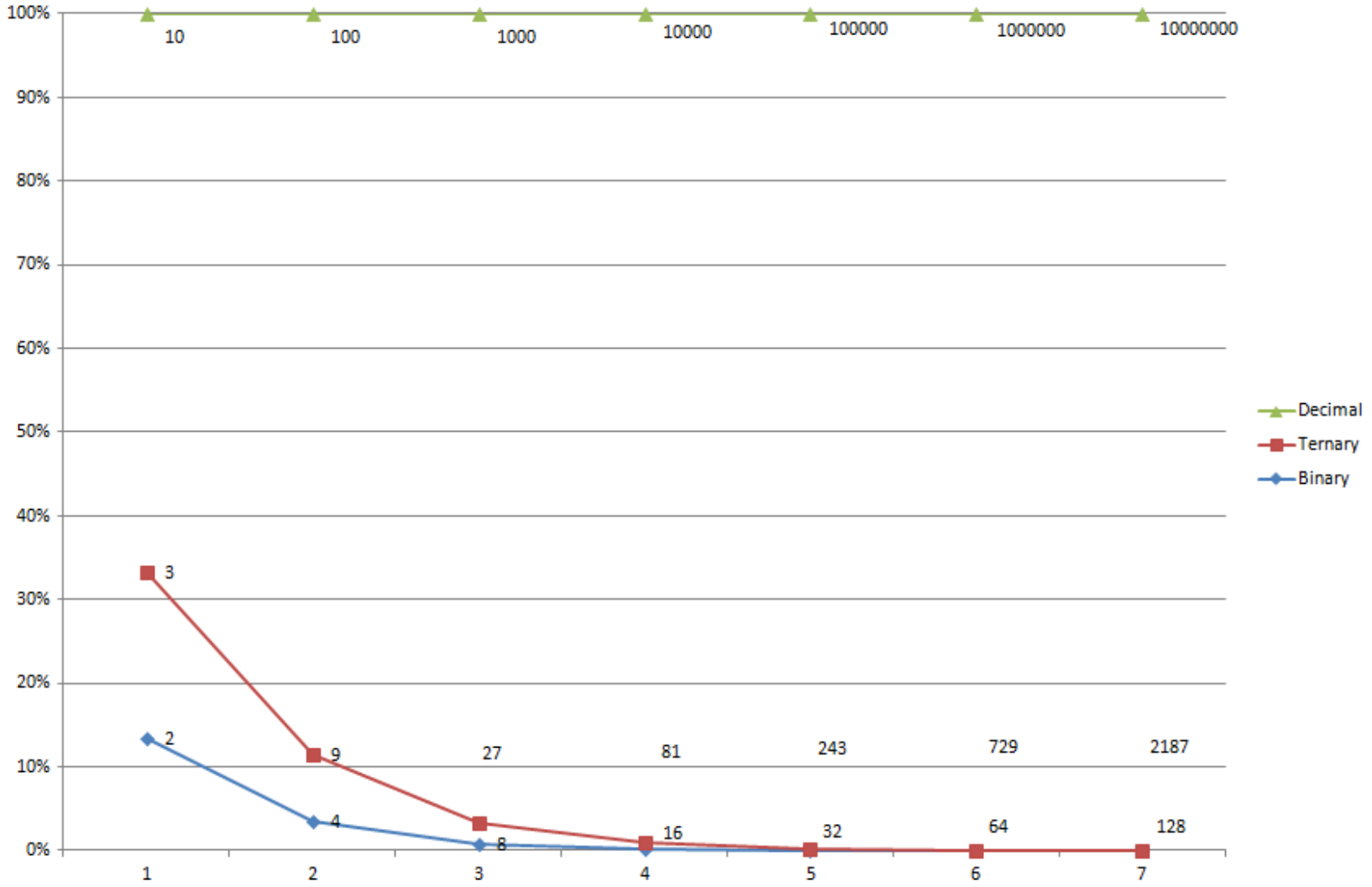

Figure 24 Forecast of growth output as growth

Chapter 5 Conclusion

Answer to research questions

The first research question Sought to find out *what the available models of computations are*. Secondary in this question is to identify *the most suitable models of computation for the purpose of computing addition*.

During the evaluation of the literature the following types of models of computation are identified:

- logic circuit models
- Automaton model of computation (finite state machines)
- Turing model of computation
- Formal language model of computation (Lambda calculus)
- Object oriented model of computation
- Quantum model of computation

Different forms of computation have historically taken place, compared to modern day models of computation. These approaches are significantly distinguished from the models outlined above. This is based on the idea of computation not being binary as in modern models of computation.

The research focuses on models of computation that model machines rather than machine instructions. Identified example of machine instruction models are: formal language and object oriented models of computation. These models of computation are used to model programming. So these models of computation are identified as unsuitable to the current research. Instead this research requires models of computation which *model the functioning of machines*.

The *circuit model of computation* is particularly relevant to this research. Most of the literature focused around the computation of *addition* is presented through this model of computation. However, the circuit model of computation is found not suitable for implementing this research. This is because it's specific to technology. As this technology is significantly different between paradigms it's not possible to compare approaches between different paradigms or forms of computational logic. The automaton model of computation on the other hand is to be able to model a computation independent of its physical existence. Therefore, it's considered more suitable for this research. The automaton model of computation is a subset of the *Turing model* of computation. This research seeks to evaluate the *task of addition* which can be achieved through *a finite automaton*. The Turing model of computation is more suitable for general purpose computation for which multiple finite automaton are required.

As found in the literature the *circuit model* of computation is extended from the *classical computing paradigm into the quantum paradigm* to *model quantum computation.* In this research, the automaton model of computation is extended to the quantum paradigm. This means both paradigms with *binary and multi-value logic* can be evaluated for performance when performing the *task of addition*. More importantly it's possible to compare between the findings of each evaluation. Therefore, the automaton model of computation is considered the most suitable model of computation for this research.

The second research question drives the research forth to investigate the current literature with regard to how researchers have proposed to perform addition. Within the different given computing paradigms the research seeks to develop an understanding of the different approaches to addition for each form of computational logic. Furthermore, the evaluation the research seeks to develop an understanding of the performance, for each of the different

approaches to addition, and how the researches have expressed performance and measured it.

A section within the literature review chapter is devoted to answering the greater part of this research question. This is in the summary of the different approaches to computation section. The interesting aspect of this evaluation is the performance measurement. Most researchers established a metric in term of performance expressed in computational complexity, although several research papers did not consider this factor. In several cases where a form of measurement applied, it's not being compared as some studies investigate the number of logic gates used, other circuit depth, etc. Therefore, the studies are not comparable in terms of performance measurement.

In the classical paradigm *carry save adder* and *conditional sum adder* are most efficient. Each of them varies in performance depending on how large scale the implementation is. For multi-value logic opposed to binary Boolean logic it's expected to find reported findings demonstrating a performance increase; however no valid findings are identified in the literature. Even so, the most significant findings are identified in the quantum computing paradigm. For binary Boolean logic based approached in the quantum paradigm similar finding to the classical counterpart found most approaches. The *conditional sum adder* is significantly faster in the quantum paradigm for which it's rated as logarithmic. The Kd mesh adder performance increases as the computation takes place. The final group under investigation is multi-value logic addition within the quantum paradigm.

As identified in the first research question, it's not possible to compare this metrics for different paradigms and computational logic because of inconsistency. But, the same methods used to measure and express the computational performance of the different

approaches to addition. Throughout the literature and so this research has adopted computational complexity theory.

Through the usage of the automaton model of computation and computational complexity, it's possible to evaluate the different approaches of *addition* in different *paradigms.* For each of the *computational logics* defined. This lead to the conclusion: *no performance difference between computational paradigms for performing addition*. However, it's found through *ternary logic* **possible** to *improve computational performance over binary computation.*

The final research question revolved about questioning the findings of this research. The aim is to better understand how further research can improve and increase computational performance of addition.

This research identified performance of computing addition is related to the radix of the performed computation. As demonstrated in the last experiment, a model of computation operates with radix ten is able to perform the computation in fewer steps and therefore be more efficient. This efficiency has become visible for very small computations of addition involving numbers with two values. However this becomes more significant as the values in the number increase. In the performance graph (Figure 23 & 24) it's depicted for binary logic computation. To add a value up to *128* in four computational steps, *1287* for ternary logic and *10000000* with a decimal computation. This is a great difference between the radix, the growth curve is increasingly steeper.

Within this research study an attempt is made to standardise the approach to measuring the performance of computation. As concluded from the literature, several studies have reported a form of performance metrics; however, even between those studies it's not possible to compare performance. This is because some refer to the number of gates used, other to the amount of computational steps, while several didn't report any performance metrics. In this study the focus is placed on the number of computational steps required to perform a specific computation. For which it's found that with higher number base models of computation it's possible to perform the same computation of addition in less computational steps.

Therefore, further investigation shows how much more can be computed using a model of computation with a higher number base system. On a large scale a significantly greater output per computational steps is attained. However, it's impossible to relay back to the initial findings in the literature reported to be able to perform addition in different categories such as O(N), O(log *N*), O($\sqrt{N}$), etc according to the different approaches and paradigms used. To be able to compare the findings of this study and other studies, it's required to develop a physical device able to perform modular arithmetic on number base encoded qubits. Once these logic gates are developed a circuit can then be used to compare the performance.

It's important to note: the current research is based on the assumption that qubits can be held in at least ten different distinct states as described in the literature as state discrimination. This is distinct from previous research which looks at performing addition based on the qubit holding two states simultaneously. This is important because it's fundamentally different in how quantum mechanics is applied in computation. Also, this research is based on modular arithmetic for the three experiments. However, in the

literature different approaches used such as the quantum fourier transformer, binary tree, mesh or cube structures.

Contribution to knowledge

The contribution to knowledge is twofold within this research. **Firstly** the researcher has systematically evaluated the existing literature to find out what the different approaches are for performing addition, and found no single approach to this computation.

Secondly the research found several different types of models of computation with different purposes. As part of this research, it's required to compare and contrast the performance between the different approaches of addition independent of the research paradigm or the computational logic. Although the literature uses predominantly circuit models of computation, the research found through an automaton model of computation it's possible to objectively compare those approaches to addition.

The research findings to this research are considered a contribution to knowledge. This is because the research informs the reader that the computation of addition is not improved in performance by quantum computing. But, through different usage of quantum computing to compute through a higher radix, it's possible to gain computation performance. Instead of current quantum computing which exploits the quantum phenomena of state duality, this research advocated state discrimination should be used. This implies the model of computation its most fundamental units are able to represent a number of finite unique states used to form a number base system. Instead of quantum parallelism through state duality or two states of *one or zero* in classical computing.

Further research

The researcher calls other researchers to evaluate this research and to perform similar studies such to verify the findings and inconsistencies highlighted. This can be through setting up modified experiments. Further research can be conducted through evaluating more specific approaches to addition such as very specific types of adders. This extended the research in this thesis that evaluated the full adders for each paradigm and computational logic.

Further research can be performed to do a practical implementation of this theoretical research. This involves identifying the right physical medium to implement a decimal model of computation to find out whether the findings of this research are theoretical. This research can then be extended by developing logic gates capable of performing addition with a higher number base.

It's interesting to develop adiabatic logic gates for a higher number base model of computation able to perform computation with greater performance. The advantage is computing great numbers with low usage of energy. This would be through using adiabatic gates where the amount of input energy is equal to the output, and therefore there is no dissipation of electricity. This means no loss of energy and a possibly environmentally friendly (green) solution to computation.

As highlighted at the start of the research, the quantum computing paradigm is found to enable very specific computational tasks to be performed more efficiently. This is based on the application of quantum phenomena. Of particular interest is the computational task of integer factorisation demonstrated on a small scale to be more efficient opposed to the classical computing paradigm. The research question seeks to find whether it's possible for a model of computation with a higher number base (based on state discrimination) to be

more efficient. At least in theory to compute integer factorisation more efficiently than a model of computation based on the current quantum mechanics interpretation.

Other areas of interest which are identified during the revision of the research are the development of logic gates based on the concepts of state discrimination to perform computation with a higher number base. This can then be built on by developing circuits and be used to compare against other implementation. Further improvements can then be made to optimise the circuits be looking at different approaches instead of modular arithmetic, such as fourier transform, binary tries, or other approaches.

References

A. RIZVI, A., ZAHEER, K. & ZUBAIRY, M. S. 1991. Design of ternary half-adder and -subtractor using frequency modulation in grating structures. Optics Communications, 84, 247-250.

AUDENAERT, K. M. R., MOSONYI, M. & VERSTRAETE, F. 2012. Quantum state discrimination bounds for finite sample size. Arxiv preprint arXiv:1204.0711.

BAE, J. & HWANG, W.-Y. 2013. Minimum-error discrimination of qubit states: Methods, solutions, and properties. Physical Review A, 87, 012334.

BARBOSA, G. A. 2006. Quantum half-adder. Physical Review A, 73, 052321.

BARENCO, A. E., A. SANPERA, A. & MACHIAVELLO, C. 1996. A Short Introduction to Quantum Computation [Online]. Available: http://www.qubit.org/tutorials/25-quantum-computing.html.

BECKMAN, D., CHARI, A. N., DEVABHAKTUNI, S. & PRESKILL, J. 1996. Efficient networks for quantum factoring. Physical Review A, 54, 1034.

BEDRIJ, O. 1962. Carry-select adder. Electronic Computers, IRE Transactions on, 340-346.

BERGOU, J. A., FUTSCHIK, U. & FELDMAN, E. 2012. Optimal Unambiguous Discrimination of Pure Quantum States. Physical review letters, 108, 250502.

BERMAN, G., DOOLEN, G., LOPEZ, G. & TSIFRINOVICH, V. 2001. A quantum full adder for a scalable nuclear spin quantum computer. Arxiv preprint quant-ph/0105133.

BERNSTEIN, E. & VAZIRANI, U. 1997. Quantum complexity theory. SIAM Journal on Computing, 26, 1411-1473.

BIRKHOFF, G. & VON NEUMANN, J. 1936. The logic of quantum mechanics. Annals of mathematics, 823-843.

BLAKEY, E. 2011. Computational Complexity in Non-Turing Models of Computation:: The What, the Why and the How. Electronic Notes in Theoretical Computer Science, 270, 17-28.

BORRELLI, M., HAIKKA, P., DE CHIARA, G. & MANISCALCO, S. 2013. Non-Markovian qubit dynamics induced by Coulomb crystals. Physical Review A, 88, 010101.

BROMLEY, A. 1998. Charles Babbage's Analytical Engine of 1938. IEEE Annals of the History of Computing, 20, 29-45.

BURGESS, N. Accelerated carry-skip adders with low hardware cost. Signals, Systems and Computers, 2001. Conference Record of the Thirty-Fifth Asilomar Conference on, 4-7 Nov. 2001 2001. 852-856 vol.1.

CHATTOPADHYAY, T., TARAPHDAR, C. & NATH ROY, J. 2009. Quaternary Galois field adder based all-optical multivalued logic circuits. Applied optics, 48, E35-E44.

CHAU, H. 1999. Quantum convolutional error correction codes. Quantum Computing and Quantum Communications, 314-324.

CHENG, K. W. & TSENG, C. C. 2002. Quantum full adder and subtractor. Electronics Letters, 38, 1343-1344.

CHO, K. 2003. Conditional sum adder. EP20020254426 20020625.

CHOI, B. S. & VAN METER, R. 2008. On the Effect of Quantum Interaction Distance on Quantum Addition Circuits. Arxiv preprint arXiv:0809.4317.

CHOMSKY, N. 1956. Three models for the description of language. Information Theory, IRE Transactions on, 2, 113-124.

CLEAR, T. 2004. Critical enquiry in computer science education. Computer Science Education Research: The Field and The Endeavour, Routledge Falmer, Taylor & Francis Group, London, 101-125.

COPELAND, J. & SHAGRIR, O. 2011. Do Accelerating Turing Machines Compute the Uncomputable? Minds and Machines, 21, 221-239.

CORKER, D., ELLSMORE,P., ABDULLAH,F.,HOWLETT, I.,. 2005. Commercial Prospects for Quantum Information Processing. Available: qserver.usc.edu/group/wp.../commercial-prospects-for-qip-v1-1.pdf.

CUCCARO, S. A., DRAPER, T. G., KUTIN, S. A. & MOULTON, D. P. 2004. A new quantum ripple-carry addition circuit. Arxiv preprint quant-ph/0410184.

DEUTSCH, D. 1985. Quantum theory, the Church-Turing principle and the universal quantum computer. Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences, 400, 97-117.

DEUTSCH, D. & JOZSA, R. 1992. Rapid solution of problems by quantum computation. Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences, 439, 553-558.

DORAN, R. 1988. Variants of an improved carry look-ahead adder. Computers, IEEE Transactions on, 37, 1110-1113.

DORNAJAFI, M., WATKINS, S. E., COOPER, B. & BALES, M. R. Performance of a quaternary logic design. 2008. IEEE, 1-6.

DRAPER, T. G. 2000. Addition on a quantum computer. Arxiv preprint quant-ph/0008033.

DRAPER, T. G., KUTIN, S. A., RAINS, E. M. & SVORE, K. M. 2006. A logarithmic-depth quantum carry-lookahead adder. Quantum Information & Computation, 6, 351-369.

DUBROVA, E. Multiple-valued logic in vlsi: Challenges and opportunities. 1999. 340-350.

DUBROVA, E. 2002. Multiple-valued logic in VLSI design. Multiple-Valued Logic, An International Journal.

DUBROVA, E., JAMAL, Y. & MATHEW, J. Non-silicon non-binary computing: Why not. 2002. 23-29.

DUWELL, A. 2007. The Many-Worlds Interpretation and Quantum Computation. Philosophy of Science, 74, 1007-1018.

EDEN, A. H. 2007. Three paradigms of computer science. Minds and Machines, 17, 135-167.

FERNÁNDEZ, M. 2009. Models of Computation: An Introduction to Computability Theory, Springer-Verlag New York Inc.

FEYNMAN, R. P. 1982. Simulating physics with computers. International journal of theoretical physics, 21, 467-488.

FIORETTI, G. 2000. A subjective measure of complexity. Advances in Complex Systems, 2, 349-370.

FORTNOW, L. & HOMER, S. 2003. A short history of computational complexity. Bulletin of the EATCS, 80, 95-133.

GAIDHANI, M. Y. A. & KALBANDE, M. M. N. Design of Some Useful Logic Blocks Using Quaternary Algebra.

GANG, W., LI, C. & QIN, L. 2009. Ternary logic circuit design based on single electron transistors. Journal of Semiconductors, 30, 025011.

GILL, J. 1977. Computational complexity of probabilistic Turing machines. SIAM Journal on Computing, 6, 675-695.

GOLUBEV, D. S. & ZAIKIN, A. D. 1998. Interaction and quantum decoherence. Physica B: Condensed Matter, 255, 164-178.

GONZALEZ, A. F. & MAZUMDER, P. 1998. Multiple-valued signed digit adder using negative differential resistance devices. Computers, IEEE Transactions on, 47, 947-959.

GOSSETT, P. 1998. Quantum carry-save arithmetic. Arxiv preprint quant-ph/9808061.

GOTTESMAN, D. 1997. Stabilizer codes and quantum error correction. Arxiv preprint quant-ph/9705052.

GOTTWALD, S. 2005. Many-Valued Logics. Available: http://www.uni-leipzig.de/~logik/gottwald/SGforDJ.pdf.

GREECHIE, R. J. 1981. A non-standard quantum logic with a strong set of states. Current issues in quantum logic, 8, 375-380.

GROVER, L. K. A fast quantum mechanical algorithm for database search. Proceedings of the 28th Annual ACM Symposium on Theory of Computing, 1996. ACM, 212-219.

HOLLOWAY, C. M. 1995. Software engineering and epistemology. ACM SIGSOFT Software Engineering Notes, 20, 20-21.

HSIEH, M. H., DEVETAK, I. & BRUN, T. 2007. General entanglement-assisted quantum error-correcting codes. Physical Review A, 76, 062313.

HUNG, W. N. N., SONG, X., YANG, G., YANG, J. & PERKOWSKI, M. Quantum logic synthesis by symbolic reachability analysis. In Proceedings of the 41st Design Automation Conference, 2004 San Diego, CA,. ACM, 838-841.

HURLEY, P. J. 2006. A concise introduction to logic, Wadsworth Publishing Company.

INSTITUTE, C. M. 2000. Millennium prize problems. Clay Mathematics Institute

ISLAM, M., KARIM, M. R., MAHMUD, A. A. & BABU, H. M. 2010. Variable block carry skip logic using reversible gates. Arxiv preprint arXiv:1008.3352.

JAHANGIR, I. & DAS, A. On the design of quaternary comparators. 2010. IEEE, 241-246.

KETELAARS, N. 2001. Pascal's Calculator. AIMe Magazine.

KHAN, M. H. A. Quantum realization of ternary adder circuits. Proceedings of Third International Conference on Electrical and Computer Engineering, 2004a.

KHAN, M. H. A. Quantum realization of ternary adder circuits. 2004b.

KHAN, M. H. A. 2008. A recursive method for synthesizing quantum/reversible quaternary parallel adder/subtractor with look-ahead carry. Journal of Systems Architecture, 54, 1113-1121.

KHAN, M. H. A. & PERKOWSKI, M. A. 2007. Quantum ternary parallel adder/subtractor with partially-look-ahead carry. Journal of Systems Architecture, 53, 453-464.

KIDWELL, P., WILLIAMS, M. 1992. The Calculating Machines: Their history and development, USA, Massachusetts Institute of Technology and Tomash Publishers.

KNAUER, K. 1989. Ripple-carry adder. US patent application.

KNOWLES, S. A family of adders. Proc. 15th IEEE symposium on Computer Arithmetic, 2001. IEEE, 177-182.

KOBAYASHI, Y., SATOH, A. & MUNETOH, S. 2004. Carry skip adder. US patent application.

LEININGER, J. C. & TAYLOR, G. P. 1978. Carry save adder. US patent application.

LLOYDS , S. 2008. Riding D-wave. Available: http://www.signallake.com/innovation/RidingD-Wave042408.pdf.

LUDWIG, G. & HEIN, C. A. 1985. Foundations of quantum mechanics, Springer-Verlag.

MACKEY, G. & BENJAMIN, W. 1967. Mathematical foundations of quantum mechanics. Bull. Amer. Math. Soc. 73 (1967), 499-500. DOI: 10.1090/S0002-9904-1967-11717-8 PII: S, 2, 11717-8.

MERMIN, N. D. 2003. Copenhagen computation. Studies In History and Philosophy of Science Part B: Studies In History and Philosophy of Modern Physics, 34, 511-522.

METER III, R. D. V. 2006. Architecture of a quantum multicomputer optimized for Shor's factoring algorithm. Arxiv preprint quant-ph/0607065.

MINGOTO, C. R., JR. A Quaternary Half-Adder Using Current-Mode Operation with Bipolar Transistors. Multiple-Valued Logic, 2006. ISMVL 2006. 36th International Symposium on, 17-20 May 2006 2006. 15-15.

NEEDLES, W. M. 1990. Manchester carry adder circuit. US patent application.

NORDHAUS, W. 2001a. The progress of computing.

NORDHAUS, W. 2001b. The progress of computing. Available: nordhaus.econ.yale.edu/prog_030402_all.pdf.

NORMAN, R. H. 1960. Binary half adder circuit. Google Patents.

NOWICK, S. M. Design of a low-latency asynchronous adder using speculative completion. IEE Proceedings - Computers and Digital Techniques, 1996. IET, 301-307.

OKLOBDZIJA, V. G., ZEYDEL, B. R., DAO, H., MATHEW, S. & RAM, K. Energy-delay estimation technique for high-performance microprocessor VLSI adders. Computer Arithmetic, 2003. Proceedings. 16th IEEE Symposium on, 15-18 June 2003 2003. 272-279.

OSNAGHI, S., FREITAS, F. & FREIRE JR, O. 2009. The origin of the Everettian heresy. Studies In History and Philosophy of Science Part B: Studies In History and Philosophy of Modern Physics, 40, 97-123.

PAI, Y. T. & CHEN, Y. K. The fastest carry lookahead adder. 2004. IEEE, 434-436.

PAPADIMITRIOU, C. H. 2003. Computational complexity, John Wiley and Sons Ltd.

RIZVI, A. A., ZAHEER, K. & ZUBAIRY, M. S. 1991. Design of ternary half-adder and -subtractor using frequency modulation in grating structures. Optics Communications, 84, 247-250.

SAVAGE, J. E. 1998. Models of computation, Addison-Wesley Reading, MA.

SCHMELZER, I. 2011. Pure quantum interpretations are not viable. Foundations of Physics, 41, 159-177.

SHADBOLT, P., VERDE, M., PERUZZO, A., POLITI, A., LAING, A., LOBINO, M., MATTHEWS, J., THOMPSON, M. & O'BRIEN, J. 2011. Generating, manipulating and measuring entanglement and mixture with a reconfigurable photonic circuit. Nature Photonics, 6, 45-49.

SHAO, Z. 2008. Boolean Algebra. In: SCIENCE, Y. U. D. O. C. (ed.) Chapter Two.

SHOR, P. W. Algorithms for quantum computation: discrete logarithms and factoring. In: GOLDWASSER, S., ed. Proceedings of the 35th

Symposium on Foundations of Computer Science, 1994 Los Alamitos. IEEE, 124-134.

SIMON, D. R. 1997. On the power of quantum computation. SIAM Journal on Computing, 26, 1474-1483.

SRIVASTAVA, A. & VENKATAPATHY, K. 1996. Design and implementation of a low power ternary full adder. VLSI Design, 4, 75-81.

STEANE, A. M. 1998. Quantum error correction. Introduction to quantum computation and information, 184.

SWADE, D. & BABBAGE, C. 2001. Difference Engine: Charles Babbage and the Quest to Build the First Computer, Viking Penguin.

TAKEUTI, G. 1981. Quantum set theory, inCurrent Issues in Quantum Logic, New York, Plenum Press.

THOIDIS, I., SOUDRIS, D., FERNANDEZ, J. & THANAILAKIS, A. The circuit design of multiple-valued logic voltage-mode adders. 2001. IEEE, 162-165 vol. 4.

TRISETYARSO, A. & VAN METER, R. 2009. Circuit design for a measurement-based quantum carry-lookahead adder. Arxiv preprint arXiv:0903.0748.

TRUESDELL, L. E. 1965. The development of punch card tabulation in the Bureau of the Census, 1890-1940: with outlines of actual tabulation programs, USGPO.

VAN DAM, W. 2007. Quantum computing: In the'death zone'? Nature Physics, 3, 220-221.

VEDRAL, V., BARENCO, A. & EKERT, A. 1996. Quantum networks for elementary arithmetic operations. Physical Review A, 54, 147.

VRANESIC, Z. & HAMACHER, V. 2009. Ternary Logic in Parallel Multipliers. Computer Journal, 52, 254.

WALLACE, D. 2002. Worlds in the Everett interpretation. Studies In History and Philosophy of Science Part B: Studies In History and Philosophy of Modern Physics, 33, 637-661.

WATROUS, J. 2008. Quantum computational complexity. Arxiv preprint arXiv:0804.3401.

ZHUANG, N. & WU, H. 1992. A new design of the CMOS full adder. Solid-State Circuits, IEEE Journal of, 27, 840-844.