UNIVERSITY OF SOUTHAMPTON

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING

Electronics and Computer Science

Progress Report: Simulation for Massively Parallel Processors

by

Lingjun Kong

A progress report submitted for continuation towards a PhD

Supervisor: Dr Tom J Kazmierski

Second Supervisor: Professor Steve R Gunn

30 May 2018

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING
Electronics and Computer Science

A progress report submitted for continuation towards a PhD

PROGRESS REPORT: SIMULATION FOR MASSIVELY PARALLEL PROCESSORS

by Lingjun Kong

It is a good age for developing a new architecture of computer[1]. Based on the old and developed single-core or multi-core architecture, with the huge computing demand of Artificial Intelligence and increasing computation workload from mobile device and other domains, with the foreseeable end of Moores Law, a more efficient and effective computer architecture is needed. In this report, a review of computer architecture will be included, a thought of how a new architecture based on massively parallel processor could be will be introduced, and a practice for achieving this aim: an incomplete RISC-V project with SystemC will be included.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem and Motivation

Recently years, the Artificial intelligence (AI) has been developed quickly especially since the AlphaGo [3] in 2016. Canonical processor has shown its weakness to produce this kind of massively matrix addition and production operation. GPU, FPGA and many other ASIC AI chip has shown their performance in this domain. It seems like the canonical processor will not able to be developed furthermore. Since 2004, the single core processor met the failed of Dennard-scaling and then turned from single-core to multi-core [4]. The increasing rate of processor performance has slowed down year by year[4]. The power-wall, memory-wall, and even the clock-synchronization has become the bottle-neck of processor performance. Canonical Processor are still occupied the most domain in computing, the slowdown of its performance has limited many domains of society. To combat these problem, we need pay more attention in memory access, energy saving techs and so on.

Massively Parallel Processors (MPPs) is a kind of techs to combine thousands of processors together to produce a huge computation problem. Many-core has shown its performance in energy management[5], such as big.LITTLE architecture of ARM[6]. SpiNNaker Project[7][8] has develop several different scales of event-driven MPPs for brain simulation based on Spiking Neural Network. This kind of architecture could also be used for another domain such as weather simulation and physical simulation. Compare with ASIC chips, this kind of architecture will be more universal. Compare with FPGA, this kind of architecture can be powerful with faster clock speed. Compare with GPU, this kind of architecture can be more energy efficiency. Above all, the aim of MPPs is using simple, cheap and self-organized cores to achieve a desktop-level supercomputer, fill up the black domain between general purpose processor and application specific chips.

1.1.1 The Trend of AI

The algorithm of AI, nowadays the most popular method is deep learning, which combine a massively of matrix computation as a network. Though the work of compression of AI algorithm is still continuing, the mainly computation of AI is still in the scope of matrix processing, or sometimes tensor processing. This kind of computation has taken a higher request for processors.

At the beginning of the application of AI algorithm, until 1st generation AlphaGo[3] in 2016, the computation of AI algorithm will always in CPU and other supercomputers. With the successfully of AlphaGo and the development of ImageNet Challenge[9], especially the help of NVIDAs CUDA, most of the work could be down more efficiently in GPU[10][11].Many AI chips were also developed for this computation, such like TPU of Google[12][13] and DaoDianNao chip[14]. FPGA also take a place of AI computation[11][15]. Xilinx and Altera also developed specific library for AI algorithm. Neuromorphic processor such like TrueNorth[16] using analogy circuit and SpiNNaker[8] based on digital circuit with off-the-shelf core. It is hard to say which one is the best, they all have their own advantages and disadvantages with different application and environment. But the challenges are still related with the challenge of the canonical processor: memory, energy performance and price.

1.1.2 The Ceiling of Multi-core

The limitation of Thermal Dissipation(TP) has lead to the dark silicon[17] problem[18]. The cores in a chip could not able work in the same time, and Esmaeilzadeh indicated this might be one reason of the end of multi-core if no driver occur[19]. Whether this statement, TP problem has actually limited the heap performance of one chip and reduce the performance-price ratio.

Another problem is latency, this latency mainly appears in processor-memory communication[4]. The gate switching speed is limited by voltage. And the communication speed is limited by light speed. Though the Moores Law is still working, the nanometre techs has actually slow down and wire delay scales is much slower than transistor performance[4].

With the development of nanometre techs, the die size has become smaller year by year. The size of a chip and number of cores will pay less in the price cost of one chip. Dark silicon limits the number working core in one time, but ARMs big.LITTLE architecture limits the working core themselves to saving energy. Eventually, the SpiNNaker chips, using event-driven mechanism, has much larger dark silicon than canonical processors. This shows that, dark silicon is a result of TP limitation, but could also be a method of saving energy. In this aspect, multi-core has much more bright future.

1.1.3 Programming Problem

With the increasing of number of cores, the thread management and task distribution might one complex problem. If all cores produce different thread, or with coarse granularity parallelism, the programming is not a problem. But such like matrix computation, it need fine granularity parallelism and need highly synchronization. Synchronization problem involves the memory, cache coherency, communication and many other question, which makes the system complex. SpiNNaker uses Global Asynchronization Local Synchronization (GALA) and with Spiking Neural Network is an excellent solution as a MPPs. But Not all application is suitable to use Spiking Neural Network, and how to organize the architecture of MPPs will affect the choice of programming model. For some problem, such as memory coherency, can be solved with either hardware or software, how to balance the work of hardware and software is another problem.

Parallel programming, or concurrent programming is still a complex work. However, if MPPs could use or transfer mathematical model directly to program, it might be much easy than single core programming.

1.2 Research Challenges

Using MPPs for science simulation, AI algorithm and other kind of application, make a more universal MPPs is a new and unpopular work. For me, my fundamental knowledge is not enough, and has less practice in progress design is the main problem.

For this research, firstly, design work is a huge challenge, how to finish the design using more efficient tools, language and with more computer assistant is one problem. Simulation is another problem, MPPs has many cores and the workload of simulation is heavy. How to evaluate the work and how to develop the application is a harder work.

1.3 Research Objective

With this research, one kind of MPPs architecture could be generated hopefully. But more important contribution is the method for design a MPPs. In this progress, inter-chip and intra-chip communication will be researched and a new and effective communication protocol might be designed. Simulation work will be another important contribution. With modern high-level synthesis language such like SystemC, many platforms could be used for the simulation of hardware. An idea of how to balance programming and hardware workload could be down and the conception of programming on MPPs could be more popular.

1.4 Report Structure

For this report, a literature will be put firstly, involves the basic knowledge of processors and parallelism. The simulation and computer-aided design method and tools will be review. Trend of hardware and architecture design will be included. GPU and SpiNNaker will be introduced as case study. Finally, an incomplete SystemC RISC-V model is the result of this period.

Chapter 2

Literature Review

In this Chapter, a review of the computer architecture will be given, to show my basic knowledge in this domain. Based on the book *Computer Architecture: A quantitative Approach 6th edition*[4] and *Computer Organization and Design: The Hardware/Software Interface - RISC-V Edition*[20]. The fundamental knowledge of computer architecture is prepared for the RISC-V practice. The section of parallel architecture, communication and network interconnection are prepared for the MPPs and NoC design. The above literature review will introduce the basic terminologies and more focused on the new trend. Finally, the simulation part is the literature preparation for the work of the first year.

2.1 Fundamental of Computer Architecture

In 2015, the International Technology Roadmap for Semiconductors (ITRS Report) had stopped at the February Workshop[21][22]. Nowadays, the Dennard scaling has actually failed[4], and Moore's Law may also come to its end in several years[23][24][25]. Meanwhile, since the Von Neumann architecture[26] been presented in 1945, there is no momentous change in the computer architecture. And until now, the development of architecture is still limited by the Semiconductor manufacturing processes.

2.1.1 Instruction Level Parallelism (ILP) Wall, Power Wall, and Memory Wall[2]

Single-core performance is limited by many factors, the ILP Wall and Power Wall are the two most important[4]. ILP techs has been developed much in last century. Instruction pipeline, super-scalar execution, out-of-order execution, register renaming, speculative execution and branch prediction all help significantly on the improvement of single-core performance. However, many limitations has been found while the people trying to continue increasing the ILP ability[27][28]. And due to the inefficient in silicon and power utilization while increasing the width of issue, the task of improve the performance has fall onto the increasing number of caches and cores per die[4].

Dynamic scheduling, speculation and multi issue are three most important ideas of ILP, but after the Intel's Itanium®, the research of speculation and multi-issue in ILP level has stopped[29] and researches about ILP may always combine it with Thread Level Parallelism together[30][31].

Memory Wall is also not a new term, after the first indication by Wulf in 1995[2], which is always the most serious problem in processor design[32].

2.1.2 Instruction Set Architecture (ISA)

There are many ISAs such as x86, MIPS, ARM, RISC-V et.al. ARM and x86 has occupied most of the market.

2.1.3 Multi-core

Asanovic listed 12 reasons about the turn from single-core to parallel computing in 2006[32], which could be concluded to four factors: power (power wall and static power), communication problem (increasing bandwidth-latency gap, memory wall), transistor level and manufacturing problem (soft and hard error rates and reliability, wire delay, noise, cross coupling of capacitive and inductive, clock jitter) , the increasing design complexity and the end of the research in ILP (ILP wall, manufacturing variability, design cost). The Berkeley Report called which is a 'brick wall'[32].

According to Hennessy[4], today's researches, based on multi-core, has met some other problems. Memory, or in terms of the communication of processor-memory and processor-processor. The processors which have many modules usually use network on chips, router and mesh topology rather than simple shared bus to help the communication. Message passing, and shared memory are two main method to exchange data packet. However, message passing request a high requirement to the router and shared memory has limited by the memory latency. The Amdahl's Law and the end of the Dennard Scaling also limit the development of multi-core.

Dark silicon or power wall, and memory wall are still the bottleneck of classical Von Neumann architecture. After a crazy increasing of the number of cores during 2014 with the boom of personal mobile device (PMD), eventually to 16 cores and more, the number of cores has come back to 8 or 4.

2.1.4 Heterogeneous and the future of computer

Start with the development of personal multimedia and the demand of image processing, GPU has start taking an important in computer as CPU. And with the demand of artificial intelligence, the will be more modules are under the need to be developed to support more function. Domain-specific and heterogenous are two important trends[33][4][18].

However, while we review the road of the computer, sometimes we can find out that there are many ideas could be found in the papers which published ten or twenty years ago. Such as the relationship between vector processor and GPU, such as the reawakening of In-Memory Computing (IMC) and Processor-in-memory (PIM). They were just limited by the nanometer technologies or materials before and can survive due to the improvement in semiconductor manufacturing processes.

In the history of computer, there are always two physical barriers has in the road of processor: silicon and light speed. The decreasing size of die has led to the popular of Internet of Things (IoT) device, the demand of the computation of IoT devices has led to the research trend of edge computing. The research of Big data could only base on the data collected from IoT devices and PMDs. Cloud computing is actually providing service for the computation of big data and mobile devices. Maybe we can say that, the improvement in computer architecture is not the contribution of computer architecture designer but the contribution of micro-electronics engineers.

2.1.5 Non-silicon and the effect of new materials

Mixed-signal IC is introducing analogue signal back to modern digital circuit. TrueNorth[16] is one of the successful achievement of analogues circuit. At the critical time for computer - high demand of artificial intelligence and the end of silicon manufactory - many new opportunities have occurred.

Carbon-nanotubes computer[34] or carbon-based circuit might be the future. Quantum computer might be another road. Biopolymer and the introduce of superconductivity[35] may also give a chance to new computer.

The techs above look like a dream, but some new materials may improve the situation in the near future. Three demission design of IC and the use of new material on memory - such as the Phase-changed RAM[36] and Resistive random-access memory (ReRAM)[37]. Which may lead to several trends in computer architecture: NoC from 2D to 3D, topology from mesh to cube, larger and quicker caches, improvement in energy efficiency. Memory management and data communication are the two factors may be affected.

2.1.6 Summary

Physical material controls the death or rebirth of the design of computer. However, we could do more with the existing materials. The trend now is from ILP and TLP to (Data Level Parallelism) DLP and Request Level Parallelism (RLP)[4]. Reconfigurable architecture and heterogeneous many-core are two import forms. Application specific or domain specific is the most important guide line while designing computer architecture. Energy efficient and performance are the only two target the designer need to catch.

It is no doubt that the physical level thing has limited the processor too much, either making architecture design more independent or helping them two more related could all help the design work reducing complex. However, it seems like the processor design is closer to software level design, FPGA is now helping the hardware and software design mixing together. The revival of neuromorphic chips is also making the boundary between software and hardware much vaguer. In the future, the design work in ILP, TLP, DLP and RLP may be intermingled, with the help of the computer-aided design and the development of artificial intelligence. Hopefully, the design progress of a domain specific chip or application specific circuit will be as simple as software design.

2.2 Parallel Architecture and Topology

The research in parallel architecture is usually in multi-core and multi-processor. Multi-processor has been the mainstream architecture at least after 2005, and the range of whose conception has also increased in these years. In Hennessy's *computer architecture*[38] written in 2011, the number of processors of a multi-processor is ”from a dual processor to dozens of processors”, and in the newest edition of the same book written in 2017[4], ”sometime hundreds of processors” has been added to the range in size. The focus of multiprocessor in this book also changed from 2 32 cores in 2011 to 4 256 cores in 2017.

For the parallel Architecture, the book *Parallel Computer Architecture: a hardware/software approach*[39] written by DE Culler in 1999 is one of the basic textbooks. Which introduces the parallel computing from performance and evaluation, shared memory multi-processor, snoopy protocol and directory protocol to scalable multi-processor, interconnect network and latency tolerance to hardware and software trade-off. To today's view, the only useful part of this book is the shared memory multiprocessor chapter. Which is a part of the multi-processor trend in the following ten years.

The most popular parallel architecture are multi-core and GPU, the former execution program in thread level and the latter execution program in instruction or data level. Meanwhile, FPGA could be another kind of parallel architecture which combined multiple Look-Up-Table (LUT) working together in parallel. If we extend the concept of parallel, ASIC or all digital circuit could be a parallel computing architecture which making all transistors working in parallel. According to Hennessy[4], there are several levels of parallelism: ILP, TLP, DLP and RLP. And which could be two basic classification in application: data level parallelism(DLP) and task level parallelism (TLP). And Flynn, according to the parallelism in instruction and data, made four classes for computer architecture: SISD, SIMD, MISD, MIMD[40]. And then, more different kinds of parallel computer appeared: Parallel vector Processor (PVP), symmetric multiprocessing (SMP), massively parallel processing (MPP), Cluster of Workstations (COW), Non-Uniform Memory Access (NUMA), Cache Coherent Non-Uniform Memory Access (CC-NUMA), Distribute Shared Memory (DSM). All these conceptions were mentioned at last century and some of them went death, and some of them rebirthed again.

For this section, the basic concept in parallel computation will be reviewed, the challenges and the trend of parallel computing will be discussed.

2.2.1 Definition and organization of parallel architecture

Levels of Parallelism

According to Hennessy[4], and some other literatures, there will be several different levels of parallelism:

Bit Level: Implicit level, refers to how many bits could be processed in the traditional computer, such as 8, 16, 32, 64bits. Usually indicates the bus width.

Instruction Level: Implicit level, is the mainstream aspect while the researchers focused on the single-core performance. Aimed to increasing the number of instruction to be processed per clock cycle or per second.

Data Level: Explicit level, aimed to process more data in one time. GPU, Vector Processor and Instruction Set SIMD Extension (such as AVX for X86 and NEON for ARM) are three important expression to this level of parallelism. And the research has start transforming from hardware to software from this level.

Thread Level: Explicit Level, for improving the utilization of single-core, by adding more program counter. For these years, TLP often appears with multi-core and multiprocessor,

Task Level: Explicit Level, sometimes is the alternative name of Thread Level. Which is composed by function or task in application view.

Request Level: Explicit Level. And aims to response the requests from customers with several decoupled tasks.

Depth/Gain of Parallelism:

Totally three kinds of grain: fine, coarse and embarrassing. It indicates the communication frequency between different thread. In the other hand, it also refers to in which level of parallelism the most important communications happen.

Types of Parallel architecture

According to the number of processors or cores, the parallelism could happen on uniprocessor, multi-core, and multi-processors. The parallelism on uniprocessor are scalar processor, vector processor and the processors using SIMD instruction set. Then have just one processor but could issue multi-instructions and execute instruction level parallelism. The multi-core system has multiple cores or execution unit. They could execute parallel tasks in data level (GPU) or thread level (multi-core) on one chip. Multi-processor system includes cluster and massively parallel processors. For this part, we will focus on the multi-processor and multi-core architecture.

According to the programming model, the parallel architecture could be shared address or message passing. Which could also be called the communication models, communication mechanisms or data exchange forms between parallel tasks. There are many researches have compared these two kinds of architectures[4][41][42]. And in fact, there are many terms or subclass for parallel programming model, such as multiprogramming, shared address, message passing and data parallel. In another view, which could also be divided to shared address space architecture or distributed address space for all large-scale multiprocessors[4]. See Figure 1. Thread model is one important kind of shared memory programming, but for programmer, they will more focus on the thread level parallelism. For example, the POSIX Thread, OpenMP and the thread developed and used by Microsoft, Java, Python and CUDA. What's more, the Data Parallel Model will also be regarded as one important class of programming model, which may also be referred to as the Partitioned Global Address Space (PGAS) model.

Figure 1 The space of large-scale multiprocessors and the relation of different classes[4]

Shared Address Space/ Shared Memory Architecture

The foundation to classical Shared Memory Multi-Processor (so called shared address space) is the organization of memories and interconnect strategy[4], in terms of how the modules in computer (mainly indicates the cores) exchange data and messages with each other. According to the organization method of memory, Shared Memory Multi-Processor has two basic class: Symmetric Multi-Processors (SMP) or Uniform Memory Access (UMA) using one centralized memory and Distributed Shared Memory (DSM) or Non-Uniform Memory Access using distributed. The former one is usually used in small scale multi-processor system, but the mainstream number of cores has increased from 8 in 2011 to 32 in 2017[38][4]. The later one could help increasing the ceiling of bandwidth and latency with a more complex software support to increasing the utilization of existing bandwidth. It could achieve better performance in large scale multi-processor systems, and in these years, more and more small scale multi-processors have start to use this architecture and most of the multi-chip systems are using this architecture[4]. All the shared memory processors using shard memory communication rather than message passing, and one of the limitation of shared memory multi-processors is the cost of communication[4]. As another limitation, the lack of parallelism of traditional program has limited the speed up according to Amdahl's Law.

Distributed Address Space / Message Passing Architecture

For distributed address space, which means that there will be many processors with private memory space. In most of the system using distributed address space, message passing is used as the communication model. For the other terms, such as Massively Parallel Processors and cluster, the meaning of them still doesn't have an exact range[43], but we could use large-scale multiprocessor to conclude them[4].

The target for this project is focusing on Massively Parallel Processors (MPPs). The MPP is one kind of large scale multiprocessor architectures, built by multiple small processors and always used for scientific applications.[4] The term "massive" will also changes with the time. General, the Illiac IV in 1970s is regarded as the first MPPs with a SIMD array consisted of 64 processing units and connected by a 2D torus with each other[4]. And then is the cosmic cube[44] with 64 processors in 1985. The connection machine[45][46] using 64K 1bit processors is one successful implementation of MPPs in 1980s. The idea of using on bit processor to build a thinking machine still have an important meaning for today's AI chips. The research on MPPs then turn to high performance computing with low dimensional topologies in 1990s, such as the "Intel Paragon, Cray T3D, Cray T3E, HP AlphaServer, Intel ASCI Red, and IBM Blue Gene/L"[4]. However, due to the cost of MPPs, the supercomputer market had been occupied by cluster[4][47].

Figure 2 Architecture of Supercomputers Nov 2017[47]

The though the massively parallel processors cost more than cluster, the performance of which is better. For example, the Sunway TaihuLight using MPP array and RISC instruction set. Meanwhile, the SpiNNaker[7][8] using massively parallel processor also provided the idea: using MPP as a neuromorphic system to run a neuro network model for massive parallel processing.

2.2.2 Synchronization problem

For parallel processing, the most important is the communication between threads, or in terms of the execution units. The communication models for massively parallel processors are message passing and shard memory, the former one will be introduced in 2.3, and the later one will be introduced in 2.4.

For the system using global clock, the synchronization of clock will be one serious issue, especially for the systems like MPPs who use large scale board. Meanwhile, with the increasing area of on chip or the increasing size of one system, the clock problem in communication, including clock skew and jitter, wire latency, should all be considered. Synchronization or asynchronization are two ways to achieve one network. Some other design also uses global asynchronization local synchronization (GALS) design[7][48].

For the practice of parallel systems, data dependencies will affect the synchronization design. The Bernstein's Condition raises the requirement for synchronization. The method for synchronization could be hardware and software. For Hardware Support, Test-and-Set is one primitive method, Load Link/Store Condition will always be used for MIPS, ARM, PowerPC and Alpha, and Compare-and-Swap (CAS), Test-and-Set, Atomic Increment, compare-and-exchange and Bus Lock Prefix will usually be supported by x86. Software synchronization like POSIX Thread support locks, barriers and condition variables, and also support a high-level method like the monitors, parallel sections and loops.

Atomic operation will always be used for concurrent processes. For example, the test-and-set for the synchronization achieving while accessing critical on uniprocessor.

2.2.3 Performance overhead and speedup

Amdahl's law and Gustafson's law have indicated one serious problem, that is the speedup of a parallel system is limited by the proportion of parallelism part of the program. Amdahl's Law shows that there will be a diminishing return in the performance with the increasing number of cores, if the parallelism of program has no change[49].

For the meantime, the Operational Intensity, which indicates the transferred operations rate between execution unit and DRAM, has affect the performance of multi-core system[50]. The transferred rate is limited by processor-memory communication speed. That is the ubiquitous memory wall.

However, manycore as FPGA, are using cheap, simple and smaller cores to provide the chance. SpiNNaker has provided an idea, that is developing one system on neuro level parallelism and making neural network as the programming model. The neural network itself could provide massively parallel chance and the performance has already be confirmed by recent year's practice. Meanwhile, using the architecture of traditional massively parallel system and manycore system, to develop a self-organized, reconfigurable network to support this kind of neural network will not be an unreachable goal. The challenges are the system and software environment.

2.3 Communication, Network on Chip (NoC) and System on Chip (SoC)

As the communication architecture, shared memory communication and message passing are two basic idea. Using shared memory with global variables for exchanging data will always use in small scale machine. Message passing communications use router or other message controllers to forward data between terminals just like a small Internet.

For the shared memory communication, the bottleneck is now focusing on the access speed to memory and the memory consistency. For the message passing communication, though the memory access bottleneck should be considered, the topologies, routing method and algorithms have become the important parts in this kind of traffic and finally were grouped together as the Network on Chip technologies. The shared memory part has been down in 2.4, and for this section, the details of interconnect network will be discussed.

Many books have already given the basic knowledges about NOC during the first decade years[4]:

Networks on Chip[51] published in 2003 by Axel Jantsch collected 14 chapters written by different experts in three parts: system design, hardware design and software interface.

Interconnection networks: an engineering approach[52] in 2003 is the most influential textbook for NoC. It introduced the NoC in engineering view.

Principles and practices of interconnection networks[53] in2004 is regard as one of the most important classical textbooks for learning NoC. It introduces the Interconnection network (System Area) with three major parts: topology, routing and analysis and is more focus on the topology.

As we can see, the theory of interconnection network on chip had already become quite mature in the last 10 years. For example, the deadlock problem was introduced by Holt in 1972[54], and the avoiding techs were developed during 1980s[16][55][56]. The switching technologies were also developed during 1980s and 1990s, for example the pipelined switching with virtual cut-through[57] in 1979 and the wormhole switching[58] in 1986. And then, such as the virtual channel and escape channel were all developed and help avoiding the deadlock.

According to Pinkston and Duato[4], the interconnection network could be grouped to four levels based on their size:

On-Chip Network (OCN) or Network on Chip (NoC) refers to the network used in one chip to connect the functional units of processors.

System Area Network (SAN) and Storage Area Network aims to solve the processor to processor connection and processor to memory connection problem. For example, the network for a massively parallel processor system.

Local Area Network (LAN) refers to the network to connect computers in room wide. Such as the Bluetooth LAN and Wi-Fi LAN.

Wide Area Network (WAN) refers to the long-haul network, for example, Asynchronous Transfer Mode and Internet.

In this section, the OCN and SAN who are useful in Massively Parallel Processors will be discussed.

In one network, the node, interface, switch/router and link/channel are four basic components, messages (packet and flit) will be transferred through the whole network and finally arrive the destination. During the design of this kind of network, topology, routing algorithm and flow control are three issues should be decided. The scalability, QoS, performance (throughput, latency) and energy efficiency will all be affected by these decisions.

2.3.1 Topologies and architecture

There are many kinds of topologies for NOC, started from the simplest shard bus (single line) and rings, meshes crossbar and finally to the hypercube in system area, researcher have almost tried all the forms could be thought and finally come back to simple topology since 1990[1][4].

The basic topologies have already been introduced by many resources, bus, crossbar, ring, mesh, and torus are still the most popular designs in modern processors. Topology is the foundation of the NoC design. For one specific topology, there are several properties to evaluate if which is suitable for one design: direct/indirect, blocking/non-blocking, the cost, latency, throughput (bisection bandwidth) can be calculate according to topology.

As mentioned before, the clock synchronization problem, whether use global clock or not, how to implement GALS should be considered while designing the topology.

2.3.2 Router, routing algorithms and flow control

Routing method decides the route of one message, and flow control decides the how the storage and buffer working in this route. The routing can be Oblivious (deterministic/static or random) or adaptive (dynamic, for avoiding deadlock and livelock)[53].

With different topology design, the routing algorithm is also different. The table describe several main routing algorithms.

Table 1 The classic routing algorithms Routing method Description and bibliographic Deterministic Randomized routing Be described by Valiant in 1981 firstly[59].

minimal oblivious routing algorithm According to Nesson and Johnsson, 1995[60]

Load-Balanced Oblivious Routing Introduced by Singh in 2002[61]

Adaptive wormhole routing strategy Linder's wormhole routing strategy with virtual channel in 1991[62]. A survey for wormhole routing in 1993[63]. For mesh and without virtual channel in 2000[64].

Adaptive routing with virtual channel Based on block faults and using virtual channel in 1992 by Chien.[65] Dally made the free virtual channel's number as the congestion metric in 1993.[66][67]

Minimal adaptive routing Used on fat tree in CM-5 by Leiserson in 1992[46]

Fully Adaptive Routing Load-Balanced Adaptive Routing Search-Based Routing

In 1988, Dally indicated that removing cyclic dependencies on channel dependency graph is one necessary and sufficient condition for designing a deadlock-free routing algorithm[56][68]. the turn model then made this theory be basic to adaptive and deterministic routing and free the restriction on wormhole networks.[68][69]

Meanwhile, Duato also showed a necessary and sufficient condition to free the restriction while using channels for designing a deadlock free fully adaptive routing.[70][68]

The flow control is not only just storing data to buffer, but also usually be considered together with adaptive routing. The topics of adaptive routing are usually how to avoid the deadlock and livelock, how to improve the fault-tolerance performance, how to work on a reconfigurable network, how to keep the load balance. There are several flow control methods, such as the circuit switching, packet-based store and forward, packet-based virtual cut through and flit-based wormhole flow control.

To analyse the performance of one routing method, worst-case analysis method[71] could be used for oblivious routing.

2.3.3 Communication and protocols

For architecture level communication model, the message passing model and shared memory communication are two classes. This part will focus on the protocol and interface in NoC view and discuss the one-to-one, one-to-many and one-to-all communication mechanism.

In massively parallel processors, whether it is using direct or indirect network, the communication can be concluded to processor-processor, processor-memory, processor-switch and switch-switch communication. For message passing model, the communication will focus on the processor-processor communication or in terms of processor-router communication and router-router communication. For shared memory system, it is processor-memory communication.

There are several properties for communication protocols: synchronization or asynchronization, how a clock aids the communication, how many wires the channel needed. Besides the simplest instruction-like control signal, there are still many protocols could help achieving a more complex network and communication task.

Moreover, which could also be classed by the type of interface: system control interface (i.e. clock gating), standard bus interface (i.e. AMBA, OCP, SRAM, MIPI), non-standard but interface, test interface (i.e. DFT) and other miscellaneous control interface.

The most common practice is Inter-Processor Communication (IPC), which contents a set of methods such as I2C, SPI, and SDIO which need to connect to a slave or a client controller. UART is the simplest interface for embedded processors. These protocols are also suitable in other communications.

For on-chip processor-memory communication, the DMA protocol also plays an important role.

2.3.4 QoS and reliability

There are many aspects on software level and algorithm level needed to be considered while designing a NoC system. The research on topology, flow control and routing are focusing on improving the efficiency of a network in more hardware level. However, while allocation the communication tasks, there will also be a fair problem. Moreover, the arbitration focuses on dealing multi-request for limited resources.

Error Control and Fault tolerance are important to keep resilience, reliability and availability.

2.3.5 Furthermore, and conclusion

There are many technologies which may change the form of NoC, such as the Optical NOC[72], wireless communication[73] and 3D Stacking.

3D stacking could help in increasing bandwidth and reducing the latency and power[32]. And many researches have pay attentions to this field[74]. But this tech has been indicated over at least 20 years, and still not be able to achieve onto custom device. It is better to ignore its influence in at least five years.

2.4 Memory Hierarchy and Cache Coherency

The performance of modern processors is limited by the communication speed, and the memory access speed is one of the important bottlenecks in the process of communication. For the best and simplest way to achieve a high-performance processor, it is better to read/write data from one large memory with high bandwidth and low latency. However, it is just a dream now due to the area limitation and material limitation and cache has become an alternative way for improving memory performance.

2.4.1 Memory Hierarchy

Cache is the most important part in the memory systems for improving the performance of a processor[75].

L1 Cache (First Level Cache)

For the most processors, there will be an Instruction Cache (I-Cache) and a Data Cache (D-Cache) as the level one cache. These two caches separate data and instruction are organized as a Harvard Architecture and aims to increasing the access speed. The L1 Cache is mad by SDRAM which using 6 transistors for one bit[4]. So, the L1 cache could not be too larger, otherwise will occupied too much of area and may influence the latency. And the L1 cache also limited by the clock cycle[4].

L2 Cache (Middle Level Cache)

L2 Cache aims to improve the hit rate with a larger SRAM than L1 Cache.

With the L1 cache and L2 cache, the hit rate could be keep at least above 95% in modern processors[76].

L3 Cache (Last Level Cache)

The L3 Cache appears with the multi-cores, as a shared memory for all cores to improve both the hit rate and access time.

L4 Cache (eDRAM)

Sometimes there will be a L4 Cache as an eDRAM for the communication between CPU and GPU.

RAM

Usually using SDRAM or DDR RAM as the main algorithm memory.

ROM

Using EEPROM (such as Flash) or Hard Drive as a storage memory.

2.4.2 Memory Materials

For nowadays, SRAM is the fast and most expensive memory. Which is built by 6 transistors per bit to achieve high performance. And DRAM is a middle performance memory with only single transistor per bit to get a balance between cost and performance. The NOR and NAND Flash memory has recently become the mainly memory material as an alternative of ROM and DRAM. However, all the memory techs are still not good enough for the modern processor.

Area is one part: the L1, 2, 3 Cache has occupied half the area in most processors, with high power consumption. Access speed is another part: even using the cache and achieving a high hit rate, it still need several clock cycles to get data from memory. What's more, the access time of main RAM and ROM also need to be improved, especially for the increasing size of data due to the development of multimedia and machine learning.

For recently years, there are several memories may affect the future processors.

NRAM[77]

NRAM is one kind of Non-Volatile Memory (NVM) based in Carbon Nano Tubes(CNT). It may able to be the alternative of EEPROM and NOR Flash.

ReRAM

ReRAM is a kind of NVM aimed to replace Flash RAM. Which also re-leads the trend of In-memory Processing and ReRAM Computing in neural network domain[78].

PCM (Phase Change Memory)

Is indicated as one important techs to replace the DRAM as the main memory[79].

2.4.3 Memory Consistency and Cache Coherency

The fundamental knowledge has already explained explicitly by Sorin in book *A Primer on Memory Consistency and Cache Coherence*[80].

Cache Coherence protocol

The Cache Coherence aimed to solve the incorrect problem while reading an address and make sure the value read from cache line is the same with the present, newest one.

In terms of spatial coherence. The main work were down in last century, such like the early work in hardware cache coherence[75][81][82] and the early survey about cache coherence protocol[83] in 1986.

The term Cache Coherence comes from Distributed System, and now has two main kinds of implementations or mechanisms in shared memory systems: Snoopy Coherence Protocol and Directory Coherence Protocol. To add states (Modified, Shared, Invalid, Exclusive, Owner) to these two protocols, VI(valid/invalid), MSI (Modified, Shared, Invalid), MESI (Illinois protocol: Modified, Exclusive, Shared, Invalid), MOSI (Modified, Owner, Shared, Invalid), MOESI (Modified, Owner, Exclusive, Shared, Invalid), MESIF (Modified, Exclusive, Shared, Invalid, Forward), MERSI (Modified, Exclusive, Recent, Shared, Invalid). And for different level of cache, protocol could be different. For example, the ARM implements MOESI to L1 Cache and MESI to L2 Cache[84].

In the traditional snooping protocol, all the cache controllers must monitor all the activities from others to maintain the state in coherence, need a bus or ring as the medium for broadcast the caches activities. Usually used with buses in small scale Chip Multi-Processor systems (such as centralized shared memory architecture[4]) and require a totally-ordered interconnect[85][80]. The snooping protocol has a bandwidth overhead due to this kind of requirement of fully-interconnect[86], and directory protocol could help this kind of ceiling by using a central control device to handle all cache activities (provide the global state or summary state) and could be implemented to distributed memory systems in shared memory multi-processors[4]. By shifting the task for providing the ordering from bus to directory, the pressure on traffic could get a relaxation.

What's more, in practice, the intra-core/processor's cache coherence has a little difference with it under distributed model. Table 2 List of Key Literatures about Cache Coherence Year Title Contribution 2003[85] Token Coherence: Decoupling Performance and Correctness proposed a new coherence framework, could separate performance from correctness. Compare with directory protocol, it has low latency and unordered interconnect, compare with snooping protocol, it avoids indirection. Is one important alternative protocol. 2008[87] Virtual Tree Coherence: Leveraging Regions and In-Network Multicast Trees for Scalable Cache Coherence Using VCT to solve the scalable coherence problem 2008[88] Verification of chip multiprocessor memory systems using a relaxed scoreboard Using memory scoreboard to aid simulation-based validation is accurate but complex while implementation sue to its memory dependence and specific implementation dependence. This paper decoupling the specific design with a relaxed scoreboard. 2010[86] A Tagless Coherence Directory Using Tagless(TL) as a scalable coherence solution to avoid the area overhead of Directory Protocol in large CMPs. 2010[89] Cohesion: a hybrid memory model for accelerators This work helps temporal data reassignment and dynamically register memory regions using both hardware and software coherence management. 2011[72] Atomic coherence: Leveraging nanophotonics to build race-free cache coherence protocols Using optical mutexes and adding support for speculative coherence. Proposed a new breed of race-free coherence protocol basing on broadcast. 2012[90] SCD: A scalable coherence directory with flexible sharer set encoding Using highly-associative cache to reduce the cost and improve the scalability.

Memory Consistency Model

The details of memory consistency problem can be found in Hennessy's book[4] and Adve's tutorial[91]. The key factor of Memory Consistency Model is the order of Read-/Write. There are many Consistency Models but only there are mainstream in commerce: Sequential Consistency (SC), Total Store Ordering (TSO) Consistency for X86 Architecture[92] and Relaxed Memory Consistency (RMC) for ARM and Power PC[93].

The SC model need both the execution of load and store instruction following the order in program strictly. This kind of consistency could be achieved by using whether a token ring or locking a cache line[94]. However, the SC model has limited the optimization in hardware and many compulers[91][95].

Total Store Ordering Consistency (TSOC) just limit the consistency between memory order and program order of the store instruction. With only a FIFO store buffer based on the SC model could achieve this consistency simply. The store could help increasing the performance of a single load operation and improve the throughput of system.

However, above two models have made the system a little more complex. The Relaxed (Weak) Memory Consistency allow all kind of reorder unless the target addresses of two instruction are the same one. The IBM Power pc and ARM are the example of RMC. It can be achieved by replacing the store buffer of TSO to a buffer who can support reorder load/store buffer.

In distributed system, the coherence is achieved by transaction with four features: ACID (Atomicity, Consistency, Isolation and Durability).

Below are several representative papers about memory consistency. Year Title Discussion 1995[96] Is sc+ ilp= rc? Achieving the SC model the same performance as release consistency by providing hardware support. 2007[97] BulkSC: bulk enforcement of sequential consistency It partitions the execution of program into atomicity, isolation and durability transactions and uses speculation to improve performance. But rely on expensive speculation hardware and cost many to buffer and prevent the speculative stores[97].

2009[98] InvisiFence: performance-transparent memory ordering in conventional multiprocessors 2011[99] Rethinking the memory hierarchy for disciplined parallelism. In Parallel Architectures and Compilation Techniques (PACT), 2012[100] RADISH: always-on sound and complete Ra D etection in S oftware and H ardware 2012[101] End-To-End Sequential Consistency Let SC model cost less than TSO by adding an unordered store buffer.

2.4.4 Communication for cache coherence

Due to the complex States and Rules for the cache coherence, there will be numerous communication operations for the transmitting message (such as to make sure there are how many copies of one cache line, how to pass a value to cache directly). The cache coherence technology has become more complex: QPI (QuickPath Interconnect) of Intel provides a point to point communication channel between cache clusters and every cache cluster has its own Integrated Memory Controller (IMC) for cache-memory communication[102]. The ARM's CoreLink[103] has several aspect related with cache coherence, for example the CoreLink Generic Interrupt Controller (GIC)[104] who makes sure the synchronization interrupt between cores, the CoreLink Cache Coherent Network (CCN)[105] who takes charge of the cache interconnection and cache coherence, and the CoreLink Dynamic Memory Controller (DMC) [106] who is responsible for the communication between cache and memory.

2.5 Simulation

With the increasing size of the chip, more function and transistors have been built to a computing system. Which has affect the emulation and simulation work in both time and resource cost. According to the report from Wilson in 2014 [107], the verification work has cost the most of project time.

Meanwhile, there are many verification method and ideas appearing. Simulation Verification, Formal Verification, Hardware Assisted Verification, Emulation and Virtual Prototyping are all used to solve different verification problem. What's more, the System Verilog has met a significate improvement and the merger with Universal Verification Methodology in both Language and Methodology (including the System Verilog Assertion and Function Coverage). More and more tools are playing their roles in verification. The trend of moving the abstraction level from register transfer level RTL to Transaction-level modelling TLM could also help the verification work.

However, there are still many challenges in the pre-silicon verification. Firstly, the increasing number of IPs and the increasing complex of the systems have put forward a serious question for verification: how to make the verification work more efficiently and powerful. Secondly, the trend of software-hardware codesign has also raised one problem, that is how to reduce the gap between pre-silicon design and verification and post-silicon test.

The virtual prototype or in terms of Transaction Level Model (TLM) could help the development start begin from a more abstract level and could also help the hardware design and software design starting in parallel. Start from software-only simulation to hardware-accelerate emulation and FPGA accelerator, the virtual prototype plays important roles in verification work.

The TLM 2.0 standard has been regarded as the common abstract level model. SystemC and UVM are two important tools support this standard. The simulation work will also base on TLM by using simulators or virtual prototyping platforms.

There are many tools could support the simulation. From MATLAB in algorithm level to ModelSim in HDL and TLM level. Also, the GEM5 support the simulation could be down in a simpler way[108]. Meanwhile, there are also many works focusing on the speedup of simulation. For example, the ASIM[19] by enable the modularity and reusability, the Graphite[109] project and Wisconsin Wind Tunnel II[110] who accelerate the simulation on multicore or using a parallel architecture.

Recent years, with the development of GPGPU and the CUDA tools of NVIDIA, the simulation on GPU has also show a new way besides the simulation on CPU[111][112][113][114][115][116].

2.6 Summary and Plan

This chapter is the review of the basic of doing a parallel system. And also shows the trend of IC design. For this time, with the slowdown of the improvement in nanometer fabrication and the perhaps failed of Moore's Law, both the architecture and design ideas will be affect significantly. Meanwhile, both the carbon-based chip and quantum computer may also affect the normal silicon chip. However, in another view, with the development of the personal multimedia devices and many other application, such as edge computing, IoT and cloud computing, the demand of computing resource also be unprecedented huge. The artificial intelligence researchers are aiming to meet the demand in algorithm and software level. Many hardware designers are also join the trend by developing AI chips. The normal processors could also change to another thought, that is using neural network as much as possible to achieving a more powerful and universal manycore system. Using simple and low-cost processors building a massively parallel processor system and implementing neuro parallelism.

Based on this thought, the RISC-V could be the simple processor, and system-c and GPU will be used as the simulation platform. The SpiNNaker will be studied as an important case study who achieving a success Spiking Neural Network on Massively Parallel Processors. Then I will start a practice by making one simple processor on RV32-I in order to understand the knowledge I have reviewed above.

What's more, there are still many factors not covered in this chapter, for example the power management and low-cost design, the programming and software for parallel system. These sections will be studied in next month.

Chapter 3

Case study

3.1 RISC-V

Before the RISC-V, there are many RISC processor architecture like OpenRISC, SPARC and ARM. However, the OpenRISC is just an open source core with GPL protocol, which let the architecture is very difficult to be developed completely. The SPARC architecture need a huge number of registers and is hard to be used to personal computer and mobile devices. ARM focus on low-power and mobile devices help its development. But the expensive IP fees make the tradespeople start to find another alternative architecture and finally is the RISC-V Foundation in 2016.

The RISC-V instruction set is much simpler than ARM, due to its young and doesnt have to be compatible to some old design. The modularity design also help the RISC-V could achieving a better adoptive ability than ARM.

3.2 SpiNNaker Project

There are many highlights in the SpiNNaker Project. The first one is, during this project, a spiking neural network was successfully implemented on a massively parallel processor architecture. Which means there is possible to employ other kind of neural network to MPPs. Eventually, this kind of MPP architecture could employ many kinds of neural network. The second one is the topology, which use a hexagonal architecture, which could help one neuro in the network can communication two neighbourhood in the next layer and one neighbourhood cross the next layer.

Figure 3 The topology of SpiNNaker Network

Chapter 4

Practice: SystemC with RISC-V

The purpose of this practice is make myself to be familiar with SystemC and RISC-V architecture. To understand all the progress from IC design to simulation.

During this period, a counter, a register, and a program memory have been written down. And for now, I am trying to generate the execution code using RISC-V tools in Ubuntu. So, the simulation of PM still not available.

4.1 Generate the execution code and disassemble code

4.2 The Simulation results

4.2.1 Counter

The counter has 4 control signals: Wait, increasing 1, directly jump and related jump. Figure 4 is the simulation result.

Figure 4 Simulation Result of Counter

Chapter 5

Plan & Future Work

5.1 Introduction and Objective

The multi-core bottleneck and the dimension returns of Moore's Law have shown the need of a more efficient architecture for computing. The increasing demand of computing resources from high performance computer to mobile devices to IoT devices is also pushing the developing of computing machines. Many core system or in terms of massively parallel processors has shown their potential in the future. Meanwhile, the increasing demand on computing has also raised a higher requirement for IC design house. However, the work load on verification stage, pre-silicon period, is no limiting the productivity. By the way, the RISC-V will also shows its vitality in almost all the area of computer.

For this project, the main target is extending the application area for Massively Parallel Platforms (The POETS of our school), focusing on using a MPPs platform as the simulation accelerator for IC design, and especially for the simulation of RISC-V. A further more target is trying to do some contribution for the development of MPPs. For the most recent target, a single core RISC-V model will be built using a high level abstraction language (SystemC) as a practice and a pre-work. Below is the details for the plan in about one year.

5.2 Single Core RISC-V Stage

A single core RISC-V SystemC model will be built as soon as possible. The target is let the model can running several RV32-I programs and further more is the Compatibility Test. Then is trying more simulation method for this model, first one is using CUDA and GPU for simulation, the second one is using FPGA as an accelerator. Meanwhile, this model will be extended to more levels: RTL, Gate and Netlist level in order to improving myself and finding out the faults in the model. Finally, the simulation will be execution in the MPPs and a primary simulator for RISC-V on MPPs will be developed.

5.3 Multi-core RISC-V Simulation Stage

The communication network is the most important part in a MPPs platform or many-core system. While the designing of Multi-core RISC-V system, a network model with a communication protocol will be built.Then is the model of multi-core RISC-V. A simulator for this model on MPPs will also be developed. Finally, a RISC-V based event-driven massively parallel processor system is hopefully generated.

5.4 Resources Needed

A GPU platform is needed for the simulation. A FPGA board is needed for the evaluation. The computing support from the POETS platform will also be needed. Moreover, I wish the library could purchase several books for the research:

1. Computer Organization and Design (RISC-V Edition) The Hardware Software Interface We have the 5th edition (2014) in our library but which is always on loan. For this book is a newer version and keeps the state of art in computer architecture: RISC-V. Which will be a very useful book.

References

[1] William J. Dally. Performance analysis of k-ary n-cube interconnection networks. *IEEE transactions on Computers*, 39(6):775–785, 1990.

[2] Wm A Wulf and Sally A McKee. Hitting the memory wall: implications of the obvious. *ACM SIGARCH computer architecture news*, 23(1):20–24, 1995.

[3] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[4] John L Hennessy and David A Patterson. *Computer architecture: a quantitative approach*. Elsevier, 6 edition, 2017.

[5] Shekhar Borkar. Thousand core chips: a technology perspective. In *Proceedings of the 44th annual Design Automation Conference*, pages 746–749. ACM, 2007.

[6] Edson Luiz Padoin, Laércio Lima Pilla, Márcio Castro, Francieli Z Boito, Philippe Olivier Alexandre Navaux, and Jean-François Méhaut. Performance/energy tradeoff in scientific computing: the case of arm big. little and intel sandy bridge. *IET Computers & Digital Techniques*, 9(1):27–35, 2014.

[7] Steve B Furber, David R Lester, Luis A Plana, Jim D Garside, Eustace Painkras, Steve Temple, and Andrew D Brown. Overview of the spinnaker system architecture. *IEEE Transactions on Computers*, 62(12):2454–2467, 2013.

[8] Steve B Furber, Francesco Galluppi, Steve Temple, and Luis A Plana. The spinnaker project. *Proceedings of the IEEE*, 102(5):652–665, 2014.

[9] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. Large scale distributed deep networks. In *Advances in neural information processing systems*, pages 1223–1231, 2012.

[10] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[11] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[12] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pages 1–12. ACM, 2017.

[13] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[14] Yunji Chen, Tao Luo, Shaoli Liu, Shijin Zhang, Liqiang He, Jia Wang, Ling Li, Tianshi Chen, Zhiwei Xu, Ninghui Sun, et al. Dadiannao: A machine-learning supercomputer. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 609–622. IEEE Computer Society, 2014.

[15] Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong. Optimizing fpga-based accelerator design for deep convolutional neural networks. In *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 161–170. ACM, 2015.

[16] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

[17] Hadi Esmaeilzadeh, Emily Blem, Renee St Amant, Karthikeyan Sankaralingam, and Doug Burger. Dark silicon and the end of multicore scaling. In *Computer Architecture (ISCA), 2011 38th Annual International Symposium on*, pages 365–376. IEEE, 2011.

[18] Michael B Taylor. Is dark silicon useful? harnessing the four horsemen of the coming dark silicon apocalypse. In *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, pages 1131–1136. IEEE, 2012.

[19] Joel Emer, Pritpal Ahuja, Eric Borch, Artur Klauser, Chi-Keung Luk, Srilatha Manne, Shubhendu S Mukherjee, Harish Patil, Steven Wallace, Nathan Binkert, et al. Asim: A performance model framework. *Computer*, 35(2):68–76, 2002.

[20] David A Patterson and John L Hennessy. *Computer Organization and Design RISC-V Edition: The Hardware/Software Interface.* Newnes, 2017.

[21] Luis Ceze, Mark D Hill, and Thomas F Wenisch. Arch2030: A vision of computer architecture research over the next 15 years. *arXiv preprint arXiv:1612.03182*, 2016.

[22] ITRS. International technology roadmap for semiconductors 2.0. http://www.itrs2.net/itrs-reports.html, 2015. Accessed 2 May 2018.

[23] Andrew A Chien and Vijay Karamcheti. Moore's law: The first ending and a new beginning. *Computer*, 46(12):48–53, 2013.

[24] Rachel Courtland. Transistors could stop shrinking in 2021. *IEEE Spectrum*, 53(9):9–11, 2016.

[25] National Research Council et al. *The Future of Computing Performance: Game Over or Next Level?* National Academies Press, 2011.

[26] John Von Neumann. First draft of a report on the edvac. *IEEE Annals of the History of Computing*, 15(4):27–75, 1993.

[27] Matthew A Postiff, David A Greene, Gary S Tyson, and Trevor N Mudge. The limits of instruction level parallelism in spec95 applications. *SIGARCH Computer Architecture News*, 27(1):31–34, 1999.

[28] David W. Wall. Limits of instruction-level parallelism. In *International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS*, volume 26, pages 176–188, Santa Clara, CA, USA, 1991. Publ by ACM, New York.

[29] Sebastian Winkel. Exploring the performance potential of itanium® processors with ilp-based scheduling. In *Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization*, page 189. IEEE Computer Society, 2004.

[30] Cheng Wang, Youfeng Wu, Edson Borin, Shiliang Hu, Wei Liu, Dave Sager, Tin-fook Ngai, and Jesse Fang. Dynamic parallelization of single-threaded binary programs using speculative slicing. In *Proceedings of the 23rd international conference on Supercomputing*, pages 158–168. ACM, 2009.

[31] Polychronis Xekalakis, Nikolas Ioannou, and Marcelo Cintra. Mixed speculative multithreaded execution models. *ACM Transactions on Architecture and Code Optimization (TACO)*, 9(3):18, 2012.

[32] Krste Asanovic, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams, et al. The landscape of parallel computing research: A view from berkeley. Technical report, Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, 2006.

[33] Eric S Chung, Peter A Milder, James C Hoe, and Ken Mai. Single-chip heterogeneous computing: Does the future include custom logic, fpgas, and gpgpus? In *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 225–236. IEEE Computer Society, 2010.

[34] Franz Kreupl. Electronics: The carbon-nanotube computer has arrived. *Nature*, 501(7468):495, 2013.

[35] Marc A Manheimer. Cryogenic computing complexity program: Phase 1 introduction. *IEEE Transactions on Applied Superconductivity*, 25(3):1–4, 2015.

[36] Shogo Hatayama, Yuji Sutou, Satoshi Shindo, Yuta Saito, Yun-Heub Song, Daisuke Ando, and Junichi Koike. Inverse resistance change cr2ge2te6-based pcram enabling ultralow-energy amorphization. *ACS applied materials & interfaces*, 10(3):2725–2734, 2018.

[37] Hiroyuki Akinaga and Hisashi Shima. Resistive random access memory (reram) based on metal oxides. *Proceedings of the IEEE*, 98(12):2237–2251, 2010.

[38] John L Hennessy and David A Patterson. *Computer architecture: a quantitative approach*. Elsevier, 5 edition, 2011.

[39] David E Culler, Jaswinder Pal Singh, and Anoop Gupta. *Parallel computer architecture: a hardware/software approach*. Gulf Professional Publishing, 1999.

[40] Michael J Flynn. Very high-speed computing systems. *Proceedings of the IEEE*, 54(12):1901–1909, 1966.

[41] Alexander C Klaiber and Henry M Levy. A comparison of message passing and shared memory architectures for data parallel programs. In *ACM SIGARCH Computer Architecture News*, volume 22, pages 94–105. IEEE Computer Society Press, 1994.

[42] Thomas J LeBlanc and Evangelos P Markatos. Shared memory vs. message passing in shared-memory multiprocessors. In *Parallel and Distributed Processing, 1992. Proceedings of the Fourth IEEE Symposium on*, pages 254–263. IEEE, 1992.

[43] Jack Dongarra, Thomas Sterling, Horst Simon, and Erich Strohmaier. High-performance computing: clusters, constellations, mpps, and future directions. *Computing in Science & Engineering*, 7(2):51–59, 2005.

[44] Charles L Seitz. The cosmic cube. *Communications of the ACM*, 28(1):22–33, 1985.

[45] William Danny Hillis. *The connection machine*. MIT press, 1985.

[46] Charles E Leiserson, Zahi S Abuhamdeh, David C Douglas, Carl R Feynman, Mahesh N Ganmukhi, Jeffrey V Hill, Daniel Hillis, Bradley C Kuszmaul, Margaret A St Pierre, David S Wells, et al. The network architecture of the connection machine cm-5. In *Proceedings of the fourth annual ACM symposium on Parallel algorithms and architectures*, pages 272–285. ACM, 1992.

[47] Top500.org. November 2017 — top500 supercomputer sites. https://www.top500.org/lists/2017/11/, 2017. Accessed 27 May 2018.

[48] Ahmed Hemani, Thomas Meincke, Shashi Kumar, Adam Postula, Thomas Olsson, Peter Nilsson, J Oberg, Peeter Ellervee, and Dan Lundqvist. Lowering power consumption in clock by using globally asynchronous locally synchronous design style. In *Proceedings of the 36th annual ACM/IEEE Design Automation Conference*, pages 873–878. ACM, 1999.

[49] Mark D Hill and Michael R Marty. Amdahl's law in the multicore era. *Computer*, 41(7), 2008.

[50] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, 2009.

[51] Axel Jantsch, Hannu Tenhunen, et al. *Networks on chip*, volume 396. Springer, 2003.

[52] Jose Duato, Sudhakar Yalamanchili, and Lionel M Ni. *Interconnection networks: an engineering approach*. Morgan Kaufmann, 2003.

[53] William James Dally and Brian Patrick Towles. *Principles and practices of interconnection networks*. Elsevier, 2004.

[54] Richard C Holt. Some deadlock properties of computer systems. *ACM Computing Surveys (CSUR)*, 4(3):179–196, 1972.

[55] Klaus Gunther. Prevention of deadlocks in packet-switched data transport systems. *IEEE Transactions on Communications*, 29(4):512–524, 1981.

[56] WJ Dally and CL Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, 36(5):547–553, 1987.

[57] Parviz Kermani and Leonard Kleinrock. Virtual cut-through: A new computer communication switching technique. *Computer Networks (1976)*, 3(4):267–286, 1979.

[58] William J Dally and Charles L Seitz. The torus routing chip. *Distributed computing*, 1(4):187–196, 1986.

[59] Leslie G Valiant and Gordon J Brebner. Universal schemes for parallel communication. In *Proceedings of the thirteenth annual ACM symposium on Theory of computing*, pages 263–277. ACM, 1981.

[60] Ted Nesson and S Lennart Johnsson. Romm routing on mesh and torus networks. In *Proceedings of the seventh annual ACM symposium on Parallel algorithms and architectures*, pages 275–287. ACM, 1995.

[61] Arjun Singh, William J Dally, Brian Towles, and Amit K Gupta. Locality-preserving randomized oblivious routing on torus networks. In *Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures*, pages 9–13. ACM, 2002.

[62] Daniel H. Linder and James C. Harden. An adaptive and fault tolerant wormhole routing strategy for k-ary n-cubes. *IEEE Transactions on computers*, 40(1):2–12, 1991.

[63] Lionel M. Ni and Philip K. McKinley. A survey of wormhole routing techniques in direct networks. *Computer*, 26(2):62–76, 1993.

[64] Ge-Ming Chiu. The odd-even turn model for adaptive routing. *IEEE Transactions on parallel and distributed systems*, 11(7):729–738, 2000.

[65] Andrew A Chien and Jae H Kim. *Planar-adaptive routing: Low-cost adaptive networks for multiprocessors*, volume 20. ACM, 1992.

[66] William J. Dally and Hiromichi Aoki. Deadlock-free adaptive routing in multicomputer networks using virtual channels. *IEEE transactions on Parallel and Distributed Systems*, 4(4):466–475, 1993.

[67] Mukund Ramakrishna, Vamsi Krishna Kodati, Paul V Gratz, and Alexander Sprintson. Gca: Global congestion awareness for load balance in networks-on-chip. *IEEE Transactions on Parallel and Distributed Systems*, 27(7):2022–2035, 2016.

[68] Masoumeh Ebrahimi and Masoud Daneshtalab. Ebda: a new theory on design and verification of deadlock-free interconnection networks. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pages 703–715. ACM, 2017.

[69] Christopher J Glass and Lionel M Ni. The turn model for adaptive routing. *ACM SIGARCH Computer Architecture News*, 20(2):278–287, 1992.

[70] José Duato. A new theory of deadlock-free adaptive routing in wormhole networks. *IEEE transactions on parallel and distributed systems*, 4(12):1320–1331, 1993.

[71] Brian Towles and William J Dally. Worst-case traffic for oblivious routing functions. In *Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures*, pages 1–8. ACM, 2002.

[72] Dana Vantrease, Mikko H Lipasti, and Nathan Binkert. Atomic coherence: Leveraging nanophotonics to build race-free cache coherence protocols. In *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on*, pages 132–143. IEEE, 2011.

[73] Sergi Abadal, Benny Sheinman, Oded Katz, Ofer Markish, Danny Elad, Yvan Fournier, Damian Roca, Mauricio Hanzich, Guillaume Houzeaux, Mario Nemirovsky, et al. Broadcast-enabled massive multicore architectures: A wireless rf approach. *IEEE micro*, 35(5):52–61, 2015.

[74] Srinivasan Murali, Ciprian Seiculescu, Luca Benini, and Giovanni De Micheli. Synthesis of networks on chips for 3d systems on chips. In *Proceedings of the 2009 Asia and South Pacific Design Automation Conference*, pages 242–247. IEEE Press, 2009.

[75] James R Goodman. Using cache memory to reduce processor-memory traffic. *ACM SIGARCH Computer Architecture News*, 11(3):124–131, 1983.

[76] Joel Hruska. How l1 and l2 cpu caches work, and why they're an essential part of modern chips. https://www.extremetech.com/extreme/188776-how-l1-and-l2-cpu-caches-work-and-why-theyre-an-essential-part-of-modern 2017. Accessed 14 May 2018.

[77] Nantero.com. Nantero nram. http://nantero.com/, 2017. Accessed 27 May 2018.

[78] Ping Chi, Shuangchen Li, Cong Xu, Tao Zhang, Jishen Zhao, Yongpan Liu, Yu Wang, and Yuan Xie. Prime: a novel processing-in-memory architecture for neural network computation in reram-based main memory. In *ACM SIGARCH Computer Architecture News*, volume 44, pages 27–39. IEEE Press, 2016.

[79] Moinuddin K Qureshi, Vijayalakshmi Srinivasan, and Jude A Rivers. Scalable high performance main memory system using phase-change memory technology. *ACM SIGARCH Computer Architecture News*, 37(3):24–33, 2009.

[80] Daniel J Sorin, Mark D Hill, and David A Wood. A primer on memory consistency and cache coherence. *Synthesis Lectures on Computer Architecture*, 6(3):1–212, 2011.

[81] Mark S Papamarcos and Janak H Patel. A low-overhead coherence solution for multiprocessors with private cache memories. In *25 years of the international symposia on Computer architecture (selected papers)*, pages 284–290. ACM, 1998.

[82] Andrew W Wilson Jr. Hierarchical cache/bus architecture for shared memory multiprocessors. In *Proceedings of the 14th annual international symposium on Computer architecture*, pages 244–252. ACM, 1987.

[83] James Archibald and Jean-Loup Baer. Cache coherence protocols: Evaluation using a multiprocessor simulation model. *ACM Transactions on Computer Systems (TOCS)*, 4(4):273–298, 1986.

[84] ARM. *ARM Cortex-A57 MPCore Processor Technical Reference Manual*, r1p1 edition, 2013.

[85] Milo MK Martin, Mark D Hill, and David A Wood. Token coherence: Decoupling performance and correctness. *ACM SIGARCH Computer Architecture News*, 31(2):182–193, 2003.

[86] Jason Zebchuk, Vijayalakshmi Srinivasan, Moinuddin K Qureshi, and Andreas Moshovos. A tagless coherence directory. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 423–434. ACM, 2009.

[87] Natalie D Enright Jerger, Li-Shiuan Peh, and Mikko H Lipasti. Virtual tree coherence: Leveraging regions and in-network multicast trees for scalable cache coherence. In *Microarchitecture, 2008. MICRO-41. 2008 41st IEEE/ACM International Symposium on*, pages 35–46. IEEE, 2008.

[88] Ofer Shacham, Megan Wachs, Alex Solomatnikov, Amin Firoozshahian, Stephen Richardson, and Mark Horowitz. Verification of chip multiprocessor memory systems using a relaxed scoreboard. In *Microarchitecture, 2008. MICRO-41. 2008 41st IEEE/ACM International Symposium on*, pages 294–305. IEEE, 2008.

[89] John H Kelm, Daniel R Johnson, William Tuohy, Steven S Lumetta, and Sanjay J Patel. Cohesion: a hybrid memory model for accelerators. In *ACM SIGARCH Computer Architecture News*, volume 38, pages 429–440. ACM, 2010.

[90] Daniel Sanchez and Christos Kozyrakis. Scd: A scalable coherence directory with flexible sharer set encoding. In *High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on*, pages 1–12. IEEE, 2012.

[91] Sarita V Adve and Kourosh Gharachorloo. Shared memory consistency models: A tutorial. *computer*, 29(12):66–76, 1996.

[92] Scott Owens, Susmit Sarkar, and Peter Sewell. A better x86 memory model: x86-tso. In *International Conference on Theorem Proving in Higher Order Logics*, pages 391–407. Springer, 2009.

[93] Luc Maranget, Susmit Sarkar, and Peter Sewell. A tutorial introduction to the arm and power relaxed memory models. *Draft available from http://www. cl. cam. ac. uk/˜ pes20/ppc-supplemental/test7. pdf*, 2012.

[94] Leslie Lamport. How to make a multiprocessor computer that correctly executes multiprocess progranm. *IEEE transactions on computers*, C-28(9):690–691, 9 1979.

[95] Hans-J Boehm and Sarita V Adve. Foundations of the c++ concurrency memory model. In *ACM SIGPLAN Notices*, volume 43, pages 68–78. ACM, 2008.

[96] Chris Guiady, Babak Falsafi, and Terani N Vijaykumar. Is sc+ ilp= rc? In *Computer Architecture, 1999. Proceedings of the 26th International Symposium on*, pages 162–171. IEEE, 1999.

[97] Luis Ceze, James Tuck, Pablo Montesinos, and Josep Torrellas. Bulksc: bulk enforcement of sequential consistency. *ACM SIGARCH Computer Architecture News*, 35(2):278–289, 2007.

[98] Colin Blundell, Milo MK Martin, and Thomas F Wenisch. Invisifence: performance-transparent memory ordering in conventional multiprocessors. In *ACM SIGARCH Computer Architecture News*, volume 37, pages 233–244. ACM, 2009.

[99] Byn Choi, Rakesh Komuravelli, Hyojin Sung, Robert Smolinski, Nima Honarmand, Sarita V Adve, Vikram S Adve, Nicholas P Carter, and Ching-Tsun Chou. Denovo: Rethinking the memory hierarchy for disciplined parallelism. In *Parallel Architectures and Compilation Techniques (PACT), 2011 International Conference on*, pages 155–166. IEEE, 2011.

[100] Joseph Devietti, Benjamin P Wood, Karin Strauss, Luis Ceze, Dan Grossman, and Shaz Qadeer. Radish: always-on sound and complete ra d etection in s oftware and h ardware. In *ACM SIGARCH Computer Architecture News*, volume 40, pages 201–212. IEEE Computer Society, 2012.

[101] Abhayendra Singh, Todd Millstein, Madan Musuvathi, Satish Narayanasamy, and Daniel Marino. End-to-end sequential consistency. *IEEE Micro*, 99(1):1, 2013.

[102] Intel. Introduction to the intel quickpath interconnect, 2009.

[103] ARM. Corelink interconnect. `https://www.arm.com/products/system-ip/corelink-interconnect`. Accessed 10 May 2018.

[104] ARM. Corelink generic interrupt controllers. `https://developer.arm.com/products/system-ip/system-controllers/interrupt-controllers`. Accessed 10 May 2018.

[105] ARM. Corelink cache coherent network family. `https://developer.arm.com/products/system-ip/corelink-interconnect/corelink-cache-coherent-network-family`. Accessed 10 May 2018.

[106] ARM. Corelink memory controllers. `https://www.arm.com/products/system-ip/corelink-memory-controllers`. Accessed 10 May 2018.

[107] Harry Foster. Part 8: The 2014 wilson research group functional verification study. https://blogs.mentor.com/verificationhorizons/blog/2015/07/13/part-8-the-2014-wilson-research-group-functional-verification-study/, 2014. Accessed 29 May 2018.

[108] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R Hower, Tushar Krishna, Somayeh Sardashti, et al. The gem5 simulator. *ACM SIGARCH Computer Architecture News*, 39(2):1–7, 2011.

[109] Jason E Miller, Harshad Kasture, George Kurian, Charles Gruenwald, Nathan Beckmann, Christopher Celio, Jonathan Eastep, and Anant Agarwal. Graphite: A distributed parallel simulator for multicores. In *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on*, pages 1–12. IEEE, 2010.

[110] Shubhendu S Mukherjee, Steven K Reinhardt, Babak Falsafi, Mike Litzkow, Mark D Hill, David A Wood, Steven Huss-Lederman, and James R Larus. Wisconsin wind tunnel ii: a fast, portable parallel architecture simulator. *IEEE Concurrency*, 8(4):12–20, 2000.

[111] Nicola Bombieri, Sara Vinco, Valeria Bertacco, and Debapriya Chatterjee. Systemc simulation on gp-gpus: Cuda vs. opencl. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 343–352. ACM, 2012.

[112] Mahesh Nanjundappa, Anirudh Kaushik, Hiren D Patel, and Sandeep K Shukla. Accelerating systemc simulations using gpus. In *High Level Design Validation and Test Workshop (HLDVT), 2012 IEEE International*, pages 132–139. IEEE, 2012.

[113] Tim Schmidt, Guantao Liu, and Rainer Dömer. Hybrid analysis of systemc models for fast and accurate parallel simulation. In *Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific*, pages 226–231. IEEE, 2017.

[114] Nicolas Ventroux, Julien Peeters, Tanguy Sassolas, and James C Hoe. Highly-parallel special-purpose multicore architecture for systemc/tlm simulations. In *Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV), 2014 International Conference on*, pages 250–257. IEEE, 2014.

[115] Sara Vinco, Valeria Bertacco, Debapriya Chatterjee, and Franco Fummi. Saga: Systemc acceleration on gpu architectures. In *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, pages 115–120. IEEE, 2012.

[116] Jan Henrik Weinstock, Rainer Leupers, Gerd Ascheid, Dietmar Petras, and Andreas Hoffmann. Systemc-link: Parallel systemc simulation using time-decoupled segments. In *Proceedings of the 2016 Conference on Design, Automation & Test in Europe*, pages 493–498. EDA Consortium, 2016.