A study of Spiking Neural Network Architecture:

Report By

Sarath Patlolla

SUID: 764816172

Abstract

Communication in the human brain is done by sending small “packets” of data through complex paths to various destinations, whereas in many conventional computers the communication is based on point to point communication. This led to the inefficiency in the high-speed data transfer in many of the super computers.

Spiking Neural Network Architecture(SpiNNaker) is a parallel, neuromorphic supercomputer architecture which simulates a human brain. This architecture was developed by the University of Manchester. SpiNNaker was generally for high speed computing operations. The SpiNNaker architecture has 65,536 identical 18 core processors of ARM 968 130 mm process, which in total is 1,179,648 cores. Each neuron in this architecture has a link to 6 neighbours (toroidal network)

The SpiNNaker has more than one million cores which is only 1% of the number of neurons in a human brain. Each core in this architecture is programmable and can hold 32KB instruction memory. Diverse types of neural networks can be tested and implemented on this architecture. Signalling chips installed in this architecture are digital. The only limitation in the number of neurons to be added in this architecture are budget and does not depend on any of the fundamental factor.

1.Introduction

1.1 Background

There were a lot of research done to understand how the smallest particle such as sub-atomic works, or else huge bodies such as how the universe works, but the study done on a human brain and its functionality is limited. The primary motivation in designing the SpiNNaker hardware architecture is to understand the human brain completely [1].

A lot of biological discovery was done on the basic component of the brain "the neuron". With brain imaging we could learn how the activity is carried out in the brain in controlling various body parts. The research on these activities is concerned with individual neurons, or a group of neurons, but, the brain is made up of huge number of neurons and understanding this with a biological model is difficult, and we needed to simulate a computer model to completely understand it.

At the University of Manchester, a group of researches called the Advanced Processor Technologies(APT) mainly focusses on the research for advanced and novel approaches for computing and processing. The APT group had a successfully developed the SpiNNaker project. This group had developed this architecture suitable for massive computing in December 2009 [6].

1.2 Introduction

SpiNNaker is a biologically inspired computer to simulate the human brain. It is massively powerful, multi-core, multi-processor neuromorphic computer, developed by University of Manchester, England. This was designed to mainly compute large- scale neural network quickly. The primary reason to design this neural network architecture was to understand and analyse how the human brain functions. The SNN (spiking neural network) is an abstraction of real neural networks.

The primary objectives of the SpiNNaker network are: [6].

- High performance parallel computing suitable for the simulation of the large scale of neural networks in real time
- Investigation of innovative computer architectures, which are different from the conventional super computer architectures, which will lead to a fundamentally new and advanced versions of computer architecture which would be developed which are suitable for massively parallel computing

1.2.1 Neurons and Spikes

The brain is composed of around 100 billion neurons, where every single neuron in the human brain receives a single spike of input and sends a single spike of output. The information transfer among the neurons is done by the transfer of "spikes" among the neurons. The spikes are generated by the electrochemical regeneration process. The spikes have a unit impulse, but the synapse has a variable efficiency [3]. In the computer model the neurons have random weights assigned to them, during the spiking if the weights are positive then the information cans flow from one neuron to the other, whereas if the weight is changed to zero, then there is no information can flow between those neurons.

Synapses, are special point of interaction between these spikes from where the information is transferred from one neuron to the other. The SpiNNaker is built for the "point neuron model" [3], here the structure of the fundamental part of a human brain i.e a neuron its dendrite structure is ignored, and all the inputs and its outputs are taken into consideration.

When modelling this neural network on electronic circuits, the SNN are represented by devices-on-devices graph. The efficiency of this network is generally decided by the speed at which these neurons transfer the information through the network, and this is manipulated by the system designer and these have some limitations.

The SpiNNaker has a chip multiprocessor-die. This contains up-to 1,036,800 ARM9 cores and 7 TB of RAM distributed throughout the system in 57 thousand nodes, and each node contains 18 cores plus a 128 MB SDRAM. Each core is made up of 64KB of DTCM and 32 KB of ITCM [2]. The number of chips on the configuration varies from 1 to 65536 chips. Each chip in the network architecture has a connection to 6 other chips in its neighbourhood, which forms a toroidal network [5].

The SpiNNaker is available for the users via HBP portal. The SpiNNaker available for the users to have a real time processing is available in following variants.

- 4- Node Board (72 processors)
- 48- Node Board (864 processors)
- 24-Board Frame (20,736 processors)
- SpiNNaker Machine (518,400 processors) [6].

1.2.2 Computer Model

There are more than 10^15 synapses taking place in a human brain and to simulate this as a computer model, the power requirements are very high, these synapses fire inputs at an average rate of 10 Hz, and each of the synaptic events would be requiring 10^2 instructions for implementing synaptic plastic algorithm. Since, the number of instructions used are high the number of operations which are performed per second are 10^18. This high number of instructions per second is that of an exascale machine.

Apart from the computer architecture which has the number of neurons like that of a human brain, the other aspect which needed to be addressed is that of the communication and the transfer of the message. In human the information is transferred in the form of “packets” from difficult paths to multiple targets, whereas the mode of transfer in that of the supercomputers the communication done is point-to-point [1].

2. Analysis

2.1 Architectural Overview

The SpiNNaker machine is a multi-data array of nodes. SpiNNaker architecture consists of many parallel processing nodes where each node is 18 ARM968 processor cores where each are 96kB of the local memory and 128MB of the shared memory, the destination to which the packet would be sent is completely dependent on the routing fabric, and the processor has no information about the routing of the data packet [1]. In order to emulate the biological brain in a computer architecture, the information transfer used in SpiNNaker is a packet switched and self-timed network, supporting efficient-muticast and low-delay in communications.

The main component in the communication infrastructure is the on-chip router and the self-timed implementation of the fabric that supports effective extension of the on- chip communications which has inter chip links [8].

2.2 Hardware

2.2.1 SpiNNaker Node

The figure 1 presents the basic structure of node of a SpiNNaker chip. There are multi-core SoC having 20 low-power ARM 968 cores. In each core there is tightly coupled memory having capacity of 32 KB of instructions and 64 KB of data in it.

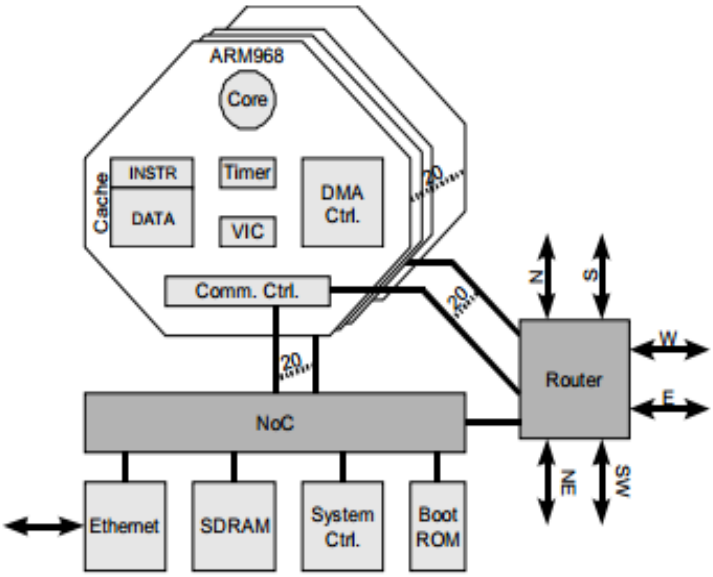


Figure 1: Schematic representation of the SpiNNaker Chip

Every node in the SpiNNaker system is an event driven neural process where events are generated by various modules such as timer, vector interrupt controller (VIC), DMA controller and communication controller. The chips in a SpiNNaker architecture share an SDRAM upto 1GB. Self-timed NoC is responsible for providing with the access of this shared memory which will be used to connect the resources in a chip. Using NoC the bandwidths provided are higher (8 GBps) and the router is accessed only for the configuration purposes, whereas to send or receive packets at the time of normal execution the ARM cores use the communication channel [9]. The chip modulation in biological real time is approximately 1000 neurons which fire at a rate of 10 Hz.

For modelling biological brains the SpiNNaker board with SpiNNaker chips will be utilized which has the largest of the prototyping available. Since, the SpiNNaker consists of 48 chips, they accumulate to 864 ARM processors, out of those 768 are used for neural applications, 48 chips are used for monitoring and the remaining are used as spares. This increases the aggregate memory of the chip to 6GB, which has been spread across multiple chips and cores. The board contains the Xilinx Spartan-6 programmable gate array of chips, which has an inter- board connection, having a speed of 3.1Gbps with serial interfaces(SATA) [10].

2.2.2 NoC Simulators

The usage of the SoC/processor development has shifted from dual and quad core processors to that of the chips to ones with tens or even hundreds of cores that are mutually connected by the Network on chip(NoC) because it would require NoC-based design which would facilitate a correct management of the data.

NS-2 was the first prototype of NoC which was developed and employed for various tasks like prototyping and simulating the conventional networks. The other model which was developed was of DARSIM. DARSIM facilitated the simulation of the mesh of NoC architecture of two and three dimensions. Another model which was developed by SystemC language was “Noxim”. This allowed the users to build a 2D mesh NoC architecture. This allowed the estimation of the NoCs in terms of throughput, latency and power consumption. ORION was a type of the simulator which was developed to estimate the power and space of NoCs [10].

2.3 Software Used

The SpiNNaker software is divided into a type which would be running on the SpiNNaker machine and the software which would be compatible with other systems. The software which runs on the SpiNNaker is written in C and mainly uses the SDP or the multicast needs for the communication process. Further, the software is divided into 2 categories as control software and application software. The interface between the SpiNNaker and the other machines from the outside world is done by IP and Ethernet protocols. UDP/IP protocol is followed for the sending of the packets at the Ethernet interface to the external world.

For the artificial neural network, the neurons in that network consists of weights, and directed graphs with neurons as nodes and the links between the neurons as the edges in that network. The interaction between the neurons for the sending and receiving of the messages is done by spikes which are fired as the dot product of the weights in the nodes and the inputs given to the network. The synapses are transferred from one node to the other by the synaptic connection present in between them. The neural system activators are used as a simple representation for the artificial neurons for the spike to cross the threshold and to fire.

The neural network architecture also caters for various programming models for the users to develop multiple SSN models. Since the biological brain which is used as the base idea for the development for the SpiNNaker network is a heterogeneously structured system, made up of number of neurons which can be differentiated based on functional, physical and anatomical properties.

```
// Two populations have been created; the neuron
// number and type have been set
pre = Population (1000, one_neuron_type)
post = Population (500, another_neuron_type)
// An all-to-all-connection projection between two
// populations has been created; then the weight and
// latency of each connection are set.
excitatory_connections = Projection (pre, post,
    AllToAllConnector (), StaticSynapse (weight = 0.13))
excitatory_connections.set (delay = 0.4)
                    ⋮
```

Figure 2: pseudocode for the simple SSN [4]

In figure 2 the pseudo code has been shown which shows the creation of a couple of neurons and a synaptic edge which connects them both. From the above SSN model the information that can be inferred is: Node information, Edge Information.

SpiNNaker Control and Monitor Program (SC&MP) is the primary software which is mainly responsible for the control of the SpiNNaker system. The SpiNNaker chips is mainly written in primary boot-strap code which would load the Ethernet interface, and this would be loading the SC&MP [1].

The protocol which would load the SpiNNaker architecture is SpiNNaker Datagram Protocol (SDP), this protocol is mainly responsible for the high-level communication to be made and the main part for it to load application. The SDP packets from the SpiNNaker are sent from or to the core is routed by SC&MP. In the cores the SDP packets are exchanged between them using a shared memory interface, and among the chips the packets are transported by point-to-point over the interchip links.

The software in the SpiNNaker system is divided into SpiNNaker side and the other as the host side. The SpiNNaker architecture running on the event bases application run time kernel called the SARK. The SpiNNaker application can run on many cores on the SpiNNaker project. Every SpiNNaker relates to a support module known as SpiNNaker Application Runtime Kernel (SARK), it also maintains a communication interface for SC&MP that runs on the monitor process which is suitable to communicate with the SpiNNaker chip in this module the start-up code for the for the basic application to be built and a runtime environment for the application to run. The memory allocation in the SARK is done efficiently to support SpiNNaker. The application is built using an ARM cross compiler and linked with SARK or other runtime libraries which it requires. Application load and execute (APLX) is the format of the file which is produced as the output file after the linking stage. This format of the file is understood by a simple loader and it is loaded into the suitable parts of the memory of the appropriate cores using a SC&MP loader [1].

Spin1 API is a library which was built to support an event management library which is used by the SpiNNaker applications, this API acts as an interface for the underlying software and the hardware. In this library, many facilities are provided for associating the common interrupts for the event handling of the code and for the management of queues of the events. For the Linux workstation a similar set of APIs have been developed for the SpiNNaker workstation.

2.4 Host Software

There are many tools that have been developed which have been developed for the applications that can run on the SpiNNaker machines. Workstation that is used to control the SpiNNaker system is called a “host”. A command-line interface “ybug” program provides for this function and allows the scripted interface for the system. There are many API that have been developed for the SpiNNaker system in C, Perl and Python.

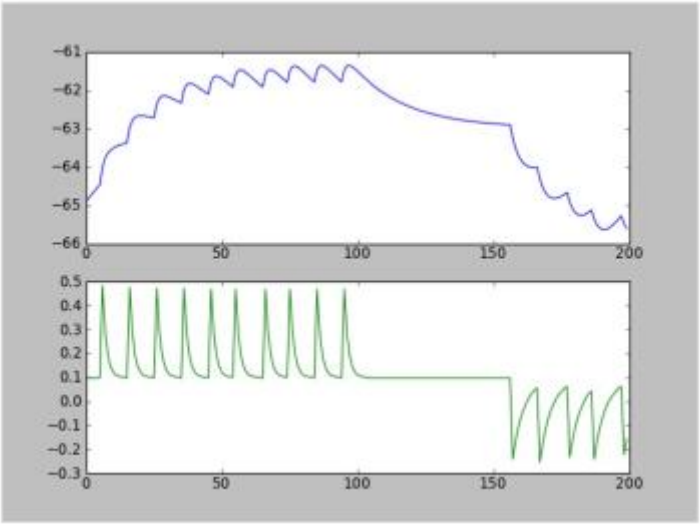


Figure 3: Visualizer output from the SpiNNaker surface

There are several “visualizer” applications which produce the output which can be seen on the host system. The simplest output of the data is shown as a text file whereas the most complex outputs are shown as a graphical visualization. The input to the SpiNNaker network architecture should be given from the software which is designed especially for this purpose only. Spike server is a specially designed software application which is to feed the SpiNNaker architecture [1]. An important aspect of the SpiNNaker system is in the development of software that will solve complex problems on the SpiNNaker system.

The system in the SpiNNaker is triggered by an event, there is always a call back loop to handle the executed event. When the execution of the ARM core is completed for all its callbacks, it switches to the power saving mode.

The following are the SpiNNaker events that are started due to the neural network simulations:

- **Timer Event:** Timer is scheduled to configure a regular periodic time interval, and the neural equations are resolved and then the spike values are updated.
- **Packet Received Event:** A look up step is initiated whenever a spike is received by each process, this requests a DMA transfer of the relevant synaptic transfer from the SDRAM of the chip. Since, the DMA is autonomous, ARM core either immediately served the other events or goes to sleep.
- **DMA Done Event:** When the DMA transfer has completed, it sends a signal to the core, so that core can service the data, such as updating the status of each of the synapse, its weigh and delay [10].

Whenever a synaptic impulse is generated, the ARM core which models this pre-synaptic neuron generates a MC AER packet which contains the source id of the synaptic neuron. The AER packet is sent to the processor chips and the processor cores which contains the firing neurons, and these triggers the packet events. The structure of the membrane joining the neurons is updated and by the DMA request which was processed. The weights of the neurons are maintained by the core of the SpiNNakr architecture and, and the important synaptic information of the neuron is stored in SDRAM chip, which is on demand retrieved.

With the help of a high-level language PyNN the users define the neural network, along with all the weights and the biases. When the neural network architecture is defined by the users, a specialized tool PACMAN (Partitioning And Configuration MANagement) is used to map it to the neural network relying on the available resources. The binary files are generated and uploaded to the SpiNNaker network using PACMAN [11].

2.5 Software model Extraction

The SpiNNaker Neural network model can be extracted by the software simulators:

NEST (Neural Simulation Tool): This tool is capable of modelling various neurons, along with several synaptic models and their plasticity methods.

Nengo is another tool to simulate the software model of the neural network. This tool was developed for the implementation and the support of the engineering framework principles. With this tool mapping a wide range of neuro-computational dynamics to the SNN has become possible.

Every model of the neural network is associated with a different simulator and, by using these opensource on the simulators, accurate neural networks architecture can be built.

2.6 Software model Simulator

Noxim is a simulator tool to model the neural network for a trace-driven cycle simulator. Using this simulator model, we can estimate the amount of power requirements.

2.6.1 Router Pipeline

Tree based routing is used for the multicast routing. The simulator has a pipeline of 4 stage. The nascent stage in the simulator is the buffer write(BW), followed by routing computation(RC), the next stage in the simulator is virtual channel allocation(VCA), and the final stage in the simulator of the neural network is switch allocations(SA). All the stages in the pipeline is executed in a single cycle time and the link traversal to the next router. Since each packet which is sent is only one flit, thus no optimization is required.

In the multicast packets, the virtual circuit for the transfer of packets is decided during the mapping process, i.e. here the routing path is replaced by the routing-table look up, and whenever a packet at any is replicated at a node, the stage at that point is continuously repeated until all the replications at the next node. Common scratch memory is allocated memory for the rather than the CAM, which is expensive. Each value into the routing table holds the 8-bit operation type value, that shows a type in a table or a combination of the values.

Operation Type	Description
Turn_up	It will be transmitted to the up node
Turn_down	It will be transmitted to the down node
Turn_left	It will be transmitted to the left node
Turn_right	It will be transmitted to the right node
Sink	Approaching destination
Valid	1/0
Reserved	2 bits

table 2: Operation Types

2.6.2 Power Consumption

For the optimal power utilisation at each stage of the pipeline in the SpiNNaker architecture Orion tool is used. The number of values to be entered in the scratch memory table should be equal to the number of population. To get the power consumption of each table in the scratch table memory we use CACTI.

2.7 Proposed Architecture

The basic algorithm for the neural network is backprop. The architecture of the SpiNNaker neural network is a dependent on the conventional microprocessor. The backpropagation is a technique to improve the weights in a neural network using gradient descent. The processor of the SpiNNaker architecture would have the record of all the data pertaining to the input, output and the activation states of the model.

The regular processor in a neural network architecture holds a collection of distributed processor nodes(DPN). These DPN's are present in each node in a simulated back-propagation model, for all the architecture of this network, this model sometimes turns out to be inefficient, and thus it would be considered that all the nodes in the architecture have their DPN in their nodes.

Ring structure connection for the processors is an efficient way for the neural network simulation. The SpiNNaker network of the computers would consist of a ring structure of DPN and it also consists of a conventional processor which performs the backprop, which are connected to the ring structure of the DPN, and output of the first DPN is sent to the next DPN [7], and thus the communications among the DPN is synchronized.

In mapping the SpiNNaker model with the back propagation, the following functions which are distributed between the conventional processor and the DPN are:

- Weight storage
- Direct weight manipulation
- Forward pass
- Backward pass
- Weight change
- Indirect weight manipulation

2.8 DPN architecture

DPN is used to process data stream which depends on the type of the instruction. The data elements in the instructions in the DPN are undistinguishable from the change in the instructions of the DPN. This frees the DPN additional memory by eliminating the requirement for any stored program. The functionalities to be performed by the DPN are predetermined and are fixed [7]. Once a new instruction is received by the DPN, the instruction is tagged by a global tag, or a

unique tag, now the instruction is stored and then the instruction is processed on all the incoming new data that is received by the DPN, until a new instruction is received by the DPN. The DPN is provided with a identity id which is unique for every DPN to simulate back propagation so that all the nodes obey all the incoming instructions. Every bit of data received by the DPN is called as an instruction/data.

At the inception of every cycle the register receives an input of 25-bit value which is either an instruction or a data value. If the received INPUT is an instruction then it is loaded in the instruction register, if the input received is a data then the value of the data, then the received instruction received by the register is executed with the data that is received by the register.

The various instructions received by the register are: IDLE, RSET, SACC, SRMA, SRMB, RACC, WMEM, RMEM, CSUM, CERR, UPDM, MODM.

Memory required during the run-time is allocated at runtime and to process and generate the required instructions within the DPN to simulate back-propagation for the fixed number of DPN. The instruction set produced by the compiler are initialisation, iteration and the conclusion sequence [7].

2.8.1 Initialisation Sequence

The initialise sequence initializes the weight to the corresponding memories of DPN at run-time.

2.8.2 Iteration Sequence

Until the back-propagation conditions are met the sequence of iterations is repeatedly applied at the runtime, so that the return values are used to produce the nodal outputs, network output and the node errors.

2.8.3 Conclusion Sequence

The conclusion sequence is used to terminate the conditions at the run time when the conditions have been met. The conclusion sequence is applied to restore the biases and the weight from the DPN.

2.9 SpiNNaker CMP (Chip Multiprocessor) die Implementation

The SpiNNaker chip has embedded in it various IP address associated devices for communication such as SDRAM controller and ARM cores, thus implementation of CMP is complex. Details of the CMP design considerations are:

2.9.1 Implementation Methodology

For the design and the implementation of a SpiNNaker network we deploy the Synopsys Galaxy Design Platform which employed Logical clock gating and the architecture for efficient use of the power and the clock cycles in the architecture. This platform achieved the most efficient and fine-tuned performance for the processor cores. The other core parts of the SpiNNaker system such as AHB, NIM, router, and the memory controller, they were all dependent on this architecture of the design platform. In the complete flow of the instructions or the data of the network, the system was made complete aware of the power requirements for the different tasks. Developing SpiNNaker chips for the conventional CAD models was a challenging task. A hierarchical procedure for the implementation was employed for asynchronous tasks. For the aysnc tasks we need to implement additional assumptions to compensate for the delay-insensitiveness in its logic [2].

2.9.2 Verification and Testing

For the testing and the validation of design and architecture of the SpiNNaker architecture, SystemC is used. The non-synchronous were built in accordance with the early delay in times whereas the synchronous models were modelled were accurate. For the simulation of the models annotated delays and gate-level simulation with extracted parasitics was employed. Initially, the simulation model was tested on the high-level modules, like the router and the cores, once when these high-level modules met the required values, the simulation testing was deployed and tested on the remaining low-level modules. Here, a single core model was simulated via back-annotated detailed model and simulating these models took a longer time, whereas a fast behaviour model was employed for the remaining 17 cores. A careful verification of the mesochronous interface to the off-die SDRAM was carried out. The Delay locked loop(DLL) was an asynchronous core chip that characterized at the transistor-level was controlled at the interface whereas the memory controller which was also asynchronous was simulated and deployed at the gate-level to correct the alignment of the interface signals.

3.Future work and Conclusion

3.1 Future Work

The University of California Berkley is currently working on modelling a connectionist super computer for modelling various neural computation tasks which will be utilized as a crucial tool for the research works to be carried out by the researchers. The SpiNNaker machine model can be deployed in cloud services, so that the services of the simulation of human brain can be used on the distributed systems. As an important aspect of the future work for the SpiNNaker network architecture, more research and study could be carried out on evaluating the system under multiple failure models (bisected system, limited models which have high density of failure rates), and various traffic models that would be favourable for the spatial communication system, which can support finite and long-distance communication in accordance to the causal and temporal relations among various spikes.

An important aspect for the future work is to deliver a neural network machine which has over half a million processors embedded on it, so that it could be used as one of the neuromorphic “platform” to a wider brain project community.

3.2 Conclusion

For the simulating a biological brain most of the time has been spent on understanding analysing the brain and developing the silicon, software and the architecture. Although continuous development is being done for the better software, the architecture and the hardware is producing satisfactory results. SpiNNaker is an application specific, high performance computer architecture which specially designed and optimized for running neuroscience applications, and for the distributed computing applications. The SpiNNaker architecture is an economical and an efficient way of achieving above 10^14 operations processing in one second. The major shortcoming is during the message passing system, between the processors and thus SpiNNaker system is developed for the messages which are multi-cast and short. Due to this optimization has an additional cost is added as the overhead for various applications such as control of the neural networks and for the loading the applications.

In addition to explore the design features of the architecture, the optimal strategy to map the SNN to NoC is important to utilize the hardware resources to the maximum.

4. References

1. Steve B. Furber, Francesco Galluppi, Steve Temple and Luis A. Plana: “The SpiNNaker Project” Vol 102, No 5, May 2014 Proceedings of the IEEE

2. Eustace Painkras, Luis A. Plana, Jim Garside, Steve Temple, Francesco Galluppi, “SpiNNaker: A 1-W 18-Core System-on-Chip for Massively-Parallel Neural Network Simulation” IEEE journal of solid-state circuits, vol. 48, no. 8, august 2013.

3. 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, vol. 62, no. 12, December 2013.

4. Yu Ji, You-Hui Zhang, Wei-Min Zheng, “Modelling Spiking Neural Network from the Architecture Evaluation Perspective” Journal of Computer Science and Technology Jan-2016.

5. *http://www.artificialbrains.com/spinnaker*

6. http://apt.cs.manchester.ac.uk/

7. M. Tureja, “A Computer Architecture to Support Neural Net Simulation”, The computer journal, vol 35 no 4 1992.

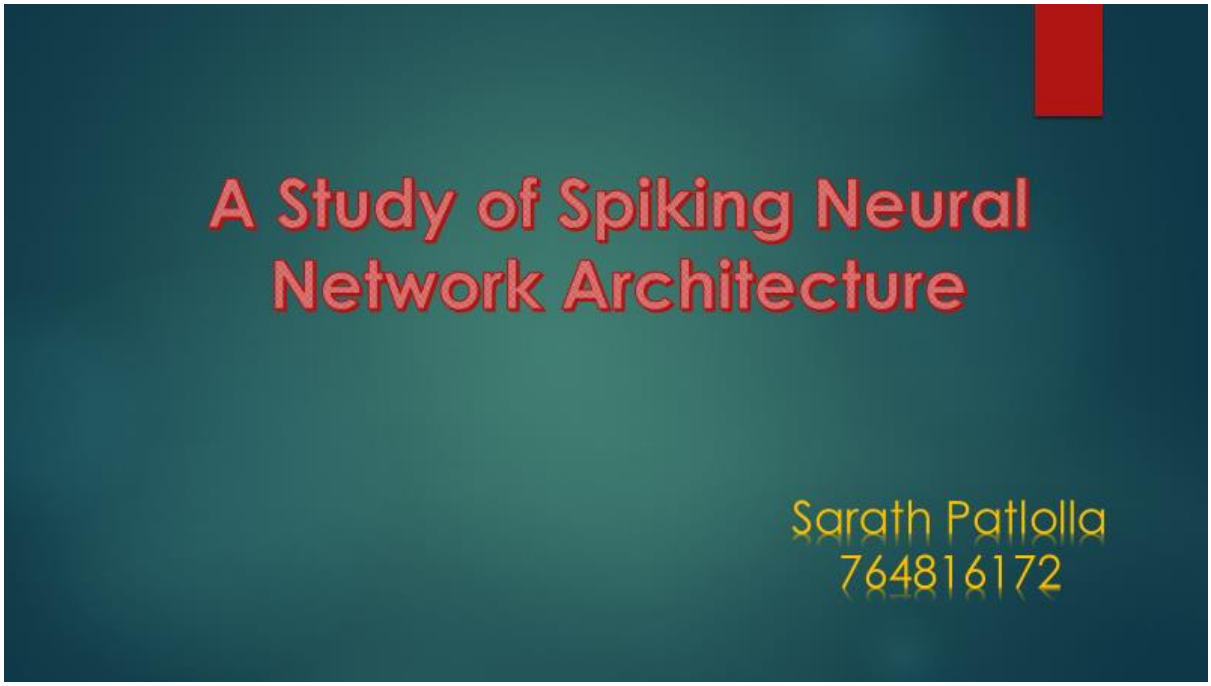
8. J.Navaridas, M.Lujan, J.M.Alonso, Luis A. Plana, Steve Furber, “Understanding the Interconnection Network of SpiNNaker”

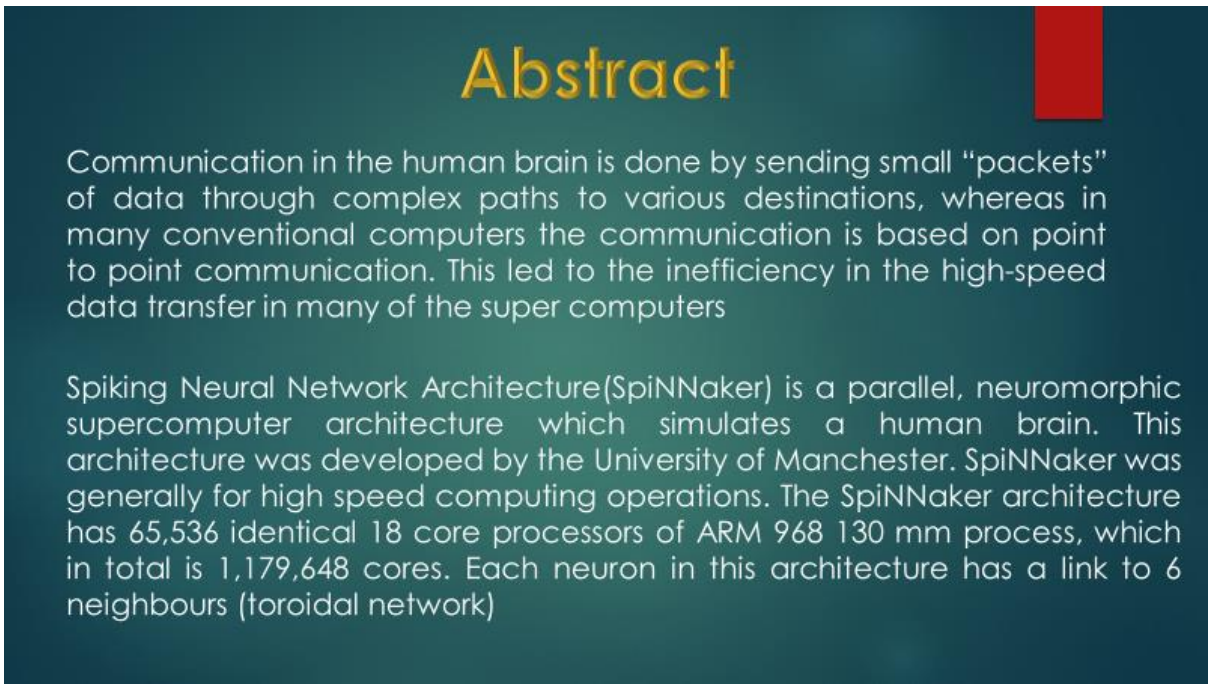
9. S Furber, S Temple, and A Brown, “On-chip and inter-chip networks for modelling large-scale neural systems,” in Proc. International Symposium on Circuits and Systems, ISCAS-2006, Kos, Greece, May 2006.

10. Evangelos Stromatias, Francesco Gallupi, Cameron Patterson and Steve Fuber, “Power analysis of large-scale, real time neural networks on SpiNNaker”.

11. Xavier Lagorce, Evangelos Stromatias, Francesco Gallupi, Luis A. Plana, Shih-Chii Liu, Steve B. Furber and Ryad B. Benosman. “Breaking the millisecond barrier of SpiNNaker: implementing the asynchronous event-based plastic models with microsecond resolution”.

5. Appendix


A Study of Spiking Neural Network Architecture
Sarath Patlolla
764816172


Abstract
Communication in the human brain is done by sending small "packets" of data through complex paths to various destinations, whereas in many conventional computers the communication is based on point to point communication. This led to the inefficiency in the high-speed data transfer in many of the super computers
Spiking Neural Network Architecture(SpiNNaker) is a parallel, neuromorphic supercomputer architecture which simulates a human brain. This architecture was developed by the University of Manchester. SpiNNaker was generally for high speed computing operations. The SpiNNaker architecture has 65,536 identical 18 core processors of ARM 968 130 mm process, which in total is 1,179,648 cores. Each neuron in this architecture has a link to 6 neighbours (toroidal network)

Introduction

- SpiNNaker architecture was developed by University of Manchester in December 2009.
- Biologically inspired to simulate human brain

Neuron and Spikes

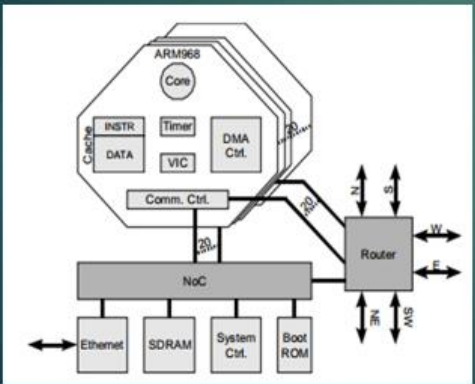
- Neuron is the fundamental part of the SpiNNaker architecture
- The information transfer among the neurons is done by the transfer of "spikes".

SpiNNaker

- SpiNNaker is built from "Point Neuron Model"
- It has a multi processor die
- 1,036,800 ARM9 cores and 7 TB of RAM distributed throughout the system in 57 thousand nodes, and each node contains 18 cores plus a 128 MB SDRAM
- Single core is made up of 64KB of DTCM and 32 KB of ITCM
- In humans the information is transferred in the form of "packets" through complex paths to multiple targets, whereas the mode of transfer in supercomputers is point-to-point

Analysis

SpiNNAker Node



There are multi-core SoC having 20 low-power ARM 968 cores. In each core there is tightly coupled memory having capacity of 32 KB of instructions and 64 KB of data in it.

Every node in the SpiNNaker system is an event driven neural process where events are generated by various modules such as timer, vector interrupt controller (VIC), DMA controller and communication controller.

Software

- The software which runs on the SpiNNaker is written in C and mainly uses the SDP or the multicast for the communication process.
- The interface between the SpiNNaker and the other machines from the outside world is by IP and Ethernet protocols.
- SpiNNaker Control and Monitor Program (SC&MP) is the primary software which is mainly responsible for the control of the SpiNNaker system. The SpiNNaker chips is mainly written in primary boot-strap code which would load the Ethernet interface, and this would be loading the SC&MP.
- The protocol which would load the SpiNNaker architecture is SpiNNaker Datagram Protocol (SDP).
- A command-line interface "ybug" program provides for this function and allows the scripted interface for the system.

Software Model Extraction

The SpiNNaker Neural network model can be extracted by these software simulators:

- NEST
- Nengo
- ORION
- Noxim

DPN Architecture

- DPN is used to process data stream which depends on the type of the instruction.
- The instruction set produced by the DPN compiler are:
  - initialisation sequence
  - iteration sequence
  - the conclusion sequence

Future work

- to deliver a neural network machine which has over half a million processors embedded on it, so that it could be used as one of the neuromorphic "platform" to a wider brain project community.
- The SpiNNaker machine model can be deployed in cloud services, so that the services of the simulation of human brain can be used on the distributed systems.
- more research and study could be carried out on evaluating the system under multiple failure models and various traffic models

References

1. Steve B. Furber, Francesco Galluppi, Steve Temple and Luis A. Plana: "The SpiNNaker Project" Vol 102, No 5, May 2014 Proceedings of the IEEE
2. Eustace Painkras, Luis A. Plana, Jim Garside, Steve Temple, Francesco Galluppi, "SpiNNaker: A 1-W 18-Core System-on-Chip for Massively-Parallel Neural Network Simulation" IEEE journal of solid-state circuits, vol. 48, no. 8, august 2013.
3. 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, vol. 62, no. 12, December 2013.
4. Yu Ji, You-Hui Zhang, Wei-Min Zheng, "Modelling Spiking Neural Network from the Architecture Evaluation Perspective" Journal of Computer Science and Technology Jan-2016.
5. http://www.artificialbrains.com/spinnaker
6. http://apt.cs.manchester.ac.uk/
7. M. Tureja, "A Computer Architecture to Support Neural Net Simulation", The computer journal, vol 35 no 4 1992.
8. J.Navaridas, M.Lujan, J.M.Alonso, Luis A. Plana, Steve Furber, "Understanding the Interconnection Network of SpiNNaker"
9. S Furber, S Temple, and A Brown, "On-chip and inter-chip networks for modelling large-scale neural systems," in Proc. International Symposium on Circuits and Systems, ISCAS-2006, Kos, Greece, May 2006.
10. Evangelos Stromatias, Francesco Gallupi, Cameron Patterson and Steve Fuber, "Power analysis of large-scale, real time neural networks on SpiNNaker".
11. Xavier Lagorce, Evangelos Stromatias, Francesco Gallupi, Luis A. Plana, Shih-Chii Liu, Steve B. Furber and Ryad B. Benosman. "Breaking the millisecond barrier of SpiNNaker: implementing the asynchronous event-based plastic models with microsecond resolution".