System-level Modelling for SpiNNaker CMP System

M.M. Khan, E. Painkras, X. Jin, L.A. Plana, J.V. Woods and S.B. Furber

School of Computer Science, The University of Manchester, UK

Abstract. The SpiNNaker Chip-multiprocessor (CMP) system is a novel SoC architecture, designed specifically for large-scale neural simulations in real-time. We have developed a multi-chip complete system simulation for the SpiNNaker massively parallel CMP system using SystemC Transaction Level Modelling (TLM) to analyse architectural tradeoffs, verify the design, and develop/test intended applications. The model has been very helpful in understanding system-level behaviour in the early stages of the design which was not possible with a Hardware Description Language (HDL) simulation because of the development time, performance and scale of simulation. We could simulate a system with up to 200 SpiNNaker CMPs to test a parallel distributed application developed for the SpiNNaker. The model helped in refining SpiNNaker's design while providing a platform for developing applications in parallel with the hardware design.

1 Introduction

Hardware components can be designed, synthesized and tested with the help of commercially available Computer Aided Design (CAD) tools, which can simulate their exact behaviour in the design phase. However, component-level simulation may fail to exhibit a holistic view of a computing system's functionality. A complete system simulation captures an entire computing system's behaviour at its full instruction set architecture, allowing it to run intended applications unaltered [1]. These simulations allow the development and testing of intended applications while the hardware is still in its design-phase. As these simulations are software-based, they are fully deterministic in behaviour and can be tailored to allow a testbench to be attached for analysis without interfering with the model's functionality. These simulations are especially helpful for testing real-time applications as the application and the underlying hardware can reliably be debugged for real-time analysis [1]. It is also important if the hardware is being designed for a specific application and we want to verify that the design meets its intended objectives. Besides functional correctness, a system-level model can simulate the execution time at the desired temporal resolution. The degree of temporal fidelity, however, depends on the modelling objectives as we need to trade temporal accuracy for simulation speed when simulating a large computing system running an application.

The SpiNNaker CMP system (Figure 1-a) is an Application Specific Integrated Circuit (ASIC) architecture, designed specifically for running large-scale neural network applications in real-time [4]. A full-scale computing system will eventually comprise over one million low-power embedded processing nodes distributed in CMPs to enable running a biologically inspired spiking neural application for over a billion neurons. Each SpiNNaker CMP may contain up to 20 such processing nodes connected over a power-efficient asynchronous Network-on-Chip (NoC). The system itself is made by interconnecting these CMPs (Figure 1-b) to form a system-wide packet-switching asynchronous network that facilitates spike transmission among the neurons as small packets. These packets are directed using a specially designed router on each SpiNNaker CMP which connects it with its six neighboring chips via bidirectional links. Each processing node (Figure 1-d) is a fully functional unit with dedicated memory and other peripherals such as an Interrupt Controller, a Timer, a Communication Controller and a Direct Memory Access Controller, to simulate a bunch of neurons. The 20 on-chip processing nodes share other chip resources such as the router, RAM, ROM, Ethernet Interface and the System Controller etc. (Figure 1-c) with the help of another asynchronous NoC for efficient and low-power memory access. To hold the huge amount of synaptic information associated with about 20,000 neurons on each chip, a dedicated SDRAM of upto 1-GB is provided with each CMP. SpiNNaker is an universal spiking neural network simulator i.e. no specific neural simulation application has been hardwired in the system. To facilitate running most available neural models, we configure the system and load the application at run-time with the help of a Host PC connected to one or more chips using Ethernet connection(s).

We have developed a complete system simulation for a multi-chip SpiNNaker computing system using SystemC TLM for architectural exploration, design verification and application development/validation while the system is still in its design phase. This paper covers our motivation for creating a system-level model for the SpiNNaker computing system. After describing some merits of the SystemC TLM technique, we focus on the modelling methodology adopted, and how it helped in our design flow. To conclude we summarise a few case studies conducted with the model to analyze the system behaviour.

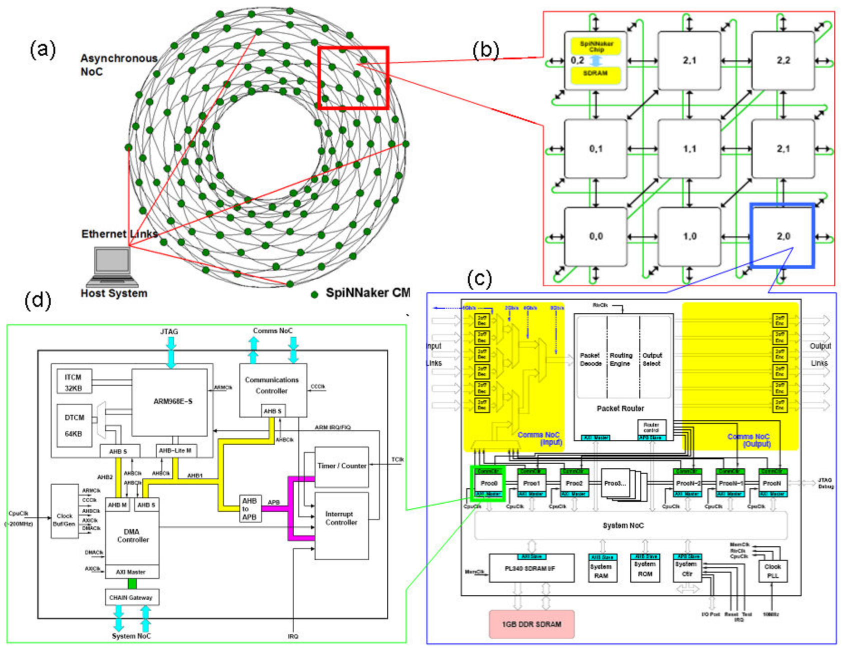

Fig. 1. SpiNNaker Computing System.

2 Simulating a SoC

Hardware Description Languages (HDL), such as Verilog and VHDL, are useful for implementing envisaged specifications and evaluating various architectural trade-offs during the hardware design phase. These are important tools in simulating a hardware artefact's behaviour at Register-Transfer Level (RTL). Once a component is simulated correctly, the simulated code can be synthesized using physical libraries. However, the technique is not suitable for system-level modelling because of its low simulation performance and high development effort. Another disadvantage of using an HDL is its inability to simulate a model at various levels of abstraction as is required for a rapid system-level simulation in the initial phases of system design.

A high-level language, such as C++, can simulate complex system architectures such as a System-on-Chip (SoC) at a much high speed. For example, booting Linux takes 30 seconds on a high-level node simulator of Blue Gene/L (bglsim), while it may take 7 hours on a hardware-accelerated VHDL simulator and more than 20 days with a software VHDL [2]. With a high-level language, we can capture a design at various levels of abstraction. However, a high-level language may not capture an accurate architectural behaviour for lack of standard hardware behavioural constructs available in the HDL and the model developed with C++ is of no use at later phases of the design cycle such as logical and physical compilation. Moreover, in the absence of a standard approach the code may not be reused and shared among various developers [6].

A solution to these issues is a blend of HDL constructs with a high-level language in the form of the SystemC library that provides the ability to achieve clear levels of abstraction with high simulation speed. SystemC supports Transaction Level Modelling (TLM) at a functional level of abstraction. The key to its wide acceptance is an improved productivity through a more reliable design methodology within a shorter time-frame to help start the software development earlier in the SoC design flow i.e. hardware/software co-design [5]. The following are some features of SystemC TLM [5] which motivated the use of this methodology for SpiNNaker system-level modelling.

– Abstraction Levels: SystemC TLM supports the simulation of a system at various levels of abstraction [9]. We could simulate our system as an Untimed Functional Model with Programmers View (UTF-PV) to support algorithmic modelling without any architectural details, as a Functional Untimed model with Architectural View (UTF-AV) to have untimed architectural details, and as a Timed Functional Model with Cycle Approximation (TF-AV(CX)) and Cycle Accuracy (TF-AV(CA)) for accurate timing behaviour. Even at the lowest level of detail and accuracy, the SystemC TLM is many times faster than typical HDL based simulations.
– Architectural Analysis: SystemC TLM offers an oppportunity to explore a system's architecture shortly after initial system specification. An untimed functional model with only functional delays could be used for

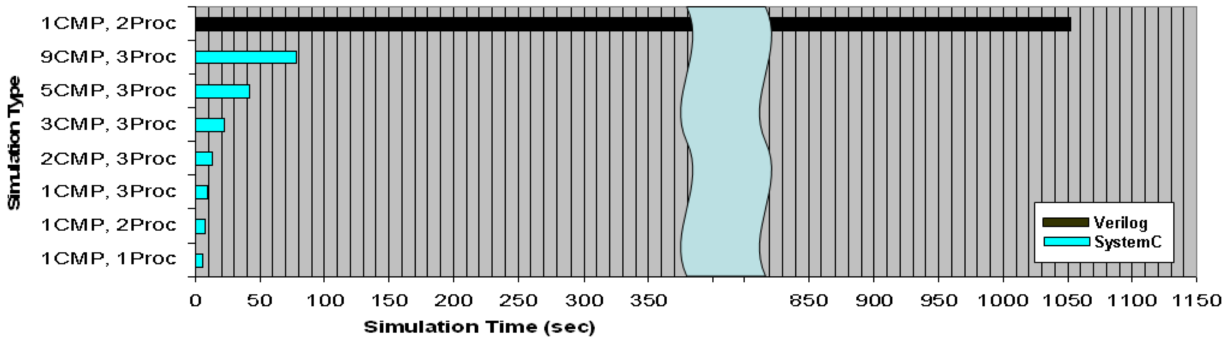

Fig. 2. Simulation Performance, SystemC vs. Verilog Model.

the conceptual feasibility analysis of SpiNNaker system while the timed TLM could be used for thorough architectural analysis in a time- and cost-efficient way.

- Functional Validation: TLM is an executable specification of a given design [5] that captures the behaviour perceived by the system designer. The tests generated with the SystemC model are being used as a reference for functional verification of the RTL models.
- Standard Methodology: SystemC TLM IEEE standard [3] encourages all Electronic Design Automation (EDA) companies and Integrated Circuit (IC) suppliers to comply with the standard of implementing their SystemC class libraries for easy incorporation in SystemC supported SoC design tools. In our case, we could easily use SystemC Intellectual Property (IP) models from Silistix Ltd. and ARM Ltd. with our system-level model for the SpiNNaker system.
- Real-time Debugging: As both the hardware and software are simulated in a high-level language as one package, this can be debugged to investigate real-time hardware-software interaction. This is important for verifying an ASIC design, such as SpiNNaker, as a feasibility study at an early stage of its design.

3 SpiNNaker System-level Modelling

Motivated by its advantages described in Section 2, we have adopted the SystemC TLM approach [6] defined by the Open SystemC Initiative (OSCI) for a system-level simulation of the SpiNNaker computing system. The implementation details of the model follow.

3.1 SpiNNaker UTF-PV Model

As a first step in the system-level modelling, we prepared an algorithmic (UTF-PV) model for a SpiNNaker CMP without any architectural details. The purpose was to depict how a single-chip system will appear to a programmer unconcerned with its architecture. To test the inter-neuron spike communication, algorithmic routing was implemented. The model was based on the initial design specifications from the SpiNNaker datasheet and helped in understanding the system and its communication behaviour as to how the neurons would communicate in a multi-processor environment. The model used modules from OSCI SystemC library to run with the OSCI simulator.

3.2 SpiNNaker UTF-AV Model

We refined the SpiNNaker algorithmic model in the second stage of its evolution to an UTF-AV model with component-level architectural details from a refined version of the SpiNNaker datasheet. Complete functionality and architectural details were incorporated for all the chip components from their specifications in the datasheet. The model was extended to a multi-chip model with some estimated functional delays to acquire an estimated temporal behaviour of the application. Based on our analysis some very important design decisions were taken including the support of a special type of packet for diagnostics and debugging, the need for the System Controller to support chip-level configuration, packet buffering, router error handling, and the need for an Ethernet Controller to connect to the Host PC. The most important contribution of this model was that it exhibited the multi-chip behaviour of the SpiNNaker computing system running a small spiking application which was a feasibility validation for the system design. At this point, we also started to research on how to configure this system in the most dynamic way to support a variety of neural applications. The model was developed with SystemC and TLM libraries to simulate with the OSCI simulator.

3.3 SpiNNaker TF-AV Model

The SpiNNaker UTF-AV model educated us on many aspects of the SpiNNaker computing system, however, it was far from giving real-time communication behaviour for studying its impact on the application. An intended application was developed with the ARM968 Instruction Set Simulator (ISS) on the ARMulator from ARM Ltd. [7] that gave us a confidence in the processing model of the SpiNNaker processing node, however, the work was confined to one processor's execution only. We needed a realistic communication behaviour in a multi-chip system to analyze fully the feasibility of the SpiNNaker system. With this in mind, we transformed our SpiNNaker UTF-AV model into a TF-AV(CX) model by incorporating timing statistics obtained from HDL component-level behavioural simulation. We simulated all synchronous components at their clock speeds. A SystemC Intellectual Property (IP) model for the asynchronous NoC provided by Silistix Ltd. was incorporated into the model with real-time delays acquired from its Verilog simulation. We did not model the ISS processing cores as their SystemC IP was to be provided by ARM Ltd. In the absence of an ISS, the model could not be used for developing and testing applications for the SpiNNaker computing system, however, we could acquire accurate processing times from an ARM968 ISS model from the ARMulator, while the SpiNNaker TF-AV(CX) model provided communication timing corresponding to a multi-chip SpiNNaker application. The two results were combined to produce an estimated overall application behaviour. Further hardware design decisions were taken based on the behavioural results from this simulation, such as changes in the router's design to support legacy multi-layer perceptron neural applications, the System Controller's composition, neighboring chips' fault-recovery mechanisms, and Ethernet Communication with the Host PC etc.

3.4 SpiNNaker TF-AV(CA) Model

The SpiNNaker TF-AV(CX) model did not support the development and running of an application that could run on the SpiNNaker hardware. This was due to the non-availability of a stand-alone SystemC IP for the ARM968 processing core. The SystemC model for the ARM968 processor and other ARM based components required us to run the model as part of ARM's SoC Designer with its own simulator. In the third stage of the SpiNNaker system-level modelling, we ported all our modelled components into SoC Designer. The SystemC IP models for all the components provided by ARM, such as the ARM968 ISS, Interrupt Controller, Timer, Watchdog Timer, buses and memories, were included from ARM-provided SystemC libraries. As the SoC Designer simulates timing at the minimum granularity of a component's clock cycle, we simulated the asynchronous NoC communication with the driving components' clock cycles, e.g. the 13-15 ns delay at the inter-chip interface has been simulated by 3 router clock cycles at 5 ns per cycle. The model can be used for application development, testing and running just as the actual hardware. We use the ARMCC compiler and ARMASM assembler from ARM Ltd. to produce an ARM968 instruction set binary image to be loaded into the Boot ROM. We can simulate a single- or multi-chip configuration with a varying number of processing cores. The model is being used extensively for design validation and application development/debugging by the SpiNNaker software team. The process of validation is to develop a test case with the TF-AV(CA) model and to run it on the Verilog model for component- or system-level behavioural testing. The model supports debugging the hardware, software or both simultaneously in the real application time, enabling us to spot the causes of error exactly. The simulation performance for running a sample configuration code on the SpiNNaker TF-AV(CA) model vs. the Verilog top-level behavioural model is shown in Figure 2.

4 Functional Validation

For component-level functional validation, we calibrated the behaviour of each natively designed component's SystemC model with its HDL behavioural simulation at cycle-level granularity for an accurate functional and timing behaviour. To test the functional correctness of the system at various levels of abstraction we carried out various case studies which gave us confidence in the system design. The details of these case studies are described in the following sections.

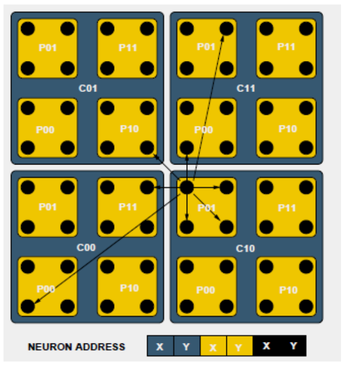

Fig. 3. Spiking NN Model [8].

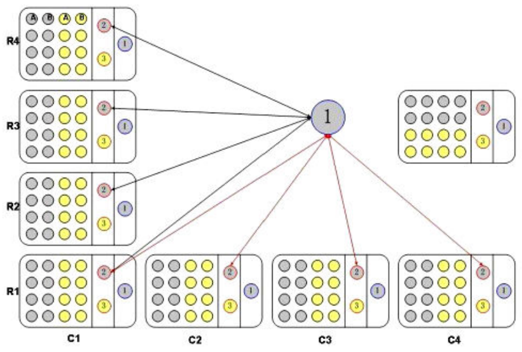

Fig. 4. MLP NN Model [8].

4.1 Case Study1

This was a very simple case study designed to verify the communication infrastructure of the multi-chip SpiNNaker computing system with the help of SpiNNaker UTF-AV model as explained by MM Khan *et al.* [8]. We created a spiking neural network environment to be modelled on the SpiNNaker with a varying number of neurons with random inter-neuron connectivity. The environment was created with the help of a C++ application (SpiNNit) developed to help map neurons onto the processing nodes and to generate the routing tables for the defined neural network connectivity. The application also generated the spike pattern to be sent by each neuron and the expected results to be compared with those from the system-level model for validation purposes. For this case study, each chip contained four application processors and one monitor processor while the system consisted of 4x4 (16) chips as shown in Figure 3. While the application processors recorded the received packets, monitor processors captured any packets dropped due to communication errors. We used the SystemC Verification (SCV) library for transaction recording to verify the communication behaviour with expected results. The experiment was run several times with varying neural mapping. After a few passes of debugging and fine tuning, the results matched the expected behaviour. This case study also helped in the traffic analysis in a multi-chip SpiNNaker system.

4.2 Case Study II

This case study was intended as a feasibility study to run legacy Multi-Layer Perceptron (MLP) neural network applications on SpiNNaker [8]. The application was run on the SpiNNaker TF-AV(CX) model for the communication results and on ARM's ARMulator for processing results. We carried out this case study with researchers in the School of Psychological Sciences, at the University of Manchester for a character recognition application. They use a PC Cluster for running the application with the Light Efficient Network Simulator (LENS). The purpose of this case study was to establish the feasibility of running legacy neural applications on SpiNNaker, and also to evaluate the performance gain over a PC cluster.

A typical MLP neural network learning algorithm consists of an input layer, an output layer and a number of hidden layers with varying connectivity. We mapped one neuron onto each of 19 application processors in each chip as shown in Figure 4. Each neuron was to compute the partial result of the input received and to hand the results over to the next layer's neurons, untill the neurons in the output layer produce the 'delta' (error) as a feedback for backpropagation. The routing tables for the multicast packets in forward and backward paths were computed with the help of SpiNNit for various sizes of the SpiNNaker system. We implemented the application in C++ on the system-level model incorporating the processing delays acquired from the ARMulator. We simulated a 200+ chips SpiNNaker computing system, and the simulation results showed a considerable performance gain over a PC cluster [8]. The case study gave us confidence that the system is equally useful for simulating MLP neural networks on the SpiNNaker computing system.

4.3 Case Study III

We carried out this case study with the help of the SpiNNaker TF-AV(CA) model with the ISS model for ARM968 processing cores. The application was the most suitable for the SpiNNaker architecture and used an event-driven real-time application model [7] to simulate 1000 neurons on each processing core with a random connectivity

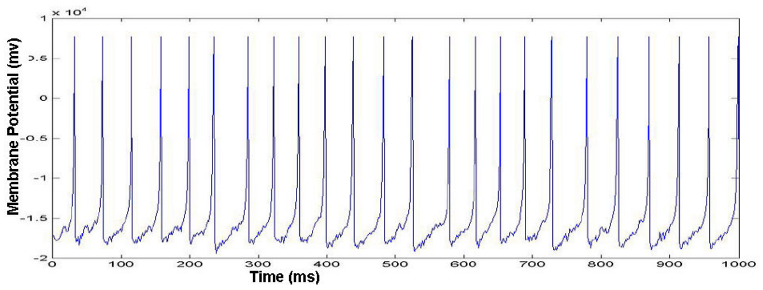

Fig. 5. Spike Train from Izhikevich Neurons.

among the neurons defined by the routing tables. The synaptic weights and axonal delays for inter-neuron synapses were stored in the SDRAM with each chip. The spikes were passed among the neurons as multicast packets which were routed by the on-chip router to the relevant on-chip/off-chip processors as per the router's configuration. The results produced (Figure 5) are comparable with those produced with the ARMulator for a single processing core [7]. These results provide satisfactory verification of the design and functionality of the SpiNNaker system.

5 Summary

A system-level model gives a holistic view of a computing system design, which is not possible with its component level simulation. Simulating a massively parallel computing system of the scale of SpiNNaker is not possible using HDL in the early stages of its design for performance and development time considerations. A complete system's simulation for the SpiNNaker computing system has been developed using the SystemC TLM technique as part of the SpiNNaker research project. The model has contributed effectively in refining the architectural details, evolving/resolving system level functionality issues, and verifying the functionality of the system at component and system level. We adopted SystemC TLM while evolving the system-level model from its algorithmic modelling (UTF-PV) to the cycle accurate TF-AV(CA) with full architectural functionality. The end product is similar in functionality to the actual hardware. We are able to develop, debug and test applications intended for the actual SpiNNaker computing system, while the system is still in its design phase. We have carried out extensive testing to verify the model to ensure a behaviour close to that of the actual hardware. We verified the design objectives of SpiNNaker computing system with the help of case studies to test very different types of potential applications for SpiNNaker.

References

1. Lars Albertsson and Peter S. Magnusson. Using Complete System Simulation for Temporal Debugging of General Purpose Operating Systems and Workload. In *MASCOTS '00*, page 191, Washington, DC, USA, 2000.
2. G. Almasi and et al. An Overview of the BlueGene/L System Software Organization. In *Euro-Par '03 Conference on Parallel and Distributed Computing*, 2003.
3. Design Automation Standards Committee. *IEEE Standard SystemC Language Reference Manual - std. 1666-2005*. IEEE Computer Society, Mar. 2006.
4. Steve Furber and Steve Temple. Neural Systems Engineering. *J. R. Soc. Interface*, 4(13):193–206, April 2007.
5. Frank Ghenassia, editor. *Transaction-Level Modeling with SystemC: TLM Concepts and Applications for Embedded System*. Springer Publishers, New York, NY, USA, 2005.
6. Thorsten Grötker, Stan Liao, Grant Martin, and Stuart Swan. *System Design with SystemC*. Kluwer Academic Publishers, Boston, 2002.
7. X. Jin, S.B. Furber, and J.V. Woods. Efficient Modelling of Spiking Neural Networks on a Scalable Chip Multiprocessor. In *Proc. Int'l Joint Conf. on Neural Networks (IJCNN2008)*, 2008.
8. M.M. Khan, D.R. Lester, L.A. Plana, A. Rast, X. Jin, E. Painkras, and S.B. Furber. Spinnaker: Mapping Neural Networks onto a Massively-Parallel Chip Multiprocessor. In *Proc. IJCNN2008*, 2008.
9. Doulos Ltd. *Comprehensive SystemC Training - Training Manual*. Doulos Ltd, UK, 2006.