

# System-level Model 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 massively parallel Chip-multiprocessor (CMP) system<sup>1</sup> is a novel SoC architecture, being designed specifically for large-scale neural simulations in real-time. We have developed a multi-CMP complete system’s simulation for the SpiNNaker computing system using SystemC Transaction Level Modelling (TLM) to analyze 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 for 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 actual hardware. The model helped in refining SpiNNaker’s design while providing a platform for developing its applications in parallel with the hardware design.

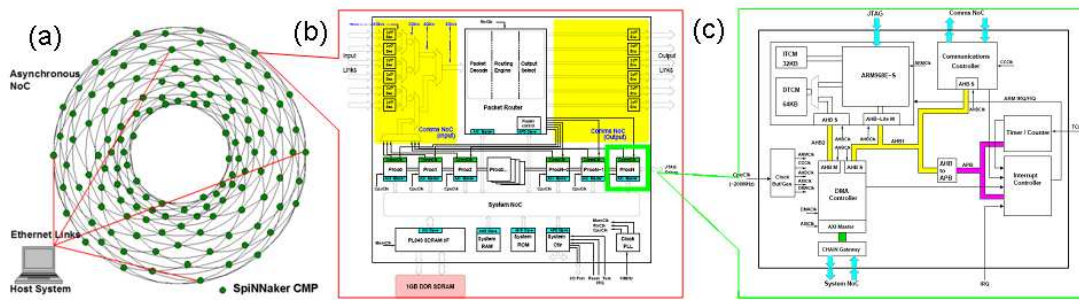


Fig. 1. SpiNNaker Computing System.

## 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 developing and testing of intended applications while the actual hardware is still in its design-phase. As these simulations are software-based, they are fully deterministic in behaviour and can be tailored to attach a testbench for analysis without interfering with the actual system. A system-level model also helps temporal debugging of an entire target system along with the application simultaneously. It is important if the hardware is being designed for a specific type of application and we want to verify the design for its intended purposes. These simulations are especially helpful for testing real-time systems as the application and the underlying hardware can reliably be debugged for real-time analysis [1]. Besides functional correctness, a system-level model can reliably capture its execution time at the desired temporal resolution. For a simulation of a large computing system running a large application, we need to trade temporal

<sup>1</sup> The SpiNNaker project is supported by EPSRC grant EP/D07908X/1, ARM and Silistix Ltd. Steve Furber holds a Royal Society-Wolfson Research Merit Award. email: kxanm@cs.man.ac.uk

accuracy for speed by using low-level abstraction. The degree of abstraction, however, depends on the modelling objectives.

The SpiNNaker CMP system (Fig. 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 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 asynchronous links. Each processing node (Fig. 1-c) is a fully functional unit with dedicated memory and other peripherals such as a Vectored Interrupt Controller, a Timer, a Communication Controller and a Direct Memory Access Controller. This is able to run a neural dynamics application simulating a bunch of neurons (1000). 20 on-chip processing nodes share other chip resources such as the router, RAM, ROM, Ethernet Interface and the System Controller (Fig. 1-b) with the help of another asynchronous NoC for efficient and low-power memory access. To hold huge amount of synaptic information associated with 20,000 neurons on each chip, a dedicated SDRAM of 128MB is provided with each CMP. The CMPs themselves are interconnected in the form of a toroidal mesh (Fig. 1-a). The SpiNNaker is a 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 the run-time with the help of a Host PC connected to one or more chips using the Ethernet connection(s).

We have developed a system-level model for a multi-chip SpiNNaker computing system with all its architectural details 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 shall focus on the modelling methodology adapted, and how it helped in our design flow. To conclude we shall summarise a few case studies conducted with the model to analyze system behaviour with a few intended applications.

## 2 Simulating an SoC

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

High-level languages such as C++ can simulate complex system architectures such as a System-on-Chip (SoC) at a much higher speed. For example, booting Linux takes 30 seconds on a C++ based high-level simulator, 7 hours on a hardware-accelerated VHDL simulator and more than 20 days with standard VHDL [2]. With a high-level language, we can capture the design views 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].

## 3 SystemC Transaction-Level Modelling

A solution to the issues raised in Sec. 2 is a blend of HDL constructs with a high-level language in the form of the SystemC library that provides an ability to achieve clear levels of abstraction with high simulation speed. SystemC has introduced Transaction Level Modelling (TLM) at a functional level of abstraction [10]. The key to its wide acceptance is an improved productivity through a more reliable design methodology within a shorter time-frame to help starting the software development earlier in the

SoC design flow i.e. a hardware/software co-design [5]. The following are the main features of SystemC TLM [5] which motivated the use of this methodology for the SpiNNaker system-level modelling.

- Abstraction Levels: SystemC TLM supports the simulation of a system at various levels of abstraction [8]. We could simulate our system as an Untimed Functional Model with Programmer’s View (UTF-PV) without any architectural details, as a Untimed Functional model with Architectural View (UTF-AV) to have untimed architectural details, and as a Timed Functional Model with Cycle Approximation (TF-AV(CX)) or Cycle Accuracy (TF-AV(CA)) with accurate timing behaviour.
- Architectural Analysis: SystemC TLM offers an opportunity to explore a system’s architecture shortly after initial system specification. An UTF-PV with only functional delays could be used for 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 behavior perceived by the system designer. The tests generated with SystemC model are being used as reference for functional verification of 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 in implementing their SystemC class libraries for easily incorporating 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 the SpiNNaker, as a feasibility study at an early stage of its design.

## 4 SpiNNaker System-level Modelling

Motivated by its advantages described in Sec. 3, we have adopted the SystemC TLM approach [5, 6] defined by Open SystemC Initiative (OSCI) for a system-level simulation of the SpiNNaker computing system. The implementation details of the model follow.

### 4.1 SpiNNaker UTF-PV Model

As a first step in the system-level modelling, we prepared an algorithmic (UTF-PV) model for a SpiNNaker CMP. The purpose was to depict how a single-chip system will appear to a programmer not concerned with its architecture. To test the packet-based inter-neuron spike communication, an algorithmic routing functionality was provided. The model was based on the initial design specifications from the SpiNNaker datasheet [11] and helped in understanding the system and its communication behaviour as to how the neurons would communicate with each other. The model used modules from OSCI SystemC library 2.0 to run with OSCI simulator.

### 4.2 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 in the CMP components captured 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 purpose, need for the System Controller to support chip-level control functions, packet buffering, router error handling mechanism, need for an Ethernet to connect to the Host PC, design of inter-chip interface etc. The most important contribution of this model was that it exhibited a multi-chip behaviour of the SpiNNaker computing system running small spiking application which was a feasibility validation for the system design. We developed most modules with SystemC and TLM libraries to simulate with OSCI simulator.

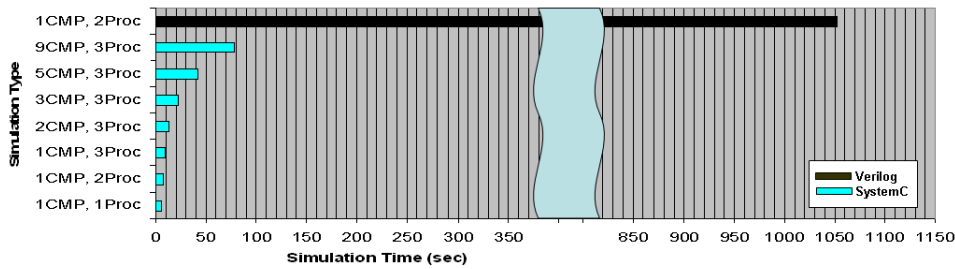


Fig. 2. Simulation Performance, SystemC vs. Verilog Model.

### 4.3 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 ARM 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 processing node only. We needed some 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, while 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 an ISS processing cores as a SystemC IP was to be provided by ARM. Some more 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 (MLP) neural applications, System Controller's composition, neighboring chips' fault-recovery mechanism, and Ethernet Communication with the Host PC etc.

### 4.4 SpiNNaker TF-AV(CA) Model

The SpiNNaker TF-AV(CX) model did not support the developing 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 ARM968 processing core. The SystemC model for ARM968 processor and other ARM based components required us to run the model as part of ARM SoC Designer with its own simulator. In the third stage of the SpiNNaker system-level modelling, we ported all our modelled components into the SoC Designer and used the SystemC models for all the components provided by ARM, such as ARM968 ISS, Interrupt Controller, Timer, AHB, Watchdog Timer, and memories to develop a TF-AV(CA) model. As the SoC Designer simulates timing at the minimum granularity of a component's clock cycle, we simulated asynchronous NoC communication real-time delays with the relevant component's clock cycles, e.g. 13-15 nano second delay at the inter-chip interface has been simulated by 3 router clock cycles at 5 nano second 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 to produce an ARM968 instruction set binary image to be loaded to the Boot ROM in the modelled CMP. We can simulate a single- or multi-chip configuration with varying number of processing cores with the help of parameters. The model is being used extensively for design validation and application development/debugging purposes 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 a component- or system-level behavioural testing. The model supports debugging the hardware, software or both simultaneously in the real application time, enabling us to exactly spot the cause of error. 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 Fig. 2.

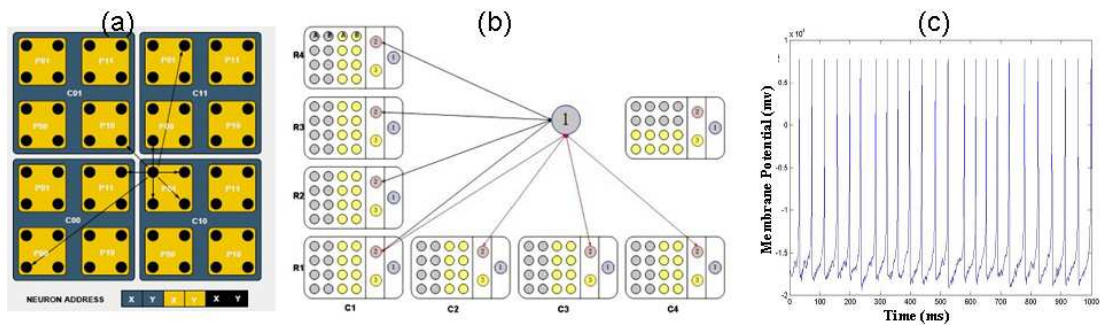


Fig. 3. (a) Spiking NN Model, (b) PDP NN Mapping, (c) Izhikevich NN Model [9].

## 5 Functional Validation

For component-level functional validation, we compared the behaviour of each natively designed component's SystemC model with its HDL behavioural simulations at cycle-level granularity and calibrated 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 a confidence on the system design. The details of these case studies are described in the following sections.

### 5.1 Case Study I

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 [9]. We created a spiking neural network environment to be modelled on the SpiNNaker with varying number of neurons using random inter-neuron connectivity. The environment was created with the help of a C++ application (SpiNNit) developed to help mapping neurons onto the processing nodes and generating 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 purpose. 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 Fig. 3-a. While the application processors recorded the received packets, monitor processors captured any packets dropped to the monitor processor due to some error such as long lasting congestion, parity or time-phase errors. We used the SystemC Verification (SCV) library for transaction recording to verify the communication. The experiment was run several times with varying neural mapping for verifying the communication behaviour. After a few passes of debugging and fine tuning, the results matched the expected behaviour.

### 5.2 Case Study II

This case study was intended as a feasibility study to run legacy MLP neural network applications on SpiNNaker [9]. The application was run on the SpiNNaker TF-AV(CX) model for the communication results and ARM ARMulator for processing delays in the absence of processor's ISS model in the system-level model. We then combined the two results to acquire unified application timings. 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 carried out this case study with researchers in the School of Psychological Sciences, at the University of Manchester. The purpose of this case study was to establish the feasibility of running legacy neural applications on the SpiNNaker, and also to evaluate the performance gain over a PC cluster. The routing tables for forward and backward path multicast packets were computed with the help of SpiNNit for various sizes of the SpiNNaker system. We mapped one neuron on each of 19 application processors in each chip as shown in Fig. 3-b. Each neuron was to compute the partial result of the input received and to hand the results over to the next layer's neurons, till the neurons in output layer produce the 'delta' (error) as a feedback for backpropagation. We implemented the PDP model in C++ on the system-level model incorporating the processing delays acquired from ARM ARMulator. We simulated a 200+ chips SpiNNaker computing system. The simulation results gave us a confidence that the SpiNNaker

computing system is equally useful for simulating legacy MLP neural networks with performance better than a PC cluster [9].

### 5.3 Case Study III

We carried out this case study with the help of SpiNNaker TF-AV(CA) model with the ISS model for ARM968 processing cores. The application was the most suitable for SpiNNaker architecture and used an event-driven real-time application model [7] to simulate 1000 neurons on each processing core with a random connectivity among the neurons defined by the routing tables. The synaptic weights and axonal delays for inter-neuron synapses were stored in each chip's SDRAM. The spikes were passed among the neurons as multi-cast 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 (Fig. 3-c) are comparable with the ones produced with the ARMulator for a single processing core [7]. These results provide satisfactory verification of the design and functionality of the SpiNNaker system.

## 6 Summary

A system-level model gives a holistic view of the functionality of a computing system. Simulating a large scale massively parallel computing system of the scale of the SpiNNaker, in the early stages of its design, was not possible using conventional HDL in terms of performance and development time. A complete system-level simulation for the SpiNNaker computing system has been developed using the SystemC TLM technique as part of SpiNNaker research project. The model has contributed effectively in refining the architectural design, evolving/resolving system level functionality issues, and verifying the functionality of the system at component and system level. The SpiNNaker system-level model has evolved from its algorithmic (UTF-PV) to the cycle accurate TF-AV(CA) model with full architectural functionality. We have carried out an 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 three case studies to test two very different types of potential application for SpiNNaker. 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.

## References

1. L. Albertsson and P.S. Magnusson. Using Complete System Simulation for Temporal Debugging of General Purpose Operating Systems and Workloads. In *In Proceedings of Mascots 2000*, pages 191–198. Society Press, 2000.
2. G. Almasi. 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*. IEEE Computer Society, std. 1666-2005 edition, Mar. 2006.
4. S.B. Furber and S. Temple. Neural Systems Engineering. *J. R. Soc. Interface*, 4(13):193–206, April 2007.
5. F. Ghenassia(ed). *Transaction-Level Modeling with SystemC: TLM Concepts and Applications for Embedded System*. Springer Publishers, New York, NY, USA, 2005.
6. T. Grtker, S. Liao, G. Martin, and S. 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. 2008 Int'l Joint Conf. on Neural Networks (IJCNN2008)*, 2008.
8. L. John. *Comprehensive SystemC Training - Training Manual*. Doulos, 2006.
9. 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. 2008 Int'l Joint Conf. on Neural Networks (IJCNN2008)*, 2008.
10. P.R. Panda. Systemc a Modeling Platform Supporting Multiple Design Abstractions. In *ACM ISSS 01*, Montreal Quebec, Canada, 2001.
11. The SpiNNaker Project. *SpiNNaker - a Chip Multiprocessor for Neural Network Simulation*. The University of Manchester, 0.5 (draft) edition, Nov. 2007.