Neural Process Lett
DOI 10.1007/s11063-012-9274-5

Modular Neural Tile Architecture for Compact Embedded Hardware Spiking Neural Network

Sandeep Pande · Fearghal Morgan · Seamus Cawley · Tom Bruintjes · Gerard Smit · Brian McGinley · Snaider Carrillo · Jim Harkin · Liam McDaid

Abstract Biologically-inspired packet switched network on chip (NoC) based hardware spiking neural network (SNN) architectures have been proposed as an embedded computing platform for classification, estimation and control applications. Storage of large synaptic connectivity (SNN topology) information in SNNs require large distributed on-chip memory, which poses serious challenges for compact hardware implementation of such architectures. Based on the structured neural organisation observed in human brain, a modular neural networks (MNN) design strategy partitions complex application tasks into smaller subtasks executing on distinct neural network modules, and integrates intermediate outputs in higher level functions. This paper proposes a hardware modular neural tile (MNT) architecture that reduces the SNN topology memory requirement of NoC-based hardware SNNs by using a combination of fixed and configurable synaptic connections. The proposed MNT contains a 16:16 fully-connected feed-forward SNN structure and integrates in a mesh topology NoC communication infrastructure. The SNN topology memory requirement is 50 % of the monolithic NoC-based hardware SNN implementation. The paper also presents a lookup table based SNN topology memory allocation technique, which further increases the memory utilisation efficiency. Overall the area requirement of the architecture is reduced by an average of 66 % for practical SNN application topologies. The paper presents micro-architecture details of the proposed MNT and digital neuron circuit. The proposed architecture has been validated on a Xilinx Virtex-6 FPGA and synthesised using 65 nm low-power CMOS technology. The evolvable capability of the proposed MNT and its suitability for executing subtasks within a MNN execution architecture is demonstrated by successfully evolving benchmark SNN application tasks representing classification and non-linear control functions. The paper

S. Pande (✉)· F. Morgan · S. Cawley · B. McGinley
Bio-Inspired Electronics and Reconfigurable Computing, National University of Ireland, Galway, Ireland
e-mail: sandeep.pande@nuigalway.ie

T. Bruintjes · G. Smit
Computer Architecture for Embedded Systems, University of Twente, Enschede, The Netherlands

S. Carrillo · J. Harkin · L. McDaid
Intelligent Systems Research Centre, University of Ulster, Derry, Northern Ireland, UK

Published online: 01 January 2013

addresses hardware modular SNN design and implementation challenges and contributes to the development of a compact hardware modular SNN architecture suitable for embedded applications

Keywords Modular neural networks (MNN) · Spiking neural networks (SNN) · Synaptic connectivity · Network on chip (NoC) · EMBRACE

1 Introduction

Artificial neural network (ANN) computing techniques, which are primarily inspired by the functioning of human brain, can provide promising solutions for designing complex and intelligent systems [1]. The organic central nervous system includes a dense and complex interconnection of neurons and synapses, where each neuron connects to thousands of other neurons through synaptic connections. Computing systems based on spiking neural networks (SNNs) emulate real biological neural networks, conveying information through the communication of short transient pulses (*spikes*) between neurons via their synaptic connections. Each neuron maintains an internal *membrane potential*, which is a function of input spikes, associated synaptic weights, current membrane potential, and a constant membrane potential *leakage coefficient* [2,3]. A neuron *fires* (emits a spike to all connected synapses/neurons) when its membrane potential exceeds the neuron's firing threshold value.

Brain-inspired computing paradigms such as SNNs offer the potential for elegant, low-power and scalable methods of embedded computing, with rich non-linear dynamics, ideally suited to applications including classification, estimation, prediction, dynamic control and signal processing. The efficient implementation of SNN-based hardware architectures for real-time embedded systems is primarily influenced by neuron design, scalable on-chip interconnect architecture and SNN training/learning algorithms [4]. The authors have investigated and proposed EMBRACE[1], as an embedded hardware neural network architecture [5,6]. The EMBRACE NoC-based SNN architecture is a 2D mesh topology array of neural tiles each comprising a single neuron and NoC router. The NoC-based synaptic connectivity approach employed in the EMBRACE architecture provides a flexible, packet-switched inter-neuron communication channels, scalable interconnect and connection reconfigurability [7,8].

SNN application topologies are characterised by a large number of synaptic connections that translate to large distributed on-chip memory in packet switched NoC-based hardware SNN architectures. The storage requirement for large SNNs poses a serious challenge for their compact hardware implementation. The modular neural networks (MNN) design strategy partitions complex application tasks into smaller subtasks executing on distinct neural network modules and integrates outputs in higher level functions [1,9]. The neural network modules in MNN execution architectures maintain internal communication, which is isolated from other modules and the global communication infrastructure. The orthogonalisation of synaptic connectivity suggested in the MNN design paradigm can help tackle the large connectivity problem of SNN architectures and offer practical system implementation for embedded applications.

This paper proposes a hardware modular neural tile (MNT) tile architecture, that reduces the SNN topology memory requirement of NoC-based hardware SNNs by using a combination of fixed and configurable synaptic connections [10]. The proposed MNT comprises a 16:16 fully connected feed-forward SNN structure and supports execution of application subtasks for MNN-based application designs. Fixed connections between the neurons within

[1] **EM**ulating **B**iologically-inspi**R**ed **A**r**C**hitectures in hardwar**E**.

the MNT remove the requirement for storage of connectivity information. The MNTs integrate in a 2D mesh topology NoC communication infrastructure to form an MNN execution architecture, where the overall SNN topology memory requirement is 50 % of the previously reported monolithic NoC-based hardware SNN implementation [5,6]. The paper also proposes a further architectural enhancement, which involves sharing the SNN topology memory (within each MNT) between the SNN structure outputs. The proposed lookup table based memory allocation scheme increases memory utilisation by offering flexible synaptic connectivity suitable for practical SNN application topologies, which are characterised by irregular connectivity patterns [11]. In total, the area requirements of the architecture is reduced by 66 %.

The paper presents micro-architecture details of the proposed MNT and the digital neuron circuit used for system validation. The architectural components are synthesised using 65 nm low-power CMOS technology and silicon area results are presented. The proposed MNT architecture has been validated on a Xilinx Virtex-6 FPGA and resource usage results are reported. The evolvable capability of the proposed MNT and its suitability for executing application subtasks in a MNN execution architectures is demonstrated by successfully evolving the XOR benchmark SNN application and a robotics obstacle avoidance controller using the player-stage robotics simulator and previously reported Genetic Algorithm based hardware SNN evolution platform. (These benchmark SNN applications represent data classification and non-linear control functions.)

The structure of the paper is as follows: Section 2 summarises reported hardware SNN architectures and their suitability as embedded hardware SNNs. Section 3 summarises the reported EMBRACE NoC-based hardware monolithic SNN architecture and highlights the impact of SNN topology memory on the overall device size and the challenge of reducing the SNN topology memory. Section 4 introduces the modular neural network computing paradigm, its benefits and its application for reducing topology memory within the EMBRACE hardware SNN. Section 5 describes the proposed MNT hardware design including neural computing module, packet encoder/decoder, digital neuron design, topology memory and SNN configuration memory. This section also reports ASIC and FPGA synthesis results. Section 6 presents classification and non-linear control benchmark functions implemented on the EMBRACE MNT FPGA prototype. Section 7 concludes the paper.

2 Hardware SNN Architectures

Inspired by biology, researchers aim to implement reconfigurable and highly interconnected arrays of neural network elements in hardware to produce computationally powerful and cognitive signal processing units [12–21]. This section summarises reported hardware and hybrid SNN architectures and their suitability as embedded hardware SNNs.

A hybrid SNN computing platform is reported in [17]. This platform includes a neuron model implemented in hardware and the network model and learning implemented in software. A time multiplexed FPGA embedded processor SNN implementation [20] reports 4.2K neurons and >1.9M synapses. The neuron model and partial SNN elements are implemented on an embedded FPGA processor, where speed–acceleration is the key motivation. The system relies on external memory for spike transfer management. Analogue spiking neuron design approaches can benefit from a compact area implementation due to their ability to model electrical charge flow in the brain [13,18,22,23]. These architectures rely on digital components for a flexible communication infrastructure. FACETS, a configurable wafer-scale mixed-signal neural ASIC system, proposes a hierarchical neural network and the use of analogue floating gate memory for storage of synaptic weights [19,21]. A mixed-signal SNN architecture of 2,400 analogue neurons, implemented using switched capacitor technology and communicating via an asynchronous event-driven bus has been reported in [18]. The chip area is reported to be 3×3 mm using $0.5\,\mu$m CMOS VLSI technology.

Practical SNN systems are characterised by a large numbers of neurons and high interconnectivity through inter-neuron synaptic connections. Each of the SNN execution architectures presented in [14–21] aim for thousands of neurons and millions of synapses, which is essential for a powerful neural computing platform. For large scale hardware implementation of SNNs, the neuron interconnect imposes problems due to high levels of inter-neuron connectivity and often the number of neurons that can be realised in hardware is limited by high fan in/out requirements [4]. Direct neuron-to-neuron interconnection exhibits switching requirements that grow exponentially with the network size. Efficient, low area and low power implementations of neuron interconnect and synaptic junctions are therefore key to scalable hardware SNN implementations [4].

The NoC design paradigm provides a promising solution for the flexible interconnection of large SNNs [7]. The SpiNNaker project [14] aims to develop a massively parallel computer capable of simulating SNNs of various sizes, topology and with programmable neuron models. The SpiNNaker architecture uses ARM-968 processor-based nodes for computation and an off-chip NoC communication infrastructure. Each NoC tile in the SpiNNaker system models 1,000 leaky-integrate-and-fire (LIF) neurons, each having 1,000 synapse inputs. Each SpiNNaker node requires approximately 4 MB of memory for storing synaptic connectivity information [24]. Hence, the SpiNNaker architecture stores the synaptic connection data in off-chip SDRAM. The SNN implementations described above are not targeted at compact embedded hardware applications. A clustered embedded hardware SNN is proposed in [25] along with a process for mapping SNNs to hardware. The variable sized neuron clusters support flexible synaptic connections even from and to within the clusters. The authors have reported the scalable NoC-based EMBRACE [5] hardware SNN architecture, targeting compact embedded hardware applications. The architecture supports large monolithic SNN topologies. In a monolithic SNN, the neuron connectivity is global (i.e. any neuron can connect to any other neuron). The EMBRACE topology memory requirement increases exponentially with network size [10]. Modular neural network techniques investigated in this paper, can enable trade-offs between the topology memory size and synaptic connectivity, leading to scalable, more compact, practical embedded hardware SNN structures.

3 EMBRACE: Hardware SNN Architecture

This section summarises the reported EMBRACE NoC-based hardware monolithic SNN architecture and analyses the impact of SNN topology memory on the overall EMBRACE device size and the challenge of reducing the SNN topology memory.

3.1 EMBRACE: NoC-based Embedded Hardware Monolithic SNN Architecture

The EMBRACE (Fig. 1) [5] architecture incorporates neural circuits within a digital NoC-based packet switching interconnect to realise a scalable hardware monolithic SNN architecture suitable for embedded systems. This architecture offers high synaptic densities while maintaining a small silicon footprint and low power consumption. EMBRACE is a 2D mesh topology array of neural tiles (NT) and NoC routers (R). Each neural tile comprises a single neuron supporting up to 64 input and 64 output synaptic connections. The embedded hardware

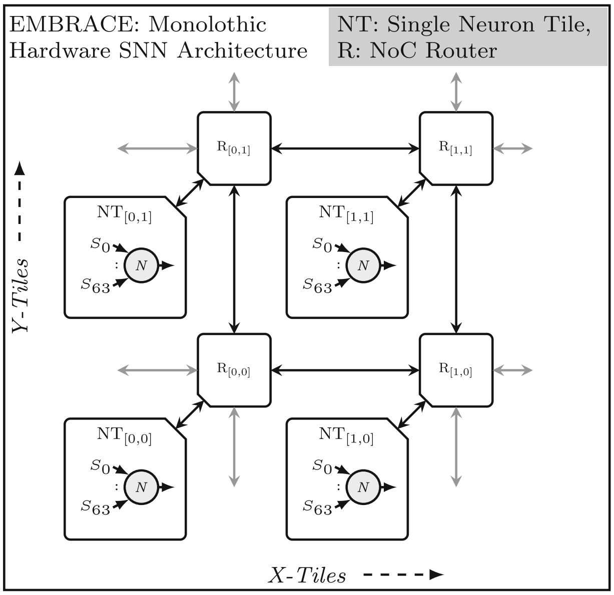


Fig. 1 EMBRACE NoC-based embedded hardware monolithic SNN architecture (the neural tile (NT) comprises a single neuron, packet encoder/decoder, SNN configuration and topology memory)

architecture supports the implementation of monolithic SNNs and provides a benchmark for comparison of the work of this paper. The SNN topology memory defines each inter-neuron synaptic connection. NoC routers are connected in North (N), East (E), South (S) and West (W) directions, forming a Manhattan-style, 2D mesh topology NoC architecture. An application specific SNN is realised on the EMBRACE architecture by programming neuron configuration parameters (SNN synaptic weights and neuron firing threshold potential) and SNN connection topology. Spike communication within the SNN is achieved by routing spike information within spike data packets over the network of routers. The reported architecture (Fig. 1) requires 11 MB of SNN topology memory to support a 64K neuron/4M synapse hardware SNN.

The authors have implemented and reported EMBRACE–FPGA [8], an FPGA prototype implementation of the EMBRACE architecture. The EMBRACE–FPGA prototype has been successfully applied to benchmark SNN control and classifier applications (such as pole balancer, two-input XOR and Wisconsin cancer dataset classifier). EMBRACE-SysC, a SystemC-based clock cycle accurate simulation and performance measurement platform for simulation and analysis of EMBRACE architecture has been reported in [26]. EMBRACE-SysC enables rapid architectural exploration and performance analysis of the EMBRACE architecture.

3.2 EMBRACE Hardware Resource Analysis

The transistor count and chip area has been estimated for the EMBRACE architecture to understand the practicality of realising the EMBRACE architecture in silicon. The silicon area estimation technique takes into account the transistor count for storage and control logic of digital components (based on standard SRAM cell design) and actual silicon area for neurons [23].

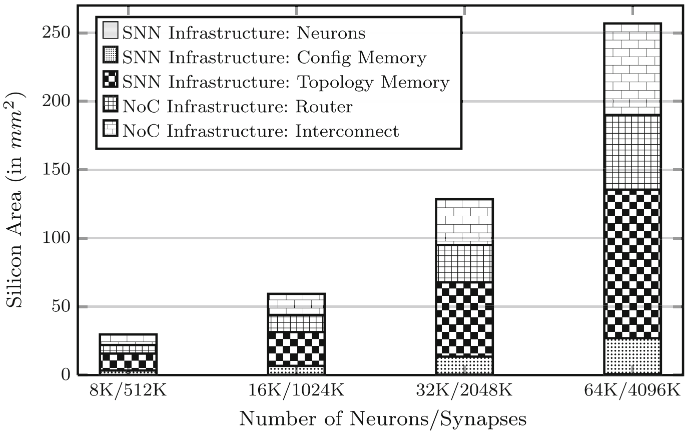


Fig. 2 Silicon area proportion (for 32 nm CMOS technology) for the EMBRACE hardware monolithic SNN architecture

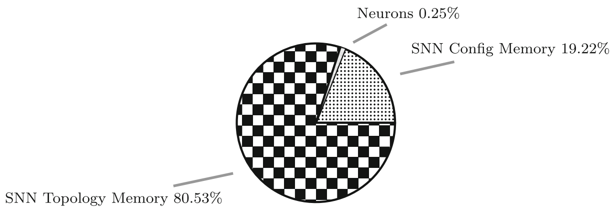


Fig. 3 Silicon area proportion for the SNN infrastructure entities within EMBRACE hardware monolithic SNN architecture

Figure 2 presents the silicon area proportions (in mm^2) by scaling the EMBRACE hardware monolithic SNN architecture in 32 nm CMOS VLSI technology. The *x*-axis indicates the number of neurons and synapses (neuron/synapse) and the stacked columns in the histogram denote the silicon area for NoC and SNN architectural entities described below:

- *NoC infrastructure:* The EMBRACE NoC infrastructure comprises NoC routers, packet buffers, NoC interconnect channels and associated control circuits. The analytical estimates indicate that the complete NoC infrastructure will occupies 47.27 % of the total chip area.
- *SNN infrastructure:* The SNN support infrastructure is made-up of SNN configuration memory (for storing synaptic weights of 5 bits each and threshold values of 16 bits each) and the SNN topology memory (for storing synaptic connectivity information) [6]. The SNN infrastructure also contains the neural elements (which includes synapses, synaptic weight summing and membrane potential thresholding circuits) [13,22,23]. Due to its compact implementation, the silicon area for the neural elements is negligible compared to the rest of the SNN support infrastructure, which occupies 52.74 % of the total chip area (Figure 3 further enumerates the silicon area of the SNN components within the EMBRACE architecture).

Architectural techniques to reduce SNN topology memory are crucial for their compact hardware implementations. This paper presents a novel MNT architecture that reduces the SNN topology memory area by 50 %.

4 Modular Neural Networks

This section introduces the modular neural network (MNN) computing paradigm and execution architecture. The section highlights the benefits and application of the MNN design approach to reduce topology memory within the reported EMBRACE hardware monolithic SNN architecture.

4.1 Biological Motivations for the MNN Computing Paradigm

The biological brain is composed of several anatomically and functionally discrete areas [27]. Various brain areas and neuron groups are dedicated to various sensory and motor tasks. For example, the visual cortex processes different aspects of the visual information (such as form, colour and motion) in separate anatomically distinct regions. These different regions exchange information but remain functionally discrete [28,29]. Damage or deterioration of part of the visual cortex can result in a partial loss of colour identification, pattern or motion detection, etc, without considerably affecting other senses. Inspired by this modular organisation in brain, the MNN design strategy of partitioning application tasks into a number of subtasks has been developed [30–32].

4.2 Modular Neural Network Computing Paradigm

The MNN computing paradigm is primarily based on the *divide-and-conquer* strategy. Figure 4 illustrates the MNN design methodology, which primarily consists of *task decomposition* i.e. breaking down a high level application into smaller, less complex, manageable subtasks. These small subtasks are then solved by individual and distinct neural modules. The intermediate outputs from these neural modules are combined to solve the high level task or the whole application [1,9]. Various task decomposition algorithms have been proposed for the MNN design strategy. These algorithms are based on different techniques such as output vector partitioning, class relationships, neuro-evolutionary approach and co-evolutionary methods [33–37]. Similarly, genetic algorithm based technique to find subnetworks from large sized complex neural networks has been proposed in [38]. Subtasks/subnetworks obtained after task decomposition are executed as neural modules in MNN computing architecture.

The MNN approach offers structured implementation, functional partition, functional mapping and re-mapping, competitive and co-operative mode of operation and fault tolerance [31]. The *Subsumption Architecture*, a widely influential computing paradigm for reactive, behaviour-based autonomous robotic control applications has been developed based on the MNN design concepts [39].

4.3 Modular Neural Network Execution Architectures

In the MNN design methodology, the task decomposition or partitioning of the overall application leads to two types of subtasks; namely the application subtasks (discrete functional subtasks) and the integration tasks. The application subtasks operate on individual and distinct system inputs to provide intermediate outputs. The integration subtasks integrate the intermediate outputs from the application subtasks to generate the overall system output. Both the application and integration subtasks are similar in terms of input/output interface and computation requirements. Figure 5 illustrates a typical MNN execution architecture comprising individual neural modules interconnected by a global communication

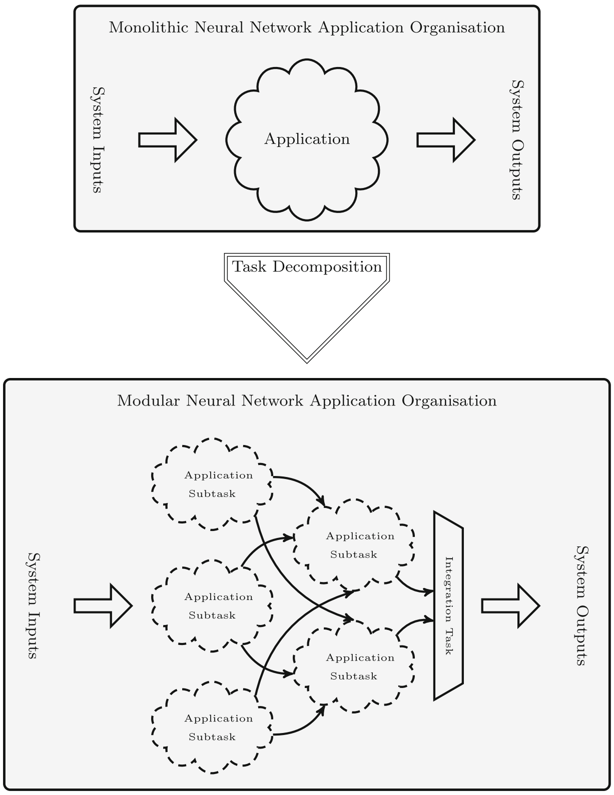


Fig. 4 Modular neural network design methodology

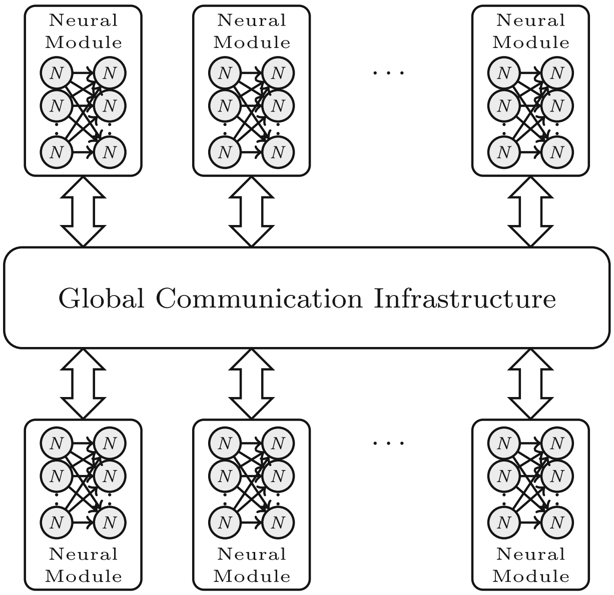


Fig. 5 Modular neural network execution architecture

infrastructure. Based on the definition of modularity in neural networks [1,9], the individual neural modules in the MNN execution architecture should:

- Include a group of neurons interconnected using an internal communication infrastructure
- Support multiple inputs and/or multiple outputs
- Include an internal communication infrastructure that is isolated from the other modular neural elements and the global (external inter-module) communication infrastructure

Since the synaptic communication between neurons in a neural module is isolated from the rest of the architecture, the requirement for the storage of associated connectivity information is eliminated. The overall MNN architecture has a considerably lower synaptic connectivity storage requirement compared to monolithic neural architectures. The SNN topologies for applications that are inherently non-modularizable exhibit uniform connectivity patterns. The neural modules in the MNN execution architecture can be cascaded using global communication infrastructure to realise such large monolithic SNN structures. However, the resource utilisation of the MNN execution architecture will be higher as compared to hardware SNN arrays supporting uniform synaptic connectivity.

5 Modular Neural Tile Architecture

This paper presents a novel Modular Neural Tile (MNT) architecture, its digital prototype and evolvable capabilities. The proposed MNT forms the basic neural module for the MNN execution architecture (proposed NoC-based modular hardware SNN execution architecture).

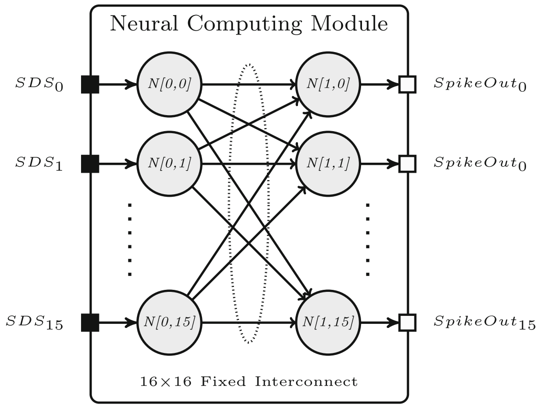
Architectural techniques for reducing the SNN topology and configuration memory are vital for compact silicon implementation of hardware SNN architectures suitable for embedded computing. This section presents the MNT architecture comprising a 16:16 fully connected feed-forward topology SNN structure as the neural computing module (NCM) and the lookup table-based SNN topology memory sharing scheme. The micro-architecture of an efficient digital neuron model is also described. Detailed ASIC and FPGA synthesis results are presented. Silicon area requirement of the NoC architecture employing the proposed MNT is compared with the previously reported EMBRACE hardware monolithic SNN architecture.

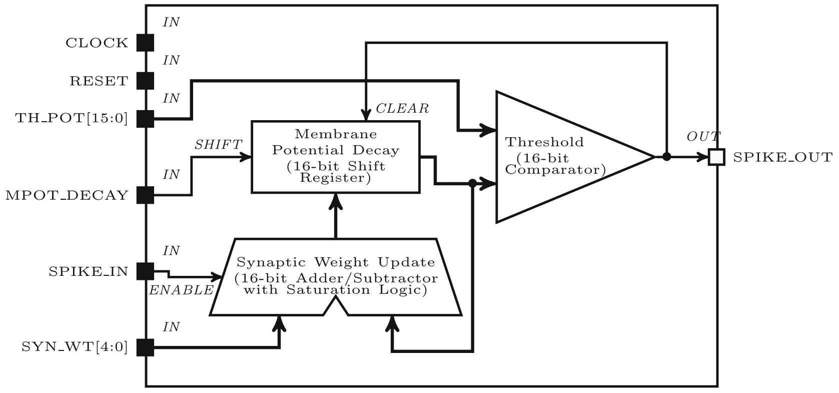
5.1 Neural Computing Module

The Neural Computing Module (NCM) forms the basic computing entity within the MNN execution architecture. The NCM micro-architecture design is primarily influenced by the following:

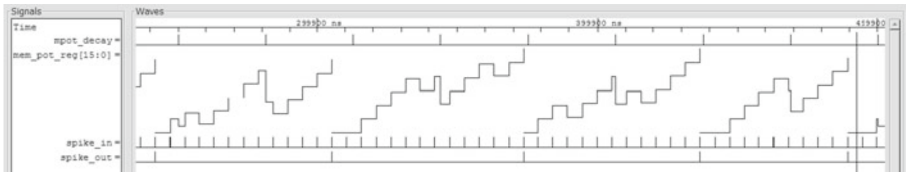
- *VLSI technology limitations:* Fixed interconnections between neurons within the SNN structure removes the need for synaptic connectivity information storage within the MNT. The size of the fully-connected hardware SNN structure is limited by the permitted fan-out of individual neuron circuits. Also, metal layer routing limitations are imposed due to interconnect crossbar capacitance and crosstalk.
- *MNN subtask granularity:* The NCM should support sufficient computing power for MNN application subtasks and integration subtasks. Large NCM designs, can accommodate a variety of MNN application subtasks, but would lead to unused neurons in the case of fine granularity MNN application subtasks. NCM designs with smaller SNN structures can be cascaded based on the computing requirements of the particular MNN application and integration subtasks.

Fig. 6 Two layered 16:16 fully connected SNN structure as the neural computing module within the proposed MNT




CLOCK
RESET
TH_POT[15:0]
MPOT_DECAY
SPIKE_IN
SYN_WT[4:0]
IN
SHIFT
ENABLE
CLEAR
Membrane Potential Decay (16-bit Shift Register)
Synaptic Weight Update (16-bit Adder/Subtractor with Saturation Logic)
Threshold (16-bit Comparator)
OUT
SPIKE_OUT

(a) Micro-Architecture



(b) Simulation Waveform

Fig. 7 Digital neuron model

Considering the above, a two-layered 16:16 fully connected feed-forward SNN structure (Fig. 6) has been proposed as the NCM inside each MNT [10]. The input layer ($N[0, n]$) and output layer ($N[1, n]$) of the NCM comprises 16 LIF neurons. The neurons support a *Single Dynamic Synapse* (SDS_n) approach (Fig. 7a), where the synaptic weight is supplied along with spike input. This shared synapse approach removes the need for internal multiplexers for selection of synaptic weight and results in compact hardware implementation [40]. Each of the 16 input layer neurons connects directly to each of the 16 output layer neurons to form a fully connected feed-forward SNN structure. Each output layer neuron has 16 input synapses, which individually receive spikes from the corresponding input layer neurons. The NCM has 16 spike outputs ($SpikeOut_n$) each corresponding to the 16 output layer neurons.

The NCM in a MNN execution architecture should be capable of solving application subtasks and integration tasks. The proposed modular NCM made-up of 32 LIF neurons, supports multiple synaptic inputs and provides 16 spike outputs that can connect to multiple synaptic inputs in the architecture. This paper demonstrates the suitability of the proposed NCM by evolving SNN benchmark functions.

5.1.1 Digital Neuron Model

This section describes the multiplier-less, compact hardware digital neuron design.

Based on biological plausibility, computational power and implementation complexity, various mathematical models representing the spiking behaviour of biological neurons have been proposed [2,3]. Amongst them the LIF model is a popular choice for hardware SNN architectures due to its simplicity. The proposed EMBRACE research project ultimately aims to develop a mixed-signal VLSI architecture with compact, low power, high-resolution CMOS-compatible analogue synapse and LIF neuron cells [13,23] to achieve very high synaptic density. The LIF neuron model has multiple synaptic inputs, a single spike output and maintains an internal *membrane potential*, which constantly decays (to its resting value) based on a constant *leakage coefficient*. The synaptic weight value associated with the input synapse is summed with the current membrane potential value on receipt of an input spike. The exhibitory/inhibitory synaptic weights increase/decrease the membrane potential by the weight value. The neuron *fires* (emits a spike) when the membrane potential reaches the threshold potential value. Firing of the LIF neuron causes the membrane potential to reset to the resting value.

For validation of the proposed MNT architecture on FPGA, a digital neuron circuit exhibiting LIF behaviour with sufficient resolution has been developed (Fig. 7a). The digital neuron circuit uses a 16-bit shift register for storage of the *Membrane Potential* value. Stepwise exponential decay of the membrane potential value is achieved by periodic right shift (divide by two) controlled by the MPOT_DECAY input pulse. The desired decay rate or *Leakage Coefficient* can be controlled by programming the membrane potential decay strobe generator (Fig. 7b). The membrane potential value is updated by the input *Synaptic Weight* (SYN_WT) value using a 16-bit adder/subtractor enabled by the incoming spike pulse (SPIKE_IN). The overflow/underflow bit of the adder/subtractor is used to determine the upper/lower limit of the membrane potential value, and activate the saturation logic. A 16-bit comparator generates the spike output (SPIKE_OUT) if the internal *membrane potential* >*threshold* (TH_POT) input. The generated spike pulse (SPIKE_OUT) is also used to clear the 16-bit shift register, bringing the membrane potential to the resting value.

Figure 7b shows a simulation waveform for the digital neuron circuit including membrane potential decay due to MPOT_DECAY pulses, membrane potential variation due to input spikes, and output spike generation. The synthesizable digital neuron circuit provides the required resolution for practical SNN applications. The neuron occupies only 12 slices in the Xilinx Virtex-6 FPGA and occupies 608.4 μm^2 in 65 nm CMOS VLSI technology. This compact neuron model contributes to the goal of portable embedded hardware SNNs.

5.1.2 Neural Computing Module Architecture

Fixed interconnection between neurons within the neural computing module removes the need to store synaptic connectivity information. Figure 8 shows the fixed connection micro-architecture of the NCM. The interconnect architecture transfers the generated spikes from the

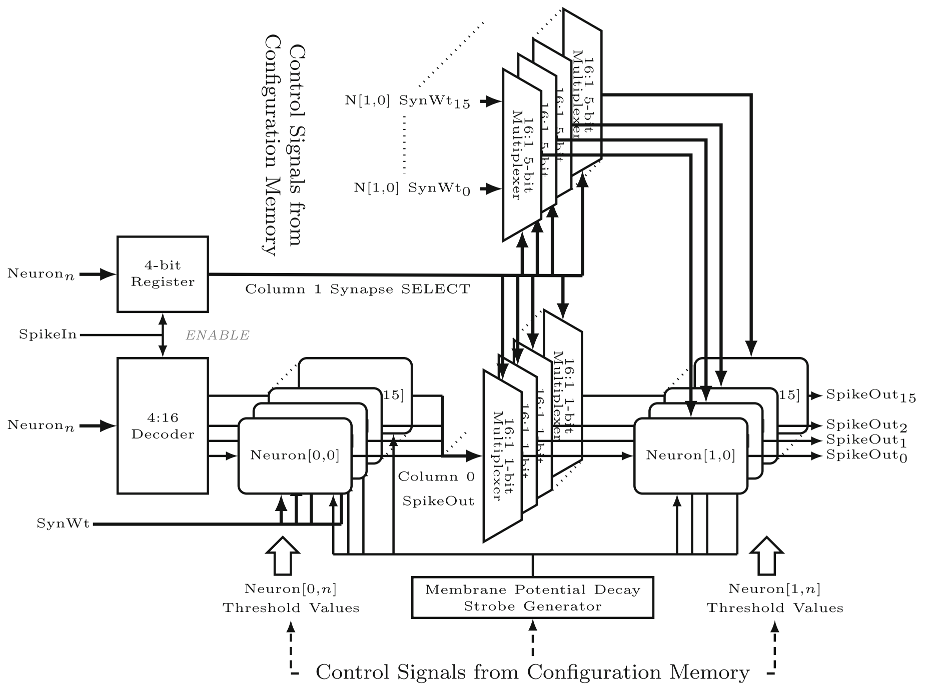


Fig. 8 Neural computing module micro-architecture

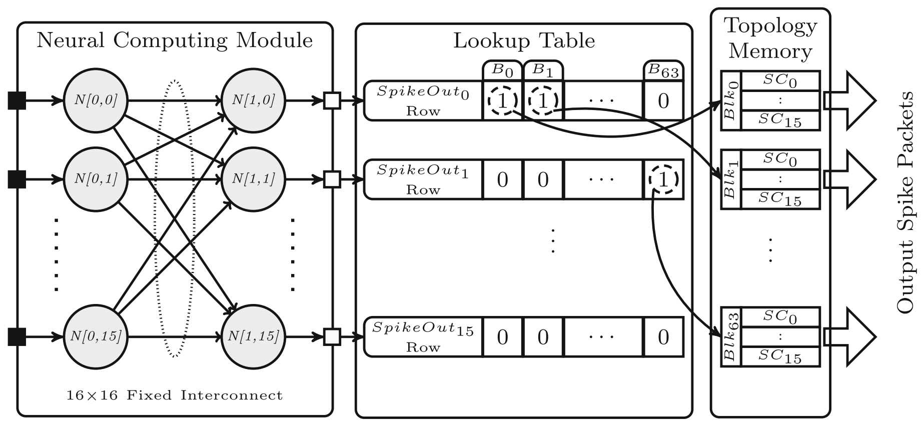
input layer neurons to output layer neurons. The interconnect architecture enables selection of synaptic weights from configuration memory for the output layer neurons while applying spikes.

The input interface to the NCM consists of neuron number (Neuron$_n$), synaptic weight (SynWt) and input spike pulse (SpikeIn). The NCM layers operate in a pipelined fashion, where the input spike is applied to the selected input layer neuron during the first clock cycle and any generated spike is transferred to output layer neurons in the next clock cycle. The input neuron number (Neuron$_n$) is stored in a 4-bit register for use in the next clock cycle. The input synaptic weight is directly connected to all of the input layer neurons. The 4:16 decoder enables the selected input neuron number. The decoder is enabled by the input spike pulse, which causes the selected neuron to activate and update its internal membrane potential.

According to the LIF neuron behaviour [3], neurons can generate a spike only on the occurrence of an input spike. Hence, the stored neuron number in nth clock cycle is used for transfer of spike and selection of synaptic weight for output layer neurons in (n+1)th clock cycle within the NCM. The NCM output interface comprises spike outputs (SpikeOut$_{0:15}$) from all output layer neurons. Spikes pulses generated by the NCM are further processed within the MNT to generate output spike packets for synaptic connections outside the MNT.

5.2 Topology Memory Sharing and Spike Packet Output Flow Control

Spike communication between MNTs is achieved by routing spike information within spike data packets over the network of routers. Within each MNT a lookup table based SNN topology memory allocation technique is implemented, which enables variable synaptic



(a) The Lookup Table-based Shared Topology Memory Organisation

$X_{Address}$ NoC Tile X Address (4-bit)	$Y_{Address}$ NoC Tile Y Address (4-bit)	$Neuron_n$ MNT Input Neuron Number (4-bit)	$SynWt$ Synaptic Weight (5-bit)

(b) Synaptic Connection (SC) Information Entry Stored in the Topology Memory

Fig. 9 The MNT spike packet output flow control

connection densities for efficient topology memory resource usage. This section describes spike packet generation based on the proposed lookup table-based topology memory sharing scheme, which offers a flexible number of synaptic connections for NCM outputs.

Figure 9 illustrates the lookup table-based shared topology memory organisation. The synaptic connection information for the NCM outputs is stored in this topology memory. Fields include destination MNT address (*[X,Y]* mesh topology NoC tile address), destination neuron ($Neuron_n$) and synaptic weight ($SynWt$) (see Fig. 9b). The topology memory is partitioned into 64 blocks (B_0 to B_{63}), where each block is made-up of 16 synaptic connection information entries ($B_x SC_0$ to $B_x SC_{15}$), giving a total of 1,024 neuron/synapse destinations. The lookup table maintains the topology memory block allocation information in designated rows for each NCM output. Each bit in the 64-bit lookup table row allocates the corresponding topology memory block to the NCM output. For example, bit number B_x of the row number *N[1,0]* allocates topology memory block number X to the NCM output *N[1,0]*. For the row number *N[1,0]*, asserting the bit value B_x, allocates the topology memory block X to NCM output *N[1,0]*; deasserting B_x disassociates the topology memory block X from the NCM output *N[1,0]*. The packet encoder generates spike packets for NCM outputs based on the allocated topology memory blocks.

The process of mapping the MNN application topology onto the proposed MNT involves populating the lookup table and MNT topology memory entries, such that the correct synaptic connections are established between the neural computing modules in the architecture. If the required number of synaptic connections for a particular NCM cannot be accommodated in the available topology memory, additional MNTs can be used as spike repeaters. The NCM can be configured as spike repeater by configuring synaptic weights and threshold to generate a spike for each input spike.

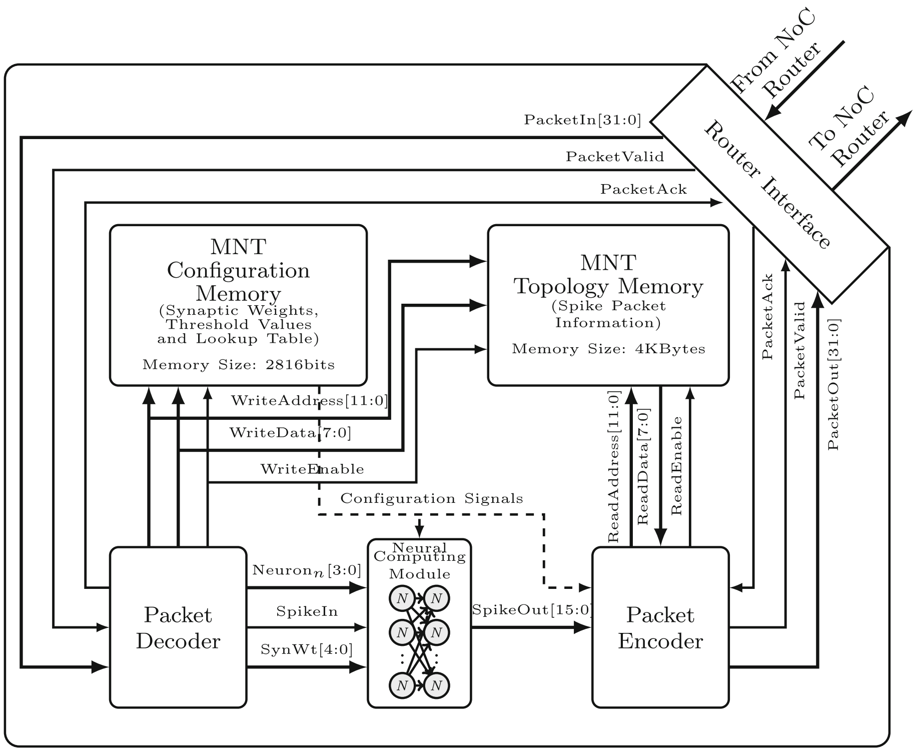


Fig. 10 Modular neural tile internal architecture

5.3 Modular Neural Tile Hardware Design

The MNT (Fig. 10) comprises a NCM, configuration memory (for storing neuron synaptic weights and threshold values), topology memory (for storing synaptic connectivity information) and packet decoder and encoder. This section describes the interfaces and functionality of the hardware entities within the proposed MNT.

5.3.1 Modular Neural Tile Internal Architecture

The MNT connects to the NoC router through the router interface, which comprises 32-bit PacketIn/PacketOut, valid (PacketValid) and acknowledgment (PacketAck) handshake signals. The MNT supports the following packet types (Fig. 11):

- *Configuration packet:* used for configuration of neurons (synaptic weights and threshold values), the lookup table and output synaptic connectivity. The configuration packet consists of destination tile address (XY NoC tile address), configuration memory address (13-bit) and data (8-bit).
- *Spike packet:* used for transferring spikes from source MNT to destination MNT. It consists of the destination tile address 2-D array format (XY NoC tile address), neuron number ($Neuron_n$) and synaptic weight. Inclusion of synaptic weight in the spike packet reduces the overall storage requirement of the architecture, and removes the necessity

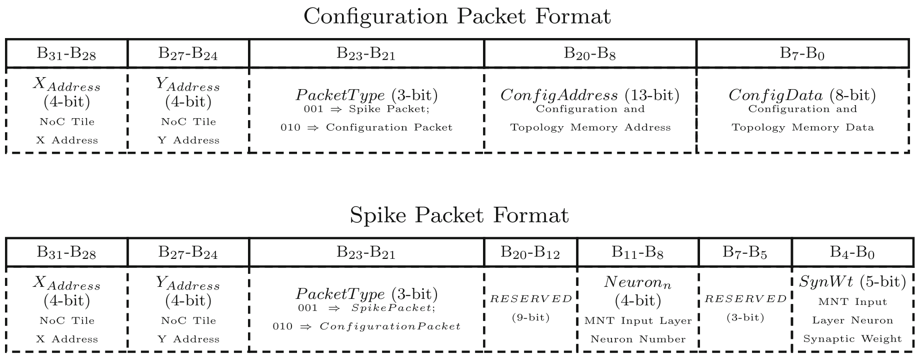

Configuration Packet Format
B31-B28
B27-B24
B23-B21
B20-B8
B7-B0
$X_{Address}$ (4-bit) NoC Tile X Address
$Y_{Address}$ (4-bit) NoC Tile Y Address
$PacketType$ (3-bit) 001 ⇒ Spike Packet; 010 ⇒ Configuration Packet
$ConfigAddress$ (13-bit) Configuration and Topology Memory Address
$ConfigData$ (8-bit) Configuration and Topology Memory Data
Spike Packet Format
B31-B28
B27-B24
B23-B21
B20-B12
B11-B8
B7-B5
B4-B0
$X_{Address}$ (4-bit) NoC Tile X Address
$Y_{Address}$ (4-bit) NoC Tile Y Address
$PacketType$ (3-bit) 001 ⇒ $SpikePacket$; 010 ⇒ $ConfigurationPacket$
$RESERVED$ (9-bit)
$Neuron_n$ (4-bit) MNT Input Layer Neuron Number
$RESERVED$ (3-bit)
$SynWt$ (5-bit) MNT Input Layer Neuron Synaptic Weight

Fig. 11 MNT configuration and spike packet format

Table 1 Modular neural tile synthesis results (clock frequency: 200 Mhz)

RTL entity	ASIC synthesis (65 nm low-power CMOS)		FPGA synthesis (Xilinx Virtex-6 FPGA)	
	Area (in μm^2)	Percentage (%)	Slices	BRAMs
Modular neural tile	110,806.63	100.0	1,204	0
Neural computing module	20,505.24	18.7	458	0
Digital neuron circuit	608.40	0.5	12	0
Packet decoder	408.71	0.4	7	0
Packet encoder	2,368.60	2.1	66	0
NCM configuration memory	31,444.47	28.4	662	0
NCM topology memory	55,800.00	50.4	0	1

for synaptic weight selection multiplexers for the NCM inputs, resulting in compact hardware implementation [40].

The packet decoder interfaces with the NCM, configuration and topology memory by receiving and decoding the input packets. The input packet type (spike or configuration packet) is decoded from the *Packet Type* bits (B_{23}–B_{21}). For a configuration packet, the configuration memory address and data are retrieved from the packet and written to configuration or topology memory. For a spike packet, neuron number ($Neuron_n$) and synaptic weight ($SynWt$) are retrieved from the packet and forwarded to the NCM along with an internally generated spike pulse. The configuration memory supplies synaptic weight and threshold values to the NCM and lookup table bits to the packet encoder through dedicated control signals. The topology memory is a dual-ported RAM module as it interfaces with the packet decoder for memory write operation and with the packet encoder for memory read operation. The packet encoder generates individual spike packets based on spike inputs from the NCM, synaptic connectivity information in the topology memory and memory block allocation information from the lookup table. Refer to Section 5.2 for the detailed output spike packet flow control.

Table 1 shows the detailed hardware synthesis results for the proposed MNT. ASIC synthesis has been performed using synopsys design compiler 2009-sp5 and TSMC 65 nm low-power CMOS libraries. FPGA synthesis has been carried out using Xilinx XST 13.2. Since the configuration memory supplies all the MNT control signals (2,816 bits, synaptic weights, threshold values and lookup table) to the NCM and packet encoder, the silicon footprint is

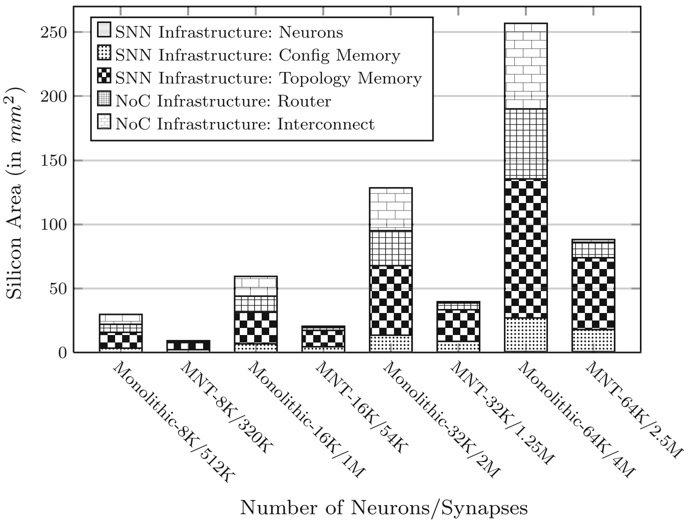


Fig. 12 Silicon area estimate (for 32 nm CMOS technology) comparison for the EMBRACE monolithic and the proposed MNT hardware SNN architecture

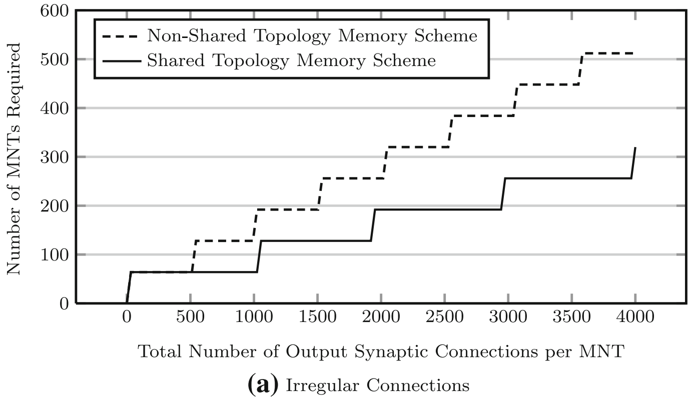
considerably higher compared to memory size. Due to massive interconnectivity in neural architectures, the majority of the area is occupied by topology memory (50 % in the proposed MNT). The topology memory is realized using STMicroelectronics low-power CMOS dual-ported memory IP in ASIC implementation.

Figure 12 compares the silicon area of various sized SNNs MNT NoC architecture with that of the reported EMBRACE hardware monolithic SNN architecture. The reduced number of NoC routers, resulting from the 16:16 fully connected SNN (NCM), decreases the area occupied by the NoC infrastructure in the proposed MNT NoC architecture by 89 % as compared to the reported EMBRACE SNN architecture. The fixed interconnection within the NCM removes the need for storing the output synaptic connectivity information for the input layer neurons. The regularly structured interconnect requires much less silicon area than the SRAM-based synaptic connectivity storage and the associated control circuitry. Consequently, the topology memory for the proposed MNT NoC architecture is reduced by 54.05 %. The size of the entire MNT NoC-based hardware SNN is approximately 33 % of that of the previously reported EMBRACE chip area estimation.

5.3.2 Hardware Resource Requirements of Practical SNN Topologies

Practical SNN application topologies exhibit a variety of connectivity patterns. Flexible sharing of the topology memory within the MNT addresses diverse connectivity requirements of the practical modular SNN application topologies. This section presents and compares hardware resource requirements for the proposed MNT architecture with shared and non-shared topology memory schemes for SNN application topologies with irregular and random connectivity patterns [11]. Additional MNTs are used for relaying spikes, if the synaptic connectivity requirement of the NCM cannot be accommodated in the topology memory within the MNT.

A large modular neural network application, made-up of 64 individual NCMs, has been mapped to the proposed MNT-based NoC architecture. The application implementations



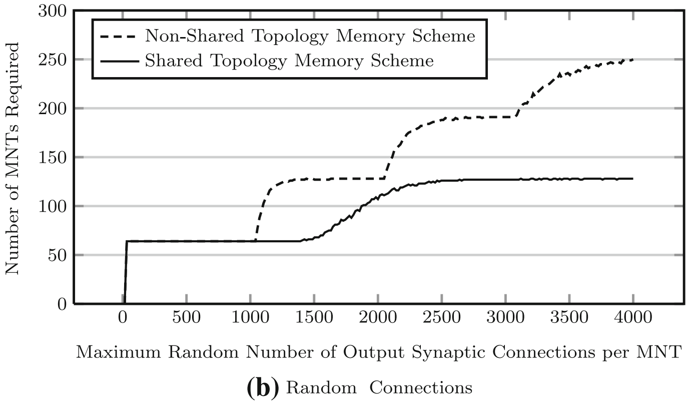


Fig. 13 NoC tile requirements for non-shared and shared topology memory schemes for the **a** irregularly and **b** randomly connected example modular topology

using non-shared and shared topology memory configurations have been compared. The non-shared topology memory scheme uses a fixed allocation of four topology memory blocks to each NCM output. The NCMs in the shared topology format are configured such that eight outputs from each NCM remain inactive (by configuring zero synaptic connections). The number of required MNTs and the size of the NoC are calculated for various synaptic connection densities in the remaining eight active NCM outputs. Figure 13a compares the number of MNTs required using non-shared and shared topology memory approach, executing the MNN application topology with irregular synaptic connectivity pattern. The topology memory in the MNT can hold 1K (i.e. 1,024) synaptic connection entries. When the synaptic connectivity requirement of each NCM increases by 1K steps, additional set of MNTs are used for relaying spike packets. This can be seen in the step wise ascending graph in Fig. 13a.

The SNN topologies evolved using Genetic Algorithm (GA) based search methods often exhibit random connectivity patterns [11]. The application topology described above has been configured for a random number of output synaptic connections from each of the 64 individual NCMs. This MNN application representing a random synaptic connectivity pattern has been mapped to the proposed MNT NoC architecture and tested under non-shared and shared topology memory configuration. Figure 13b illustrates the MNT requirement for the proposed MNT NoC architecture under non-shared and shared topology memory scheme, executing the application topology with a random synaptic connectivity pattern.

The proposed shared topology memory architecture facilitates the allocation of topology memory blocks to the NCM outputs based on the synaptic connectivity requirement. The lookup table based shared topology memory architecture offers a flexible number of synaptic connections from the NCM outputs resulting in efficient usage of each MNT. Figure 13 illustrates that, the shared topology memory scheme requires a smaller number of MNTs for MNN application topologies with irregular and random synaptic connectivity patterns (observed in practical SNN application topologies). This enables implementation of larger application topologies within the given architectural configuration.

6 Modular Neural Tile Applications

This section presents classification and controller benchmark functions implemented on the EMBRACE modular neural tile FPGA prototype. The benchmark XOR function (a basic data classifier) and robotics controller (closed loop non-linear control system) have been used to test the evolvable capability of the proposed MNT. Accuracy (fitness) of the SNN configuration and training time results (in terms of GA generation count) are presented for successfully evolved XOR and robotics controller application on the proposed MNT prototype. This demonstrates the capability of the proposed MNT to evolve small-sized subtasks within a larger MNN applications. The intrinsic evolution setup comprising the proposed MNT prototype on FPGA and GA-based SNN configuration platform running on host computer [8], highlight the hardware validation aspects of the modular SNN architecture.

6.1 Classification Subtasks

The XOR function is a basic benchmark data classification problem for neural networks and is a sub-problem of more complicated classifiers [41]. A two-input XOR function is

Table 2 Two-input XOR function fitness score assignment

Number of correct outputs	0	1	2	3	4
Fitness score assignment value	0	1	4	9	16

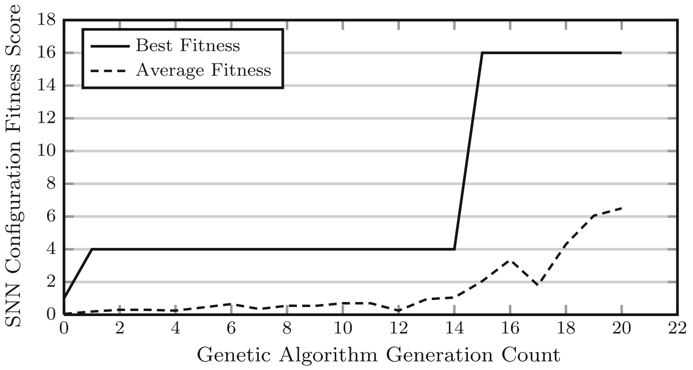


Fig. 14 XOR benchmark SNN application on the proposed MNT

implemented on the proposed MNT. The XOR function uses two spike rate encoders feeding distinct spike rates for logic '0' and logic '1' and an output spike rate decoder. The GA-based SNN evolution and configuration platform configures the SNN configuration comprising synaptic weights and threshold potential for all the neurons in the SNN, check the accuracy (fitness) of the configuration and evolves the desired functionality by searching the correct configuration. The evolved XOR function on the proposed MNT outputs a high spike rate when the two inputs differ in spike rate, and a low spike rate if the inputs are equal. Table 2 illustrates the fitness score assignment used for the two input binary XOR function. The three neuron XOR SNN partially uses the MNT. Figure 14 illustrates the average and best fitness of the evolved SNN XOR function on the proposed MNT.

6.2 Non-Linear Control Subtasks

The robotics controller application is a classical example of a non-linear closed loop control system. Neural network systems are widely used as controllers for practical robotics applications. A robotics obstacle avoidance controller, using the player-stage robotics simulator [42] and the previously reported Genetic Algorithm (GA)-based hardware SNN evolution and implementation platform (Fig. 15). The simulated robot is equipped with 16 sonar sensors. The average values of front, rear and side sonars are used as inputs to the SNN. The sonar values are converted to spike rates by the host application and are passed to spike rate encoders, which continuously generate and feed spikes to the NCM within the MNT. Spikes from two NCN outputs are monitored by spike rate decoders and are converted to analogue values as robot acceleration and turning angle inputs to the simulator.

Fitness criteria for the robotic obstacle avoidance controller application is defined as travelling finite distance and avoiding obstacles for $\geq$120 s, and the fitness of the SNN configuration is calculated as:

$$F = \alpha T + \beta D + \gamma S \qquad (1)$$

where, F fitness of the individual, T robot travel time (in s), D robot travel distance (in cm) and S robot travel speed (in cm/s).

Given:

Number of crashes = 0
SNN outputs are within the operating range
The fitness evaluation constants α, β and γ are chosen to prioritise the robot motion behaviour.

Evaluation of the individual configurations has been accomplished with the robot roaming within the simulated environment for $300 \geq t > 120$. On timeout, or if the robot has crashed, the GA-based evolution and configuration platform processes the recorded robot behaviour and assigns a fitness score to the individual SNN configuration. Fitness scores are then used by the GA to determine the probability of an individual configuration progressing to later evolved generations. Figure 16 illustrates the average and best fitness of the evolved robotic obstacle avoidance controller application on the proposed MNT.

Successful evolution of classification and non-linear control functions with constant and time varying input patterns on the proposed MNT demonstrates its suitability for a variety of MNN application designs.

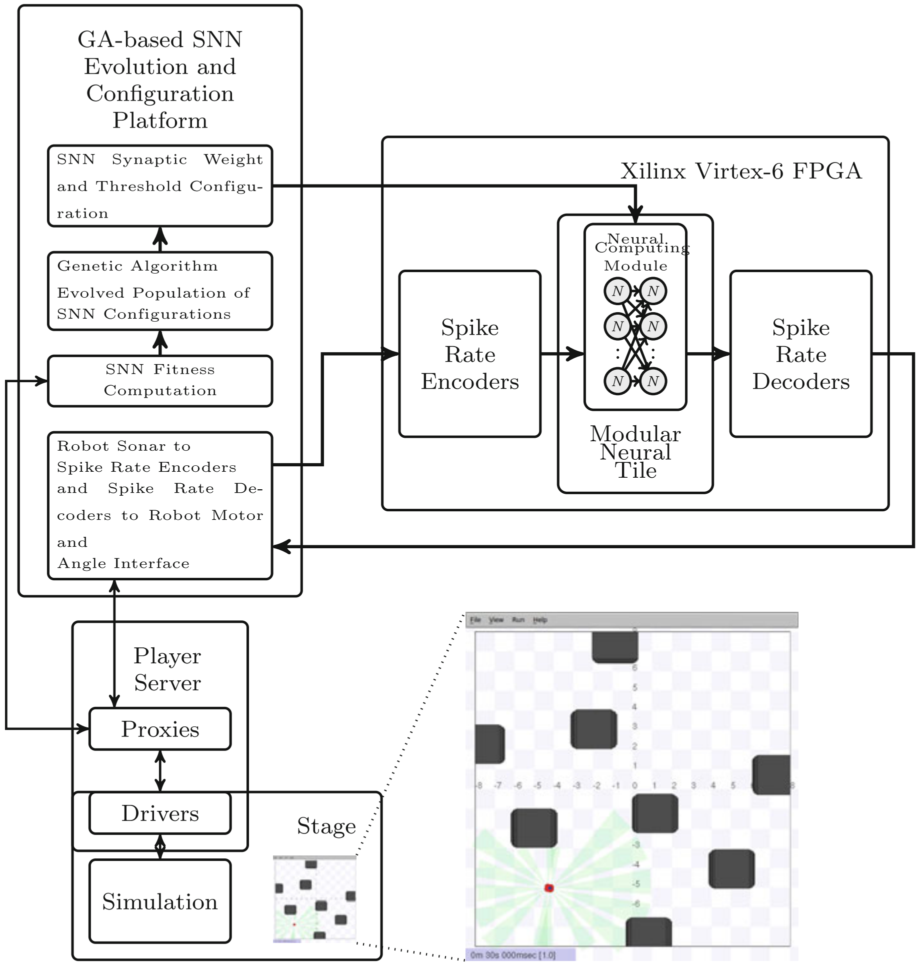

GA-based SNN Evolution and Configuration Platform
SNN Synaptic Weight and Threshold Configuration
Genetic Algorithm Evolved Population of SNN Configurations
SNN Fitness Computation
Robot Sonar to Spike Rate Encoders and Spike Rate Decoders to Robot Motor and Angle Interface
Xilinx Virtex-6 FPGA
Spike Rate Encoders
Neural Computing Module
N
Modular Neural Tile
Spike Rate Decoders
Player Server
Proxies
Drivers
Stage
Simulation

Fig. 15 Robotic obstacle avoidance controller setup

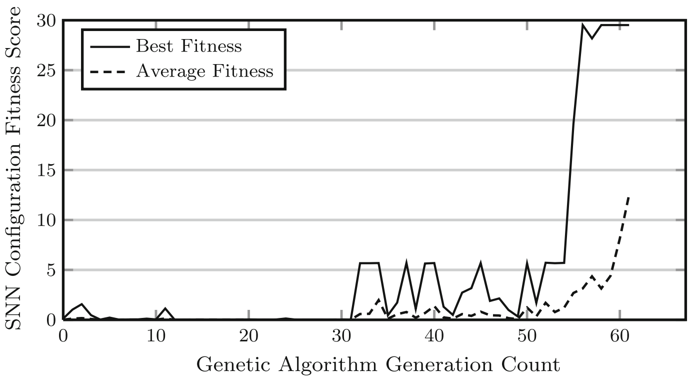


Fig. 16 Robotic obstacle avoidance controller benchmark SNN application on the proposed MNT (fitness evaluation constants: $\alpha = 2$, $\beta = 1$ and $\gamma = 0.5$)

7 Conclusions

Neural architectures are typically characterised by thousands of neurons and millions of synapses which are essential for a powerful neural computing platform. Storage of the large synaptic connectivity information in packet switched network on chip (NoC)-based hardware Spiking neural network (SNN) architectures translates to large distributed on-chip memory and poses a serious challenge for compact hardware implementations. Discrete synaptic connectivity observed in MNN execution architectures help in reducing the storage requirement of NoC-based hardware neural architectures.

This paper presented a novel MNT architecture comprising a 16:16 fully connected feed-forward topology SNN structure as NCM. The topology memory of the architecture is 50 % of the previously reported NoC-based hardware monolithic SNN implementation. Furthermore, a lookup table-based topology memory sharing scheme is presented that provides a flexible number of synaptic connections from NCM outputs. The proposed shared topology memory scheme requires less number of MNTs for practical MNN application topologies with irregular and random synaptic connectivity patterns. Overall the area requirement of the architecture is reduced by an average 66 % for practical SNN application topologies. This facilitates accommodation of larger application topologies in the given architectural configuration.

The paper presented micro-architecture details of the proposed MNT and the digital neuron circuit used for system validation. The architectural components are synthesised using 65 nm low-power CMOS technology and silicon area results are presented. The proposed MNT architecture has been validated on Xilinx Virtex-6 FPGA and resource utilisation is presented. The evolvable capability of the proposed MNT, and its suitability for executing application subtasks, in a modular hardware SNN architecture is demonstrated by successfully evolving the XOR benchmark SNN function and a robotics obstacle avoidance controller, using the player-stage robotics simulator and the previously reported Genetic Algorithm (GA)-based hardware SNN evolution and configuration platform. Successful evolution of classification and non-linear control functions with constant and time varying input patterns on the proposed MNT demonstrates its suitability for variety of MNN application designs.

The architectural enhancements proposed and validated in this paper helps to achieve a compact neural modular hardware implementation and demonstrate the ability of the proposed MNT to successfully evolve benchmark SNN functions. This work contributes to the development of the EMBRACE NoC-based embedded hardware SNN device [12].

Acknowledgments This research is supported by International Centre for Graduate Education in Micro and Nano-Engineering (ICGEE), Irish Research Council for Science, Engineering and Technology (IRCSET), Science Foundation, Ireland (Grant No. 07/SRC/I1169) and Xilinx University Programme.

References

1. Haykin SS (1999) Neural networks: a comprehensive foundation, vol 13. Prentice Hall, New Jersey
2. Maass W (1997) Networks of spiking neurons: the third generation of neural network models. Neural Netw 10(9):1659–1671
3. Wulfram G, Werner MK (2002) Spiking neuron models. Cambridge University Press, Cambridge
4. Maguire LP, McGinnity TM, Glackin B, Ghani A, Belatreche A, Harkin J (2007) Challenges for large-scale implementations of spiking neural networks on FPGAs. Neurocomputing 71(1–3):13–29
5. Jim H, Fearghal M, Liam M, Steve H, Brian M, Seamus C (2009) A reconfigurable and biologically inspired paradigm for computation using network-on-chip and spiking neural networks. Int J Reconfig Comput 2009:1–13

6. Seamus C, Fearghal M, Brian M, Sandeep P, Liam M, Snaider C, Jim H (2011) Hardware spiking neural network prototyping and application. Genet Program Evolvable Mach 12:257–280
7. Dmitri V, Ran G (2011) Scalable network-on-chip architecture for configurable neural networks. Microprocess Microsyst 35(2):152–166 (special issue on network-on-chip architectures and design methodologies)
8. Morgan F, Cawley S, McGinley B, Pande S, McDaid LJ, Glackin B, Maher J, Harkin J (2009) Exploring the evolution of NoC-based spiking neural networks on FPGAs. In: International conference on field-programmable technology, 2009. IEEE, Sydney, pp 300–303
9. Daniel NO, Scott W, Michael S (1993) Modular learning. MIT Press, Cambridge, pp 369–377
10. Sandeep P, Fearghal M, Seamus C, McGinley B, Jim H, Snaider C, McDaid L (2011) Addressing the hardware resource requirements of network-on-chip based neural architectures. In: International conference on neural computation theory and applications. NCTA, Paris
11. Nate K, Risto M (2008) Evolving neural networks for fractured domains. In: Proceedings of the 10th annual conference on genetic and evolutionary computation, GECCO '08. ACM, New York, pp 1405–1412
12. Harkin J, Morgan F, Hall S, Dudek P, Dowrick T, McDaid L (2008) Reconfigurable platforms and the challenges for large-scale implementations of spiking neural networks. In: International conference on field programmable logic and applications, 2008. IEEE, Heidelberg, pp 483–486
13. Yajie C, Hall S, McDaid L, Buiu O, Kelly P (2006) A solid state neuron for the realisation of highly scaleable third generation neural networks. In: 8th International conference on solid-state and integrated circuit technology, 2006. ICSICT, Beijing, pp 1071–1073
14. Furber S, Brown A (2009) Biologically-inspired massively-parallel architectures-computing beyond a million processors. In: Ninth international conference on application of concurrency to system design, 2009. ACSD, Augsburg, pp 3–12
15. Andres U, Ca Pea-Reyes, Sanchez E (2005) An FPGA platform for on-line topology exploration of spiking neural networks. Microprocess Microsyst 29(5):211–223
16. Pearson MJ, Pipe AG, Mitchinson B, Gurney K, Melhuish C, Gilhespy I, Nibouche M (2007) Implementing spiking neural networks for real-time signal-processing and control applications: a model-validated FPGA approach. IEEE Trans Neural Netw 18(5):1472–1487
17. Ros E, Ortigosa EM, Agis R, Carrillo R, Arnold M (2006) Real-time computing platform for spiking neurons (RT-spike). IEEE Trans Neural Netw 17(4):1050–1063
18. Vogelstein RJ, Mallik U, Vogelstein JT, Cauwenberghs G (2007) Dynamically reconfigurable silicon array of spiking neurons with conductance-based synapses. IEEE Trans Neural Netw 18(1):253–265
19. Ehrlich M, Mayr C, Eisenreich H, Henker S, Srowig A, Grubl A, Schemmel J, Schuffny R (2007) Wafer-scale VLSI implementations of pulse coupled neural networks. Proceedings of the international conference on sensors, circuits and instrumentation systems. CEA, Acapulco
20. Glackin B, McGinnity TM, Maguire LP, Wu QX, Belatreche A (2005) A novel approach for the implementation of large scale spiking neural networks on FPGA hardware. In: Computational intelligence and bioinspired systems. CIBS, Barcelona, pp 552–563
21. Schemmel J, Fieres J, Meier K (2008) Wafer-scale integration of analog neural networks. In: IEEE international joint conference on neural networks (2008) IEEE world congress on computational intelligence. IEEE, Hong Kong, pp 431–438
22. Chen Y, Hall S, McDaid L, Buiu O, Kelly P (2006) On the design of a low power compact spiking neuron cell based on charge-coupled synapses. In: International joint conference on neural networks, 2006. IJCNN, Brisbane, pp 1511–1517
23. Chen Y, McDaid L, Hall S, Kelly P (2008) A programmable facilitating synapse device. In: IEEE international joint conference on neural networks (2008) IEEE world congress on computational intelligence. IJCNN, Barcelona, pp 1615–1620
24. Furber S, Temple S, Brown A (2006) On-chip and inter-chip networks for modeling large-scale neural systems. In: Proceedings of 2006 IEEE international symposium on circuits and Systems, 2006. ISCAS, Kos, p 4
25. Emery R, Yakovlev A, Chester G (2009) Connection-centric network for spiking neural networks. In: 3rd ACM/IEEE international symposium on networks-on-chip (2009) NoCS 2009. IEEE, San Diego, pp 144–152
26. Pande S, Morgan F, Cawley S, McGinley B, Carrillo S, Harkin J, McDaid L (2010) EMBRACE-SysC for analysis of NoC-based spiking neural network architectures. In: System on chip international symposium on (SoC), 2010. SoC-2010, Tampere, Finland, pp 139–145
27. Johannes S, Wieringa BM, Matzke M, Mnte TF (1996) Hierarchical visual stimuli: electrophysiological evidence for separate left hemispheric global and local processing mechanisms in humans. Neurosci Lett 210(2):111–114

28. Van Essen DC, Anderson CH, Felleman DJ (1992) Information processing in the primate visual system: an integrated systems perspective. Science 255(5043):419–423
29. Binzegger T, Douglas RJ, Martin KAC (2007) Stereotypical bouton clustering of individual neurons in cat primary visual cortex. J Neurosci 27(45):12242–12254
30. Happel BLM, Murre JMJ (1994) Design and evolution of modular neural network architectures. Neural Netw 7(6–7):985–1004
31. Auda G, Kamel MS (1999) Modular neural networks a survey. Int J Neural Syst 9(2):129–151
32. Ronco GP (1995) Modular neural networks: a state of the art. Rapport technique CSC95026, vol 1. Center of System and Control, University of Glasgow, Lanarkshire, pp 1–22
33. Sheng-uei G, Shanchun L, Tan SK (2004) Neural network task decomposition based on output partitioning. J Instit Eng Singap 44:78–89
34. Bao-Liang L, Ito M (1999) Task decomposition and module combination based on class relations: a modular neural network for pattern classification. IEEE Trans Neural Netw 10(5):1244–1256
35. Thangavelautham J, D'Eleuterio GMT (2004) A neuroevolutionary approach to emergent task decomposition. Proceedings of 8th parallel problem solving from nature. Springer, Heidelberg, pp 991–1000
36. Khare VR, Yao Xin, Sendhoff B, Jin Yaochu, Wersing H, (2005) Co-evolutionary modular neural networks for automatic problem decomposition. In: The (2005) IEEE congress on evolutionary computation, vol 3. CES, Edinburgh, pp 2691–2698
37. Santos JM, Alexandre LA, de Sa JM (2006) Modular neural network task decomposition via entropic clustering. In: Sixth international conference on intelligent systems design and applications (2006) vol 1. IEEE Computer Society Press, Jinan, pp 62–67
38. Pizzuti C (2012) A multiobjective genetic algorithm to find communities in complex networks. IEEE Trans Evolut Comput 16(3):418–430
39. Brooks R (1986) A robust layered control system for a mobile robot. IEEE J Robot Autom 2(1):14–23
40. Cawley S, Pande S, McDaid L, McGinley B, Morgan F (2009) Memory efficient storage of reconfigurable topology information in network-on-chip based spiking neural networks, internal report. National University of Ireland, Galway, Bio-Inspired Electronics and Reconfigurable Systems
41. Maher J, McGinley B, Rocke P, Morgan F (2006) Intrinsic hardware evolution of neural networks in reconfigurable analogue and digital devices. In: 14th Annual IEEE symposium on field-programmable custom computing machines. FCCM, Seattle, pp 321–322
42. Vaughan R (2008) Massively multi-robot simulation in stage. Swarm Intell 2:189–208