Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2009, Article ID 908740, 13 pages
doi:10.1155/2009/908740

Research Article

A Reconfigurable and Biologically Inspired Paradigm for Computation Using Network-On-Chip and Spiking Neural Networks

Jim Harkin,[1] Fearghal Morgan,[2] Liam McDaid,[1] Steve Hall,[3] Brian McGinley,[2] and Seamus Cawley[2]

[1] *School of Computing and Intelligent Systems, University of Ulster, Derry BT48 7JL, Northern Ireland*
[2] *Bio-Inspired Electronics & Reconfigurable Computing Group, NUI Galway, Galway, Ireland*
[3] *Department of Electrical Engineering & Electronics, University of Liverpool, Liverpool L69 3GJ, UK*

Correspondence should be addressed to Jim Harkin, jg.harkin@ulster.ac.uk

Received 1 December 2008; Accepted 13 April 2009

Recommended by Michael Huebner

FPGA devices have emerged as a popular platform for the rapid prototyping of biological Spiking Neural Networks (SNNs) applications, offering the key requirement of reconfigurability. However, FPGAs do not efficiently realise the biologically plausible neuron and synaptic models of SNNs, and current FPGA routing structures cannot accommodate the high levels of interneuron connectivity inherent in complex SNNs. This paper highlights and discusses the current challenges of implementing scalable SNNs on reconfigurable FPGAs. The paper proposes a novel field programmable neural network architecture (EMBRACE), incorporating low-power analogue spiking neurons, interconnected using a Network-on-Chip architecture. Results on the evaluation of the EMBRACE architecture using the XOR benchmark problem are presented, and the performance of the architecture is discussed. The paper also discusses the adaptability of the EMBRACE architecture in supporting fault tolerant computing.

1. Introduction

Biological research has accumulated an enormous amount of detailed knowledge about the structure and function of the brain. The basic processing units in the human brain are neurons that are interconnected in a complex pattern [1]. The current understanding of real biological neurons is that they communicate through pulses and use the timing of the pulses to transmit information and perform computations. Spiking Neural Networks (SNNs), which interact using pulses or spikes, emulate more closely real biological neurons of the brain and have therefore the potential to be computationally more powerful than traditional artificial neural network models [2]. Fault tolerant computing can exploit the brain's behaviour in the repair and restoring of functionality. This is a feature of particular interest to SoCs designers where reliability is becoming difficult to guarantee postdeployment with scaling device geometries, and also in the deployment of computing systems in harsh environments.

Inspired by biology, researchers aim to implement reconfigurable and highly interconnected arrays of Neural Network (NN) elements in hardware to produce robust and powerful signal processing units. However, the standard topologies employed to model biological SNNs are proving difficult to emulate and accelerate in hardware, even for moderately complex networks.

Current software environments available for simulating Spiking Neural Networks (SNNs) provide biophysically realistic models, albeit with large simulation times [3]. Moreover, software simulations of SNN topologies and connection strategies face the problem of scalability in that biological systems are inherently parallel in their architecture whereas commercial PCs are based on sequential processing architectures [4] making it difficult to assess the efficiency of these models to solve complex problems.

The ability to reconfigure FPGA logic blocks and interconnect has attracted researchers to explore the mapping of SNNs to FPGAs [5–9]. Efficient, low-area/power implementations of synaptic junctions and neuron interconnect are key to scalable SNN hardware implementations. Existing FPGAs limit the synaptic density achievable as they map biological synaptic computations onto arrays of digital logic blocks, which are not optimised in area or power consumption for scalability [3]. Additionally, current FPGA routing structures cannot accommodate the high levels of neuron interconnectivity inherent in complex SNNs [3]. The Manhattan-style mesh routing schemes of FPGAs typically exhibit switching requirements which grow nonlinearly with the mesh sizes [10]. A similar interconnect problem exists in System-on-Chip (SoC) design where interconnect scalability and high degrees of connectivity are paramount [11]. The use of networking concepts has been investigated to address the interconnectivity problem using Network-on-Chip (NoC) approaches which time-multiplex communication channels [12]. The NoC approach employs concepts from traditional computer networking to realise a similar communicating hardware structure. The key benefit provided by NoCs is scalable connectivity; higher levels of connectivity can be provided without incurring a large interconnect-to-device area ratio [11].

This paper presents the EMulating Biologically-inspiRed ArChitectures in hardwarE (EMBRACE) hardware platform for the realisation of SNNs. EMBRACE uses an NoC-based neural tile architecture and programmable neuron cell which address the interconnect and biocomputational resources challenges. The paper illustrates how the EMBRACE architecture supports the routing, biological computation, and configuration of SNN topologies on hardware to offer scalable SNNs with a synaptic density significantly in excess of what is currently achievable in hardware [7–9, 13]. In addition, the paper discusses the potential opportunities EMBRACE offers in providing a new hardware information processing paradigm which has the inherent ability to accommodate faults via its neural-based structures.

Section 2 of the paper provides a review of related work and identifies the challenges of scalable SNN hardware implementations. Section 3 describes the proposed EMBRACE architecture, and Section 4 discusses the neural tile. Section 5 presents results on the evaluation of the architecture using the XOR benchmark problem and on neural tile implementations from the perspective of performance-architectural optimisations. Section 6 provides a discussion on the fault tolerance opportunities of the new paradigm, and Section 7 provides a summary and outline of future work.

2. Background

SNNs differ from conventional artificial NN models because information is transmitted by the means of pulses or spikes. Brain-inspired paradigms such as SNNs offer the potential of an elegant, low-powered, and robust method of performing computing. In particular, SNNs offer the potential to emulate the ability of the brain to tolerate faults and repair itself. The human brain continually replaces neurons by reconnecting to newly generated neighbouring neurons via synaptic junctions. This adaptability allows the brain to overcome faults, so providing the inspiration for establishing robust fault tolerant computers. Other similar approaches are currently under investigation for fault tolerant computing paradigms based on biological organisms [14].

When implemented in hardware, SNNs can take full advantage of their inherent parallelism and offer the potential to meet the demands of real-time fault tolerant applications. However, the first step towards SNN-based fault tolerant systems is the creation of a hardware platform which can support the levels of parallelism and adaptability required.

FPGAs have been widely recognised as a platform which can provide the adaptable requirements of NNs [8] and so enable rapid acceleration of NNs and hardware-in-loop training. The popularity of FPGAs has been fuelled by the reconfigurability offered by such devices and the introduction of platform FPGAs with increased logic and embedded processors [15].

2.1. Biological Spiking Neurons.

FPGA implementations of SNNs typically aim to accelerate and prototype biologically inspired computations [5, 7, 8, 13, 16]. However, FPGAs are not appropriately suited for efficient SNN implementation as they attempt to map biological neuron and synaptic computations onto general arrays of configurable digital logic blocks which are not optimised in area and power for dense network realisations [8]. Existing FPGAs can only provide limited synapse density. More importantly, they underutilise silicon area as larger numbers of general programmable units are required to achieve specific, optimal implementations. Providing configurable, dedicated computational SNN blocks within a custom FPGA-type structure will offer a more suitable reconfigurable platform for SNN accelerated prototyping, with optimised utilisation of hardware area and power. Also, issues associated with training must be addressed whereby the programmability and efficient storage of synaptic weights are accommodated.

2.2. FPGAs and Neuron Interconnectivity.

SNNs have a significant level of connectivity between neurons and, increasing SNN neuron density results in a nonlinear interconnect growth. For example, a 2-layered feed forward fully interconnected network with m neurons per layer exhibits an interconnect density of m^2, which rapidly increases as the number of neurons per layer increases. Neuron interconnection in current FPGAs is typically achieved using Manhattan style layouts with diagonal [17], segmented, or hierarchical 2-dimensional routing structures [10]. Scaling is one of the key issues associated with all complex electronic systems because interconnect consumes large areas of chip real-estate and subsequently causes longer critical path delays. For example, the switching requirements of Manhattan-style routing schemes typically grow nonlinearly with the number of logic units on the device [10]. This FPGA interconnect challenge is a significant limiting factor in the suitability of FPGAs for SNN implementation.

Researchers have investigated several routing optimisations and topologies in their attempts to improve the FPGA routing latency and performance. Rose et al. [17] investigated architectural-level techniques with the introduction of bus-based routing between clusters of multibit logic blocks to reduce routing interconnect density. Increased semiconductor multilevel metalisation has increased the number of metal layers that can be realised and has provided new opportunities for scaling FPGA routing using the Mesh-of-Trees (MoTs) topology [10]. More recently, nanotechnology has provided nanowire-based routing topologies for FPGAs [18].

2.3. Network-on-Chip (NoC). SoC design [11] has seen a considerable interconnect challenge with the introduction of multiprocessors. SoCs commonly use Network-on-Chip (NoC) schemes [19], with varied NoC topologies [20], router architectures, and the provision of low-power and high-Quality-of-Service (QoS) designs. Router architectures include asynchronous [19], circuit-based, packet, and wormhole switching [14, 21]. The NoC approach uses computer networking concepts to achieve a similar networking structure on hardware. The key benefit from using the NoC is scalable connectivity; higher levels of connectivity can be achieved without incurring a large ratio of interconnect-to-device area. Researchers have demonstrated performance benefits from incorporating NoC structures in FPGA application designs [14, 22–24]. The success of NoCs has seen the release of commercially available technologies from Silistix [25] and Arteris [26].

The SNN interconnect problem is similar to that of SoCs; SNNs have large numbers of neurons (typically in excess 1000 for complex applications), which exhibit high inter-neuron connectivity requirements.

Current attempts to realise SNNs using multiprocessors in hardware have included limited numbers of neurons due to connectivity issues [3, 6, 8, 9, 27–29]. An FPGA-based multiprocessor SNN [6], incorporating reduced interneuron connectivity, has been demonstrated. However, this implementation strategy does not scale efficiently since large FPGA SNN networks are inherently limited by the interconnect requirements between large numbers of processors [10]. Similarly, multiprocessor approaches to support the accelerated simulation of SNNs are also under investigation [27, 28] and use network-type communication structures. However, in general these approaches do not accommodate the dedicated computational requirements of the biological neurons and the temporal dynamics of SNN. Reported implementations provide suboptimal strategies which attempt to “force-fit” SNNs into current multiprocessor architectures which typically target regular data-path computational applications.

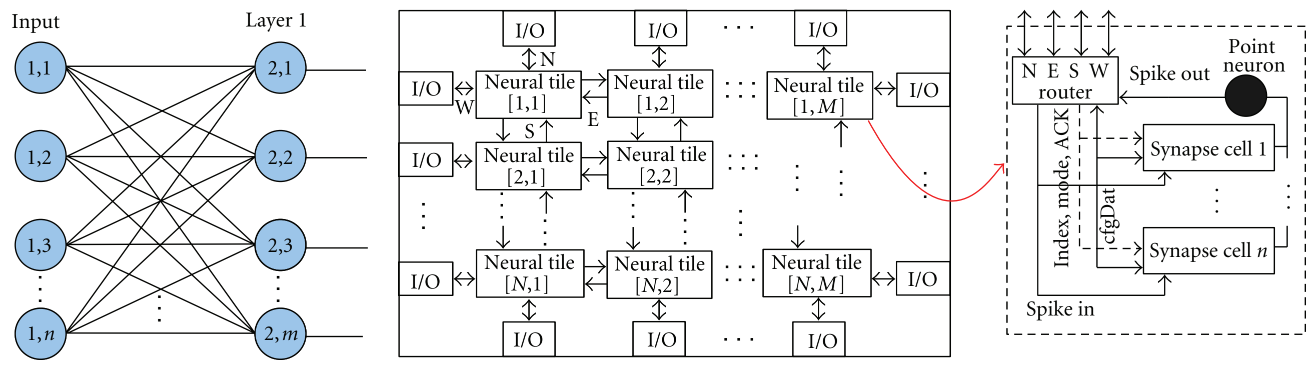
Several full-custom neuromorphic architecture devices have been proposed [13, 30, 31] which aim to address the inefficiencies of FPGAs by including optimised synaptic cells and Address Event Representation (AER) routing [32]. However, these architectures cannot scale due to limited connectivity provided by AER between neurons. More importantly, their “fixed” full-custom nature has a significant impact on the level of reconfigurability and prohibits the computational benefits afforded by programmable SNN interconnect.

2.4. Current Emulation Challenges. A major challenge is the development of bioinspired platforms which can support scalable low-power realisations and reprogrammable interconnect. In particular, the problem of SNN interneuron connectivity is the dominant obstacle that prohibits the implementation of biological scale NNs. The rapid increase in the ratio of fixed connections to the number of neurons self-limits the network size [1]. If scalable, reconfigurable networks are to be realised on SNN, domain-specific programmable devices, the interconnect issue must be addressed. This sets the primary focus of the paper. To address the large scale implementation issues of programmability, synapse weight storage, power consumption, and inter-neuron connectivity, a new device architecture tailored to the requirements of SNNs is required.

This paper proposes EMBRACE, a custom field programmable neural network architecture (EMBRACE) which merges the programmability features of FPGAs and the scalable interconnectivity of NoCs with low-area/power spiking neuron cells that have an associated training capability. The EMBRACE architecture supports the programmability of SNN topologies on hardware, providing an architecture which will enable the accelerated prototyping and hardware-in-loop training of SNNs. Earlier investigations by the authors in using NoC router strategies to implement large scale artificial neural networks [33] have demonstrated benefits in network scalability. In addition, the authors have developed an initial custom low-area/power programmable synapse cell with characteristics similar to real biological synapses [34, 35]. The following sections introduce the proposed EMBRACE architecture.

3. Embrace Architecture

The EMBRACE architecture is illustrated in Figure 1(b) as a 2-dimensional array of interconnected neural tiles surrounded by I/O blocks. The authors recognise that approaches to implementing the brain, which is a 3D structure, are currently limited to 2D because current fabrication techniques are not mature enough to reliably support large scale 3D SNN architectures. Therefore, the neural tiles are connected in North, East, South, and West directions forming a nearest neighbour connect scheme. Each neural tile can be programmed to realise neuron-level functions which collectively implement an SNN. An SNN is realised on the EMBRACE architecture by programming the tile functionality and connectivity. Consider the interconnectivity requirements of a feed-forward (FF), 2-layer $n \times m$ SNN network with each neuron in layer 1 connected to m neurons



(a) 2-layer SNN network (b) EMBRACE: Architecture (c) EEMBRACE: Neural Tile

Figure 1: Architecture overview.

in layer 2; see Figure 1(a). When a neuron in layer 1 fires (spikes), its pulse signal is propagated to the target neurons in layer 2 via dedicated individual lines. Using an NoC strategy, the same pattern of connectivity between layers is achieved through time-multiplexing of the communication channels. This significantly reduces interconnect density as the proposed architecture of Figure 1(b) includes a reduced number of fixed, regular-layout communication lines, and a network of NoC routers.

EMBRACE runtime and configuration data is propagated from source to destination via time-multiplexing using the NoC routing lines. For example, during runtime, spike events occurring in layer 1 are forwarded to the associated synapses of layer 2 via several router transmission steps. The proposed NoC strategy uses individual routers to group n synapses and the associated neuron, using a novel structure referred to as a *neural tile*, illustrated in Figure 1(c); the neural tile is viewed as a macro-block of EMBRACE. The novelty of the tile resides in the merging of analogue synapse/neuron circuitry with NoC digital interconnect to provide a scalable and reconfigurable neural building block. Other approaches [28, 29] do utilise NoC routers but with software models of synapse and neurons running on microprocessors. Figures 2 and 3 illustrate how the $n \times m$ network can be realised using the NoC strategy, whereby m neural tiles are required, with each one corresponding to one of the m postsynaptic neurons of layer 2; a feed-forward (FF) network with 10^3 neurons per layer would require 10^3 neural tiles each containing 10^3 synapses. The m neural tiles are arranged in a 2D array structure, as shown in Figure 3, with i number of rows and j columns, for example, $i = 200$ and $j = 5$ for $m = 10^3$. Note that Figure 2 highlights which synapses are connected to neuron (2, 1), and Figure 3 illustrates how the synapse and neuron functionality are mapped to tile number 1; for example, each synapse maps to one of the n synapse cells in the tile. The mapping process is repeated for neurons (2, 2) through to (2, m), where the unique n synapses for each neuron are allocated to tiles 2 through to m, respectively.

3.1. Distribution of Computational Resources. The aim of the NoC strategy is to use the router of each tile to communicate spike events to its group of n synapses. This enables a reduced

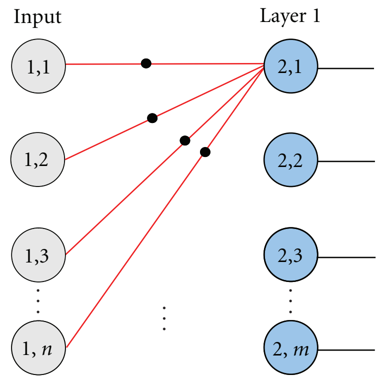


Figure 2: n x m network showing only the synapse inputs to neuron (2,1). Each of the m neurons in layer 1 has n synapse connections.

number of connections; for example, an SNN interconnect density of 10^6 ($n \times m$) can be implemented using 4×10^3 ($4 \times m$ connections). The 4 term stems from the N, E, S and W router connections. The n individual synapses in a neural tile are referred to as *synapse cells* and are combined with the point neuron to form the neuron cell. The synapse cell is analogue in nature and captures the pertinent biological features of real synapses [34]. The output responses from each synapse cell are connected together to produce the desired biological response from the point neuron.

Due to the time multiplexing manner of the interconnections between neurons, the data transfer time (spike-interval) between neurons is increased; however this is not a significant degradation to the speed-performance of EMBRACE as the biological speed (spike-interval) of the human brain is typically in the order of 10 milliseconds [2, 4].

3.2. Spike Event Communication. The inputs and outputs of the synapse cells are controlled via the NoC router. The events of a spike train are received as data packets from neighbouring routers where each spike-event packet includes a source address (indicating the SNN neuron/tile in which the event originated) and destination address (of neural tile and synapse). Figure 4 illustrates the packet format where the

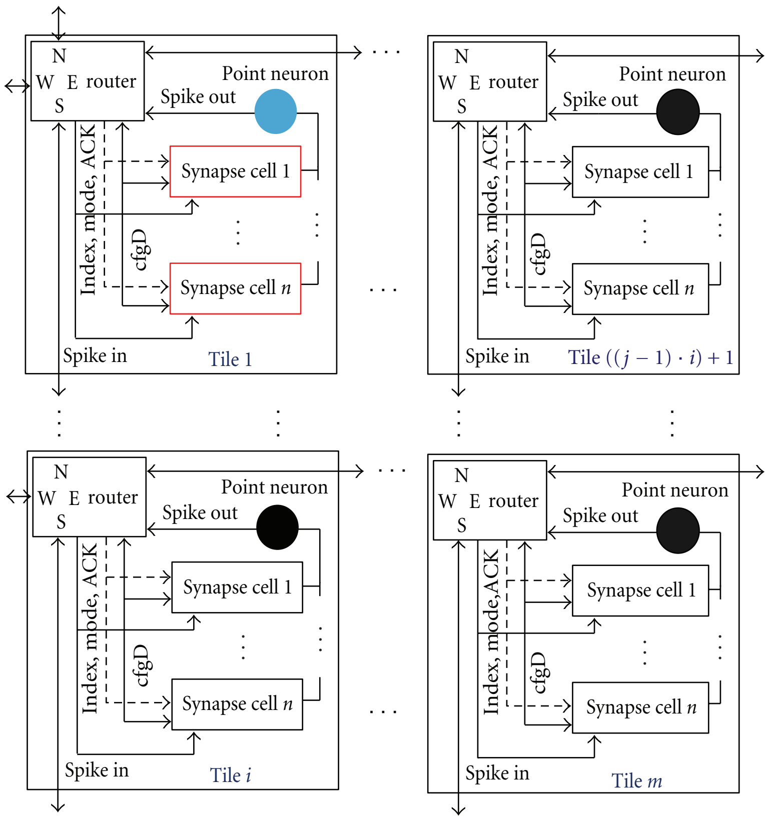


Figure 3: Mapping SNNs to EMBRACE: realisation of the neuron (2,1) and its n synapse connections to tile 1.

header contains information on the type of data contained in the payload (i.e., a packet can be contain runtime, configuration, or error data). The maximum payload size per packet is 22-bits where in runtime mode it contains the addresses of the source and target neuron. Each packet is transmitted as two 12-bit flits in runtime mode; in configuration mode the number of flits can verify as discussed later in Section 4.1. Tile routers send an output packet each time a spike-event occurs within the tile. Exploiting the relatively lowfrequency of biological spike trains (~Hz) [2], EMBRACE can use a regular and scalable NoC structure. The time-multiplexing of spike data along router paths between SNN layers enables large parallel networks and high levels of routability, without the need for an overwhelmingly large number of connections; see Figure 1(a). The proposed method of connecting neural tiles enables various SNN topologies to be realised for example, multilayered feed-forward and recurrent networks can be implemented.

4. Reconfigurable Neural Tile

The EMBRACE neural tile is illustrated in Figure 5 and highlights the connections between the NoC router, synapse cells, and point neuron. The packet-switched router implements 12-bit communication paths with buffer support where a round-robin scheduling policy is used for arbitration at the buffer input/outputs. The NoC router uses an XY routing scheme where all flits are transmitted in the same path direction (wormhole operation).

The intra-tile communication buses include the following signals: *Spike I/P* and *O/P*, *Mode*, *ACK*, *Config Data* and *Indexing*.

(i) *Spike I/P* initiates a spike on an individual synapse cell.

(ii) *Spike O/P* receives spike events from the neuron.

(iii) *Mode* specifies the tile operation, namely, runtime or configuration programming

(iv) *Indexing* bus is used to address individual synapse cells (via address decoders) for receiving spike events or configuration data.

(v) *ACK* acknowledges the correct synapse addressing.

(vi) The *Config Data* bus is used to transmit configuration data to the cells. Cells configure the connections to the q global voltage lines, V, which are common to tiles and run throughout the device.

Each neural tile has a unique address, and the synaptic connectivity is also specified using an Address Table (AT) within each router. The AT is programmed during EMBRACE configuration to specify the desired connectivity between tiles and enables spike events to be routed. A spike event is detected at the *Spike O/P*, and the AT identifies which target neural tiles and synapses must receive event notification.

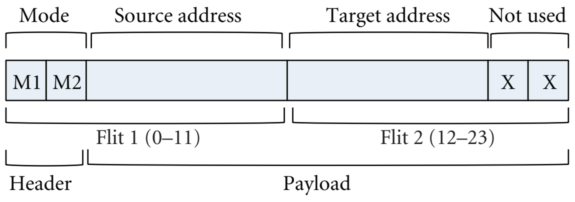


Figure 4: General Packet Format.

Table 1: Packet header definitions.

Header Information	M1, M2
Runtime mode	00
Configuration mode (Start)	01
N/A	10
Configuration mode (End)	11

4.1. Reconfigurable and Runtime Operations. The neural tiles operate in one of two modes: *runtime* or *configuration*. This is defined by the information in the packet header (see Table 1) where "00" and "01" define the payload to be either runtime or configuration, respectively. In runtime mode, EMBRACE routes spike events (data packets) and computes the programmed SNN functionality. In configuration or programming mode, EMBRACE is configured to realise particular synapse models and the desired SNN topology. Configuration data is also delivered to the EMBRACE device in the form of data packets where each packet is addressed to a particular neural tile and contains information on the configuration of the router's AT, the selection of cell synapse weights via the programmable voltage lines, V_q, and other neural tile parameters. However, in this mode, a packet typically contains more than two flits as the payload exceeds the 22-bit limit shown in Figure 4. The additional flits are required as several synapses are often reconfigured within a single tile. In this case, *Configuration (Start)* and *(End)* information is specified in the packet, header as shown in Table 1. For example, when a router detects "01" in the header of a packet it is alerted that the following packet payload contains configuration data and to expect more flits. When it detects the value "11" in the header, it recognises that the transmission of configuration data is complete for the tile. A wormhole routing scheme is used to forward all flits in the payload to the target tile.

This strategy fully exploits the flexibility of the NoC structure by additionally using it to select and distribute configuration data to EMBRACE tiles. It is envisaged that this type of configuration mechanism would also be able to partially support the repair implementation of a tile neural structure in the event of faults.

4.2. Synapse Weight Storage. Each synapse is analogue in nature and models key pertinent biological features of real synapses [34], such as long and short term plasticity. Spike stimuli are received by synapse inputs. Synapse outputs within a neural tile are combined as inputs to the point neuron of the tile. In keeping with biological plausibility, synapse weight updates for long term plasticity should be governed by a Hebbian-based rule [35], and the authors envisage an off-line training procedure that uses this rule.

Table 2: Example of synapse weight configurations.

S6	S5	S4	S3	S2	S1	Synapse cell weight
1	1	1	1	1	1	5.04 V
⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	0	0	0	0	0	2.56
⋮	⋮	⋮	⋮	⋮	⋮	⋮
0	0	0	0	0	1	0.08 V
0	0	0	0	0	0	0 V

While floating gate transistors with submicron feature sizes would yield compact analogue weight storage, there are significant issues to be resolved with this technology if it is to be a viable analogue memory storage capability for large scale neural networks: dynamic range, sensitivity, and stability [36]. In view of this we propose the novel weight distribution and storage architecture, shown in Fig.6, where for each synapse cell, any number or combination of p identical (smaller) charge-transfer synapses (CTSs) can be hardwired to programmable weight voltage levels during configuration. V_1 to V_q rails provide a range of supply weights (voltages), selected using digitally controlled analogue switches (S_1 to S_p). Figure 6 illustrates how the current outputs of the CTSs are summed when weight voltages V_1 and V_q are applied to synapse 1 and p. Table 2 illustrates an example synapse cell weight selection where $q = p = 6$, $V_1 = 0.08$, $V_2 = 0.16$, $V_3 = 0.32$, $V_4 = 0.64$, $V_5 = 1.28$, and $V_6 = 2.56$ volts. Varying the number of voltage rails (q) increases the weight range, and varying the weight rail voltage values modifies the weight resolution. Selecting combinations of rail voltages using S_1–S_6 provides 2^6 possible synapse weights where a sample of these combinations is shown in Table 2.

If we consider the first row in the table where all switches are closed, then each CTS is activated to a level dictated by the associated supply rail (weight). Therefore, each CTS will output a postsynaptic current whose magnitude is proportional to its weight voltage, and the net postsynaptic response is the aggregate of all CTS outputs. This is further illustrated in the remaining rows of the table where S_6 and S_1 are activated in rows two and three, respectively. In the former only one CTS is activated with a postsynaptic response proportional to a weighting of 2.56 volts whereas for the latter we have again a single CTS activated by S_1 with an output postsynaptic current proportional to a weight value of 0.08 volts. Note, it is envisaged that $q \neq p$, where more than one switch can be connected to a single voltage rail.

The authors recognise the need for a nonvolatile memory capability and are currently investigating the replacement of the latches/switches of Figure 6 with flash memory techniques using floating gates. What is novel about this approach to nonvolatile weight storage is that it only

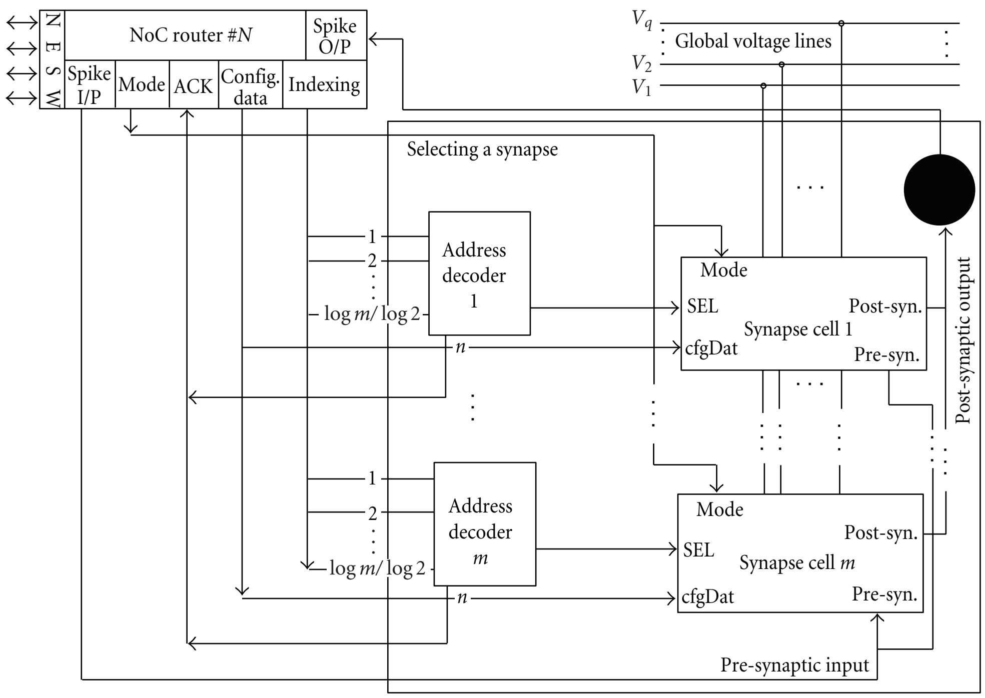
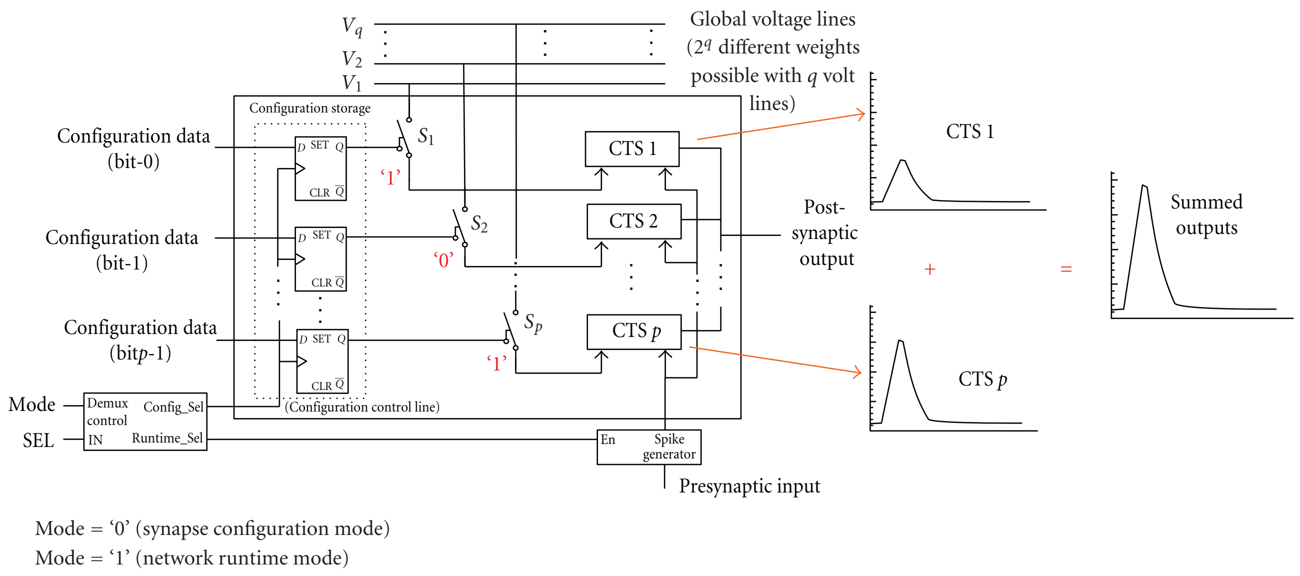


Figure 5: EMBRACE: neural tile structure.



V_q
V_2
V_1
Global voltage lines (2^q different weights possible with q volt lines)
Configuration storage
Configuration data (bit-0)
Configuration data (bit-1)
Configuration data (bitp-1)
D SET Q
CLR $\bar{Q}$
S_1
S_2
S_p
'1'
'0'
'1'
CTS 1
CTS 2
CTS p
Post-synaptic output
Summed outputs
+
=
Mode
SEL
Demux control
IN
Config_Sel
Runtime_Sel
(Configuration control line)
En
Spike generator
Presynaptic input
Mode = '0' (synapse configuration mode)
Mode = '1' (network runtime mode)

Figure 6: Synapse cell with p CTS.

requires that the transistors associated with each floating gate operate in either a fully on or off mode (binary operation) thereby avoiding the problems associated with dynamic range, sensitivity and, stability.

5. Embrace Performance

This section presents results on the functional evaluation of EMBRACE for the exemplar eXclusive-OR (XOR) problem. Results on the area performance of the neural tile are presented, and optimisations of the architecture for scalability are illustrated.

5.1. Demonstrative Application. The XOR (eXclusive-OR) problem has been used by researchers as a standard benchmark to verify the operation or evaluate the performance of artificial NNs [37] and more recently SNNs [38]. The XOR problem is based on the principle of logical pattern matching

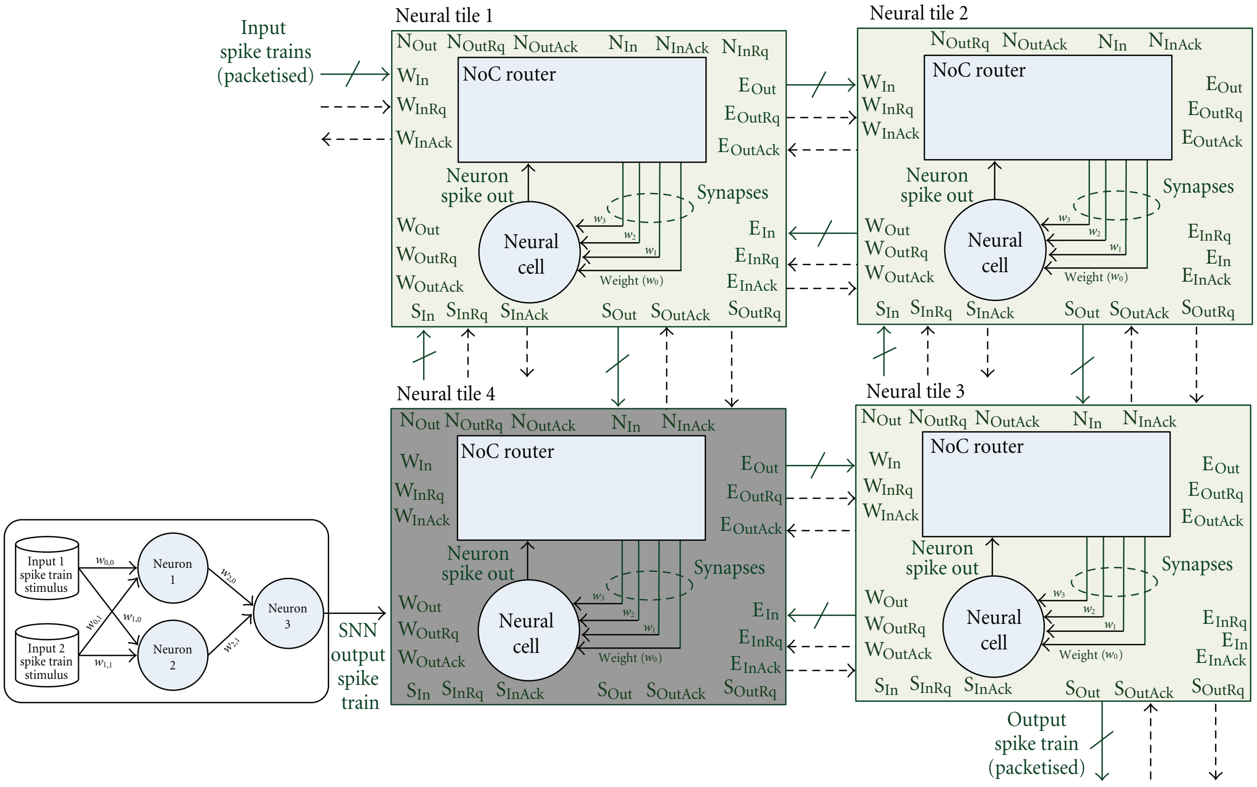

Input spike trains (packetised)
Neural tile 1
Neural tile 2
Neural tile 3
Neural tile 4
NoC router
Neuron spike out
Synapses
Neural cell
Weight (w_0)
w_1
w_2
w_3
N_{Out} N_{OutRq} N_{OutAck} N_{In} N_{InAck} N_{InRq}
W_{In} W_{InRq} W_{InAck}
W_{Out} W_{OutRq} W_{OutAck}
E_{Out} E_{OutRq} E_{OutAck}
E_{In} E_{InRq} E_{InAck}
S_{In} S_{InRq} S_{InAck} S_{Out} S_{OutAck} S_{OutRq}
Input 1 spike train stimulus
Input 2 spike train stimulus
Neuron 1
Neuron 2
Neuron 3
$w_{0,0}$
$w_{0,1}$
$w_{1,0}$
$w_{1,1}$
$w_{2,0}$
$w_{2,1}$
SNN output spike train
Output spike train (packetised)

Figure 7: Embrace configuration for SNN-based XOR function.

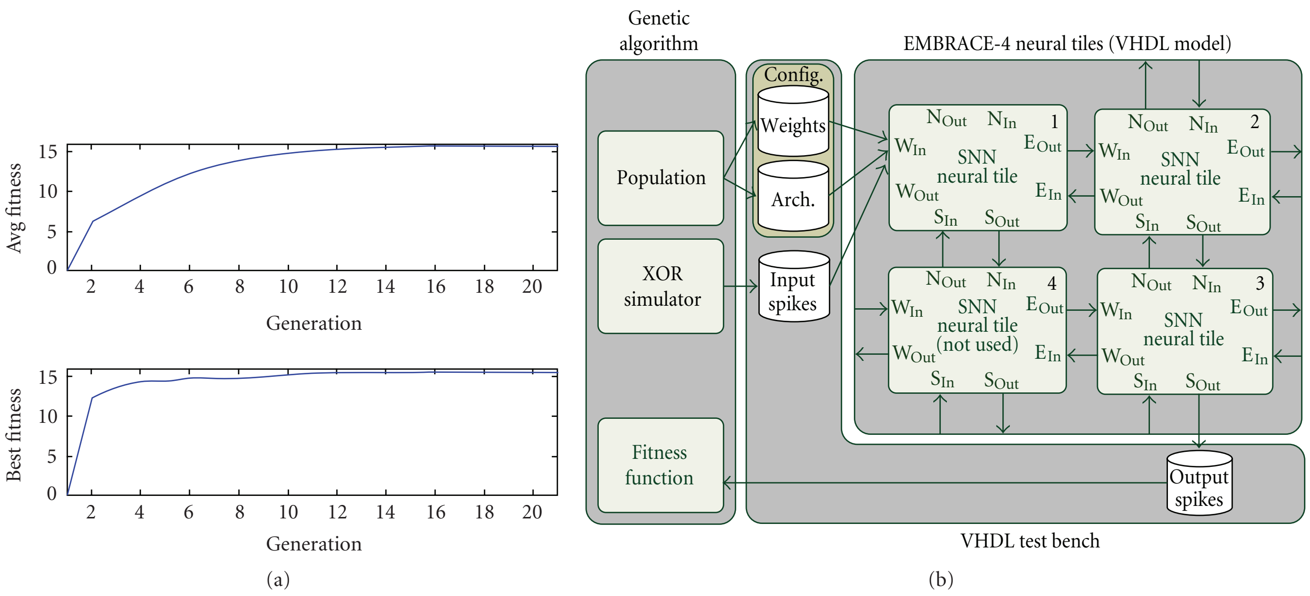


Figure 8: Evaluation setup of SNN-based XOR function (realised on 2 × 2 EMBRACE).

whereby a system must decipher whether or not an input pattern has an equal or unequal number of logic “0” or “1”. For example, take the simple case where the XOR is applied to the binary pattern “11100100”; the system must be able to detect that the pattern has an equal number of zeros to be classed as an XOR solver. The XOR problem is not linearly separable and therefore an appropriate test case to exercise SNNs.

A 2-layer feed-forward SNN consisting of 3 neurons (2 input/1 output) was created to solve the XOR; whereby each neural cell utilised one point neuron and two synapse cells. Figure 7 illustrates the layout of the SNN-based XOR configuration across a 2×2 array of neural tiles. In this example only 3 tiles are required, however; four programmable synapses (the maximum number expected for a 2×2 array) are included in each tile, one for each of the neurons in the architecture (5 CTSs per synapse cell). Note that although four synapse cells are shown in Figure 7, only two synapses per tile are utilised in the XOR problem. For larger problems additional tiles can be included by appending to the N, E, S and, W directions of the four example tiles, as depicted in Figure 1(b).

A hardware model of EMBRACE was created to evaluate its functionality in emulating the SNN-based XOR problem and, in particular, the configuration of the synapse weights and runtime switching of packet data between the NoC routers. Synthesisable VHDL was created for the digital NoC router and behavioural VHDL for the analogue synapse cells [39]. The XOR mapped EMBRACE model was simulated using commercial VHDL software tools running on a standard PC. Note: The SNN-based XOR was mapped to EMBRACE by allocating each single neuron and its multiple synapses to different tiles. The architecture was programmed by configuring the synapse weights and neuron threshold of each tile via a test bench. Figure 8 illustrates the test bed used to simulate the model of the XOR problem on the 2×2 tile array; a Genetic Algorithm (GA) was used to train the SNN to recognise the correct XOR input patterns over 20 generations. The GA fitness measure was based on the XOR function where a maximum of 16 was evaluated for all correct matched input patterns. The left-hand side of Figure 7 shows the convergence rate in generations and both the average and maximum fitness measures. Note that the GA-driven training process defines the inputs of known spike train patterns to neural tile 1 where each router forwards the target spike packets to their destination synapses in layer 1 (tiles 1 and 2).

For each of the postsynaptic neurons in layer 1 that fired, additional packets were generated by tiles 1 and 2 and then routed to the neuron in layer 2 (located in tile 3). Figure 9 illustrates the runtime mode of the tile with two packets being received and a spike train consisting of two spikes being directed to the 5 CTS, that is, one synapse cell.

During each GA generation, new synapse weights were reconfigured in the synapse cells using the NoC router of each tile and evaluated with the XOR fitness measure. This GA-driven training process was repeated until EMBRACE accurately performed the XOR function, that is, until an optimal or near-optimal set of weights were identified. The XOR evaluation of the 2×2 EMBRACE model proved successful as it was able to reconfigure weights, route packets, and perform the XOR operator on input patterns.

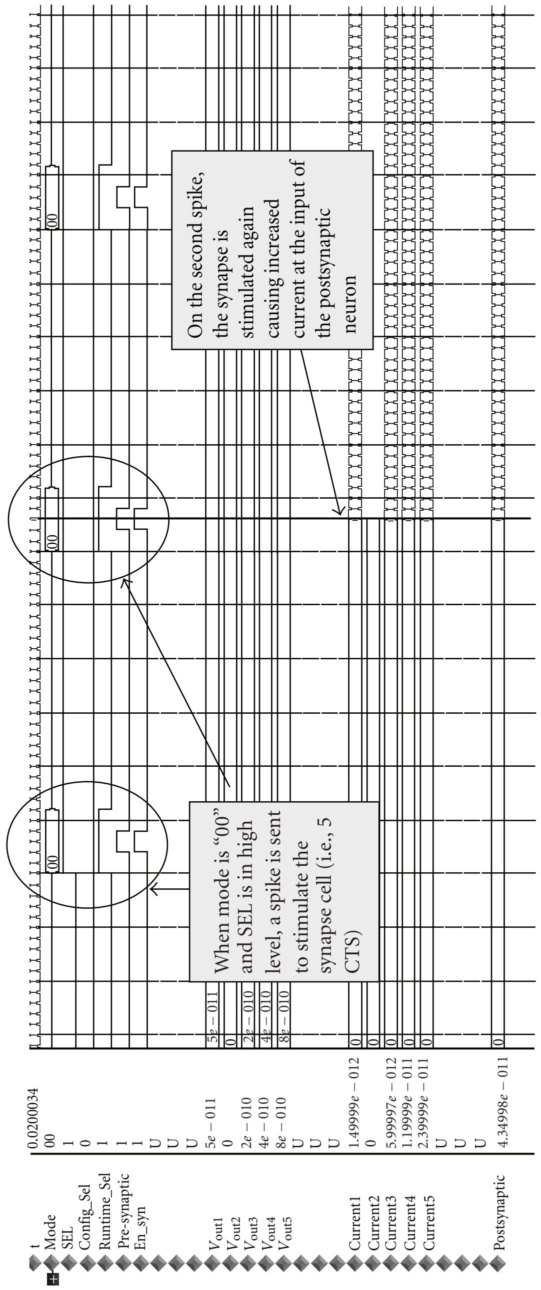


Figure 9: Runtime mode of the tile showing two packets being received. A spike train consisting of two spikes is directed to the 5 CTS. The total postsynaptic current of the individual CTS currents (current 1, 3, 4, and 5) increases in value with each spike stimulus.

5.2. Neural Tile Resources and Performance. The custom layout for an indicative neural tile with 10 synapse cells (each cell containing 10 CTSs) and a single point neuron [34] is illustrated in Figure 10. Note that the number of synaptic cells is set by the fan-in, and to avoid limiting the response time of the point neuron, the cells density was set to 10: a higher fan-in can easily be achieved by buffering the input of the point neuron circuit. The configuration storage, synapse voltage selection, and NoC router are not included in the layout. The 10 programmable synapse cells and neuron occupy a compact area size of $3 \times 11\,\mu$m using 90 nm CMOS technology (test circuits for the programmable synapse and neuron are currently being fabricated by Europractice). The small synapse area indicates the potential to realise compact neural tiles. Moreover, the synapse operates in transient mode, and consequently the associated power

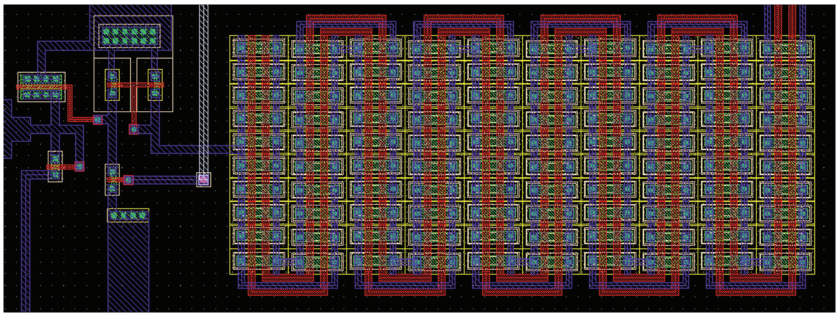


Figure 10: Neural Cell: layout of single point neuron and 10 synapse cells (10 CTS per cell).

consumption is small. Consider the extreme case where the CTS is stimulated by a 1 MHz presynaptic spike train and assume that the applied weight voltage to the CTS is 5 V (maximum weight). For each presynaptic spike, an average current spike of amplitude 10^{-7} amps is produced at the output of the CTS for a duration of 2 nanoseconds [34]. This yields a power consumption/cycle of 1 nW, which is orders of magnitude improvement when compared to circuit-based implementations of dynamic synapses [40]. Although the estimation does not include power consumption due to the tile's configuration circuitry, it does illustrate the application of the proposed synapse for low-power SNN implementation.

The initial NoC router design used in the XOR application has been implemented on a Xilinx XC2V1000 FPGA [33]. Data from [22] estimates the *AEthereal* NoC router implementation of 2658 LUTs on the Virtex-4 to be equivalent to 2.28 mm^2 using 90 nm CMOS technology. Using this data, a conservative estimate of 0.201mm^2 is determined for the area of the proposed router comprising 234 LUTs (synthesized for Virtex-4). This analysis provides an early area estimate of the proposed neural tile (single router, neuron and 10 synapse cells) at 201×10^{-3} mm^2. This initial estimate suggests promising results for EMBRACE networks, since the tile area is relatively small.

In terms of NoC router latency or performance, the router can process incoming data packets every 10 clock cycles with source packet generation requiring 12 cycles. The router was verified at 200 MHz on the Xilinx XC2V1000 providing 50 and 60 nanoseconds latencies, respectively. In the example of XOR problem the latency is negligible (~2.9 microseconds :10 spikes in the input train); however, for large scale network problem sizes the overall latency of EMBRACE is still faster than the biological spike-interval time, *ts*. Consider the network of Figure 1(a) with ($n = m = 10^3$) requiring 10^6 connections giving $m = 10^3$ (such network sizes are used in image processing tasks such as object tracking [3]).

Assume $ts = 10$ millisecondss [2], a clock frequency of 200 MHz ($tc = 5$ nanoseconds), and a 10 clock cycle period to receive and forward a packet between each NoC router; the time for the longest routing path, tp, can be estimated at $(m \times tc) = 5$ microseconds. Now assume the condition when the upper bound on data traffic load occurs with all neurons firing at the same. In consideration that each tile operates in parallel and multiple paths exist, the total latency of the 10^3 neuron layer is not an accumulation of $n \times m$ individual path times but rather an amortisation of path, times across the number of routing paths available. As each router can route data in all 4 co-ordinate paths the maximum number of parallel routes is $4 \times m = 10^3$. For the example network of 10^6 connections, the total latency with the upper bound condition will be $[10^6 \times tp/4\,m] = 5$ milliseconds. This demonstrates that the time required to time-multiplex the 10^6 connections between layers is significantly less than the 10 milliseconds interspike period and can therefore be performed in real-time. It should be noted that not all neurons spike at the same time [2, 4] so, a further speed improvement is possible.

5.3. Optimising EMBRACE. Research is currently underway to optimise the tile area further and so support the creation of dense EMBRACE architectures. For example, the addition of larger numbers of neurons and synapses within individual neural tiles will reduce the number of routers required for the SNN implementations; this has the effect of creating multicast groups where many neurons are assigned a single unique router address within the architecture. This strategy will allow reduction of the overall SNN area requirements as shown in Figure 11 (estimated by summing the router, synapse, and neuron areas). For example, EMBRACE can implement a fully connected 2-layered SNN with 10^4 neurons per layer, using a neuron/router ratio of 5 with an associated area of 642 mm^2 (shown by the dashed line in Figure 11). It can be seen that this scaling trend for increased neuron to router (N/R) ratios reduces the total device area of EMBRACE. The trade-off for minimising the number of routers (i.e., creating multicast groups) increased complexity in the router interface due to addressing and arbitration functions. This will increase the latency performance of the NoC router; however, the multicast groups will minimise the number of packet transmissions between routers and therefore minimise the overall network latency incurred.

Further optimisation of EMBRACE can be achieved by exploring the minimum number of CTS, p, per synapse cell. Figure 6 illustrated p CTS per cell where the dynamic range of the cell is defined by the number of CTS p and

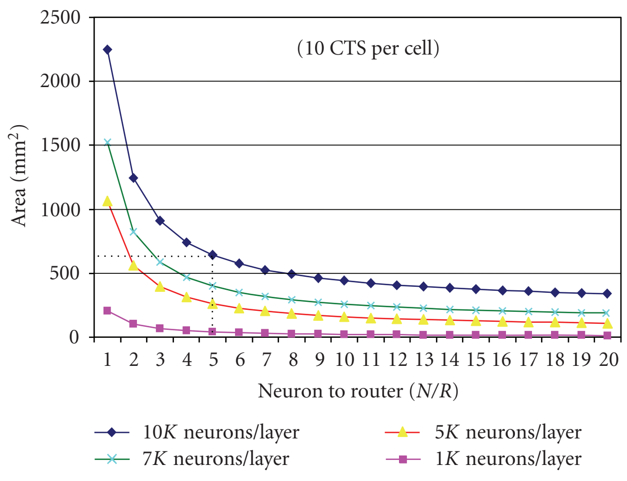


Figure 11: Optimisation: increasing number of neurons per tile.

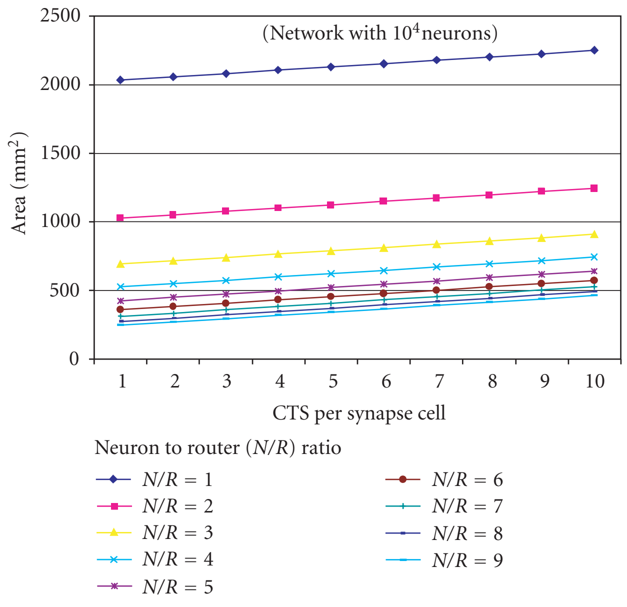


Figure 12: Optimisation: number of CTS per synapse cell.

voltage values V_q. Reducing the number of CTS per cell can reduce the EMBRACE area requirements. For example, in Figure 12 the total area for a network, with 10^4 neurons per layer and a N/R ratio of 5, can be reduced from ~642 to ~522 mm^2 by reducing the number of CTS per cell from 10 to 5, respectively. This linear reduction in area is consistent for other N/R ratios. Further research is currently underway to identify the optimum number of p CTS per synapse cell.

5.4. Scalability. Figure 13 illustrates the total EMBRACE area as a function of neuron density with a neuron/router ratio of 5. Since the proposed NoC supports a regular layout of the tiles and neuron communication, the interconnectivity between layers using EMBRACE does not limit the network size that can be implemented, unlike existing strategies. The

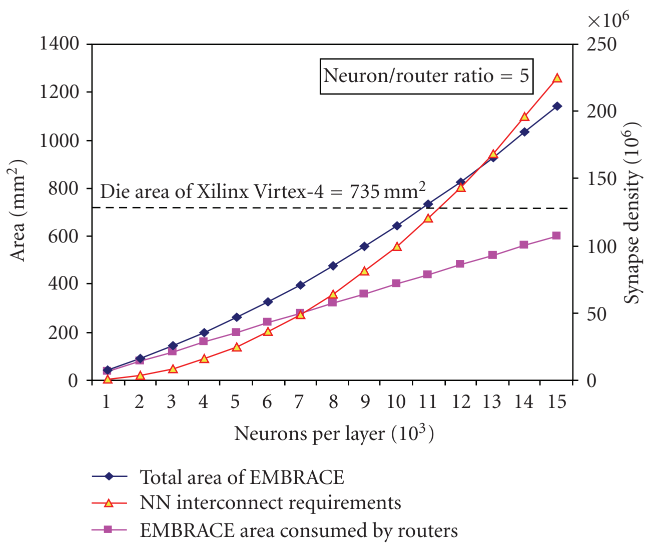


Figure 13: EMBRACE scalability.

figure also illustrates the linear growth of the area occupied by the routers with the SNN interconnect requirements. Although Figure 13 illustrates a synaptic density of the order of 10^8 for a die area equivalent to the Xilinx Virtex-4 FPGA [15], the calculations underestimate the total EMBRACE area due to configuration circuitry and global chip wiring. Results do however highlight a scaling trend, indicative of the proposed architecture.

6. Discussion

The adaptability of EMBRACE provides a framework to exploit the ability to reconfigure itself and also provide alternative routing paths for damaged areas using NoCs. For example, two levels of fault tolerance could be supported, namely, at the synapse and tile levels.

The abstract basis of SNNs is the strengthening and weakening of synaptic weights, whereby training algorithms are used to derive appropriate weight values to reflect a mapping between the input and desired output data. Faults occurring in individual synapses could be tolerated by using such algorithms to appropriately retrain the network when the output deviates from the “golden” patterns. This process can be achieved via the strengthening and/or weakening of neighbouring synapse weights within the tile.

At a more coarse level, complete tiles could be remapped or relocated to fault-free tiles, whereby the configurable data of a damaged tile is reconfigured to a new tile with updated router address contents and synaptic weights. For example, the adaptability of each neural tile could be exploited to allow its contents to be updated during runtime by any of its four coordinate neighbours. When a fault is detected in one of the neighbouring tiles, using a similar fault detection scheme such as [41] the centre tile could take control and relocate the configuration data of the faulty tile to a new available tile. An address update packet could then be broadcast to all tiles indicating the new location of the repaired tile. This approach would therefore provide a more robust distributed repair mechanism as opposed to a centrally controlled strategy.

Such a fault tolerant computing paradigm has the potential to be used in aerospace avionics (e.g., safety systems in space-craft or aeroplanes) or automotive electronics (e.g., engine management systems) where exposure to harsh environmental conditions requires adaptive computing systems.

7. Summary and Future Work

The challenges for large scale SNN implementations on reconfigurable platforms have been highlighted. A novel EMBRACE hardware architecture has been presented which utilises NoC routers and novel synapse cells to provide the programmable and scalable interconnect and biocomputational resources, required in large scale SNNs. Results have been presented, which demonstrate the functionality and performance of EMBRACE using the benchmark XOR problem. Similarly, results on the scalability of EMBRACE in terms of area and power have been presented and the capability of the EMBRACE architecture to support dense SNNs has been illustrated. Overall, the approach has been proven to be very promising.

Future work will explore NoC routing polices and topologies for SNN spike traffic. Optimising the number of neurons per neural tile will be tested, and the application of asynchronous NoCs will be investigated to support the QoS for spike-event traffic. Approaches to on-chip training will also be investigated to support the programming of the synapse cells and router. Integration of the neural tile configuration architecture and router with the analogue neural cell will be realised with the aim of developing a full-scale EMBRACE implementation. Additionally, the adaptability of EMBRACE provides an ideal framework to further explore bioinspired fault tolerant computing paradigms, whereby on-chip SNN learning and reprogrammability of EMBRACE neural structures could potentially emulate the repair behaviour of the brain. Future work will also explore the requirements to support on-chip, self-adaptation for fault tolerance computing applications.

References

[1] M. A. Arbib, *The Handbook of Brain Theory and Neural Networks*, A Bradford Book, Bradford, UK, 1995.

[2] W. Maass, *Computation with Spiking Neurons: The Handbook of Brain Theory & NNs*, MIT, Cambridge, Mass, USA, 2001.

[3] L. P. Maguire, T. M. McGinnity, B. Glackin, A. Ghani, A. Belatreche, and J. Harkin, “Challenges for large-scale implementations of spiking neural networks on FPGAs,” *Neurocomputing*, vol. 71, no. 1–3, pp. 13–29, 2007.

[4] H. Markram, “The blue brain project,” *Nature Reviews Neuroscience*, vol. 7, no. 2, pp. 153–160, 2006.

[5] A. Ghani, T. M. McGinnity, L. P. Maguire, and J. Harkin, “Area efficient architecture for large scale implementation of biologically plausible spiking neural networks on reconfigurable hardware,” in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL ’06)*, pp. 939–940, Madrid, Spain, August 2006.

[6] B. Glackin, et al., “Novel approach for the implementation of large-scale spiking neural networks on FPGAs,” in *Proceedings of the Artificial Neural Network Conference*, pp. 552–563, 2005.

[7] A. Upegui, C. A. Peña-Reyes, and E. Sanchez, “An FPGA platform for on-line topology exploration of spiking neural networks,” *Microprocessors & Microsystems*, vol. 29, no. 5, pp. 211–223, 2005.

[8] M. J. Pearson, A. G. Pipe, B. Mitchinson, et al., “Implementing spiking neural networks for real-time signal-processing and control applications: a model-validated FPGA approach,” *IEEE Transactions on Neural Networks*, vol. 18, no. 5, pp. 1472–1487, 2007.

[9] E. Ros, E. M. Ortigosa, R. Agis, R. Carrillo, and M. Arnold, “Real-time computing platform for spiking neurons (RT-spike),” *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 1050–1063, 2006.

[10] A. DeHon and R. Rubin, “Design of FPGA interconnect for multilevel metallization,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 10, pp. 1038–1050, 2004.

[11] S. Furber, “Future trends in SoC interconnect,” in *Proceedings of the IEEE VLSI-TSA International Symposium on VLSI Design, Automation and Test (VLSI-TSA-DAT ’05)*, pp. 295–298, Hsinchu, Taiwan, April 2005.

[12] L. Benini and G. DeMicheli, “NoCs: a new SoC paradigm,” *IEEE Computers*, pp. 70–78, 2002.

[13] R. J. Vogelstein, U. Mallik, J. T. Vogelstein, and G. Cauwenberghs, “Dynamically reconfigurable silicon array of spiking neurons with conductance-based synapses,” *IEEE Transactions on Neural Networks*, vol. 18, no. 1, pp. 253–265, 2007.

[14] C. Schuck, S. Lamparth, and J. Becker, “artNoC—a novel multi-functional router architecture for organic computing,” in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL ’07)*, pp. 371–376, Amsterdam, The Netherlands, August 2007.

[15] Xilinx, Virtex-4 User Guide, Ver. (V1.3), 2005.

[16] C.-S. Bouganis, P. Y. K. Cheung, and L. Zhaoping, “FPGA-accelerated pre-attentive segmentation in primary visual cortex,” in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL ’06)*, pp. 199–204, Madrid, Spain, August 2006.

[17] A. Ye and J. Rose, “Using bus-based connections to improve field-programmable gate-array density for implementing datapath circuits,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 5, pp. 462–473, 2006.

[18] G. S. Snider and R. S. Williams, “Nano/CMOS architectures using a field-programmable nanowire interconnect,” *Nanotechnology*, vol. 18, no. 3, 2007.

[19] W. J. Bainbridge, L. A. Plana, and S. B. Furber, “The design and test of a smartcard Chip using a CHAIN self-timed Network-on-Chip,” in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE ’04)*, pp. 274–279, Paris, France, February 2004.

[20] S. Jovanovic, C. Tanougast, S. Weber, and C. Bobda, “CuNoC: a scalable dynamic NoC for dynamically reconflgurable FPGAs,” in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL ’07)*, pp. 753–756, Amsterdam, The Netherlands, August 2007.

[21] I. Nousias and T. Arslan, “Wormhole routing with virtual channels using adaptive rate control for network-on-chip (NoC),” in *Proceedings of the 1st NASA/ESA Conference on Adaptive Hardware and Systems (AHS ’06)*, pp. 420–423, Istanbul, Turkey, June 2006.

[22] K. G. W. Goossens, et al., “Hardwired NoCs in FPGAs to unify data and configuration interconnects,” in *Proceedings of the International Symposium on NoCs*, pp. 45–54, 2008.

[23] C. Hilton and B. Nelson, “PNoC: a flexible circuit-switched NoC for FPGA-based systems,” *IEE Proceedings: Computers & Digital Techniques*, vol. 153, no. 3, pp. 181–188, 2006.

[24] S. Jovanovic, C. Tanougast, S. Weber, and C. Bobda, “CuNoC: a scalable dynamic NoC for dynamically reconflgurable FPGAs,” in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL ’07)*, pp. 753–756, Amsterdam, The Netherlands, August 2007.

[25] Silistix, “CHAINworks,” 2008, http://www.silistix.com/.

[26] Arteris, “Danube NoC IP,” 2005, http://www.arteris.com/.

[27] M. M. Khan, D. R. Lester, L. A. Plana, et al., “SpiNNaker: mapping neural networks onto a massively-parallel chip multiprocessor,” in *Proceedings of the International Joint Conference on Neural Networks (IJCNN ’08)*, pp. 2849–2856, Hong Kong, June 2008.

[28] A. D. Rast, S. Yang, M. Khan, and S. B. Furber, “Virtual synaptic interconnect using an asynchronous network-on-chip,” in *Proceedings of the International Joint Conference on Neural Networks (IJCNN ’08)*, pp. 2727–2734, Hong Kong, June 2008.

[29] B. Glackin, J. Harkin, T. M. McGinnity, and L. P. Maguire, “A hardware accelerated simulation environment for spiking neural networks,” in *Proceedings of the Applied Reconfigurable Computing Workshop*, pp. 336–341, 2009.

[30] P. A. Merolla, J. V. Arthur, B. E. Shi, and K. A. Boahen, “Expandable networks for neuromorphic chips,” *IEEE Transactions on Circuits and Systems I*, vol. 54, no. 2, pp. 301–311, 2007.

[31] J. Schemmel, et al., “Mixed-mode analog NN using current-steering synapses,” *Analog Integrated Circuits & Signal Processing*, vol. 38, 2004.

[32] G. Indiveri, “A neuromorphic VLSI device for implementing 2D selective attention systems,” *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1455–1463, 2001.

[33] J. Harkin and M. McElholm, “Novel interconnect strategy for large scale implementations of NNs,” *IEEE Soft Computing in Industrial Applications*, July 2007.

[34] Y. Chen, L. McDaid, S. Hall, and P. Kelly, “A programmable facilitating synapse device,” in *Proceedings of the International Joint Conference on Neural Networks (IJCNN ’08)*, pp. 1615–1620, Hong Kong, June 2008.

[35] Y. Chen, et al., “A silicon synapse based on a charge transfer device for SNNs,” in *Proceedings of the 3rd International Symposium on Neural Networks (ISNN ’06)*, Chengdu, China, May 2006.

[36] D. E. Johnson, J. S. Marsland, and W. Eccleston, “Neural network implementation using a single MOST per synapse,” *IEEE Transactions on Neural Networks*, vol. 6, no. 4, pp. 1008–1011, 1995.

[37] H. M. El-Bakry, “Modular neural networks for solving high complexity problems,” in *Proceedings of the International Joint Conference on Neural Networks (IJCNN ’03)*, vol. 3, pp. 2202–2207, Portland, Ore, USA, July 2003.

[38] O. Booij and H. Tat Nguyen, “A gradient descent rule for spiking neurons emitting multiple spikes,” *Information Processing Letters*, vol. 95, no. 6, pp. 552–558, 2005.

[39] P. Rocke, B. McGinley, et al., “Investigating the suitability of FPAAs for evolved hardware spiking neural networks,” in *Proceedings of the International Conference on Evolvable Systems (ICES ’08)*, pp. 118–129, 2008.

[40] G. Indiveri, E. Chicca, and R. Douglas, “A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity,” *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 211–221, 2006.

[41] J. Harkin, P. Dempster, T. M. McGinnity, and B. Cather, “Fault Detection Strategy for Self-Repairing Systems,” in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics UK Chapter (SMC ’07)*, pp. 23–28, Dublin, Ireland, September 2007.