FPGA implementation of biologically-inspired auto-associative memory

C.H. Ang, A.L. McEwan, A.van Schaik, C. Jin and P.H.W. Leong

The design of an auto-associative memory based on a spiking neural network is described. Delays rather than binary values are used to represent signals and coincidence is used to perform pattern matching. A complete implementation of the memory on a single FPGA is presented.

Introduction: In this Letter, the design and implementation of a biologically-inspired auto-associative memory, which uses the massive logic resources available in an FPGA to model axonal delay elements (DEs), are presented. As illustrated in Fig. 1, the system learns a certain input pattern through a training process. When a partial input pattern is applied, the complete version of the training pattern is retrieved and a sustained replay effected.

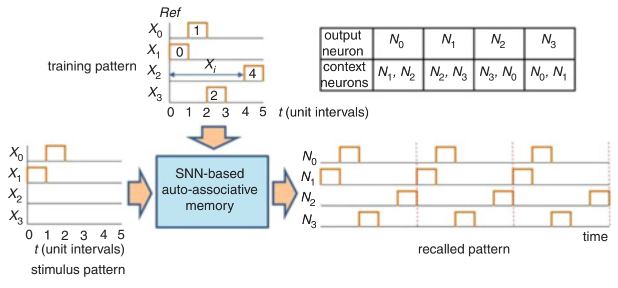

Fig. 1 *Auto-associative memory*

Architecture: We define a pattern as a sequence of pulses. We refer to the pulses as spikes and each spike has width t_{spike}. The sequence of pulses is represented by a vector of k elements $\{x_0, x_1, \ldots, x_{k-1}\}$, with each x_i representing the time between a reference signal and the rising edge of each pulse [1]. The spiking neural network (SNN) model consists of k neurons. Each neuron contains a coincidence detector driven by a subset of the other $k-1$ neurons. The input spikes to a neuron's coincidence detector are referred to as the context spikes for the given neuron [2]. The training pattern is treated with wrap-around in the time domain, i.e. the last spike precedes the first spike. If each coincidence detector receives c context spikes, then a total of $k \times c$ programmable delay lines are required, as is illustrated in Fig. 2 for the case $k = 4$, $c = 2$.

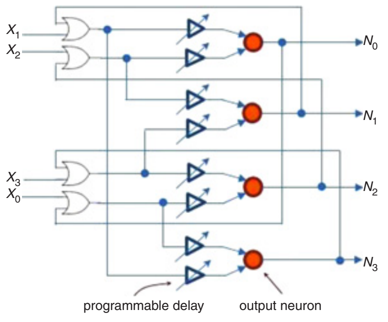

Fig. 2 *Network configuration of four-input SNN-based auto-associative memory with two contexts*

Information, i.e. patterns, is stored via programmable delay lines that interconnect the neurons. As shown in Fig. 3, each programmable delay line consists of a cascade of m DEs, where m is determined by the maximum possible delay, $d_{\max}$. The delay of each DE can be set as d_{long} or d_{short} through the configuration of the control multiplexer (mux). The long delay is an interconnection of 15 logic elements (LEs) while the short delay is a direct connection of a single wire. The total delay of an m-DE delay line, d_{line}, must satisfy $md_{short} \leq d_{line} \leq md_{long}$. Using the programmable delay lines, the SNN is configured as a feedback network that drives the recurrent activation of the stored spike pattern. The feedback connection from a neuron to a delay line introduces an additional delay of $d_{feedback}$. The total delay imposed on a spike, d_{total}, is therefore $d_{line} + d_{feedback}$. The coincidence detector of each neuron is implemented using an AND gate followed by a D flip-flop (see Fig. 3). The pulse width of each output spike is set to t_{spike} using a fixed delay element that resets the flip-flop.

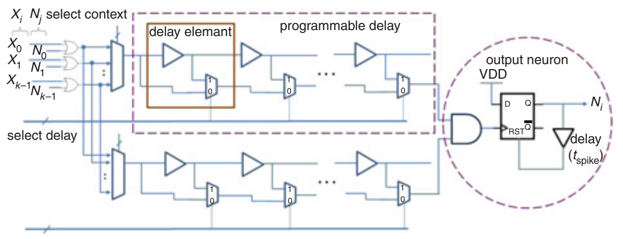

Fig. 3 *Detailed architecture of programmable delay lines and output neuron*

The training algorithm for the SNN is described in the pseudo code below and consists of two steps: establishing the correct context neuron interconnection and setting the correct delays for the programmable delay lines:

```
1. For each $N_i$ spike in the pattern,
    • Identify a set of preceding context spikes $N_j$ that triggers $N_i$
    • Make a connection from each context spike neuron $N_j$ to $N_i$ neuron
2. For each context spike $N_j$ of $N_i$ spike,
    • Adapt the delay of $N_j$ spike such that it coincides with $N_i$ spike.
```

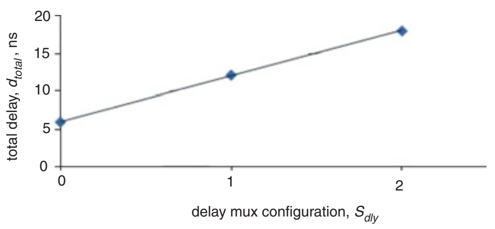

Fig. 4 *Amount of delay against different mux configurations*

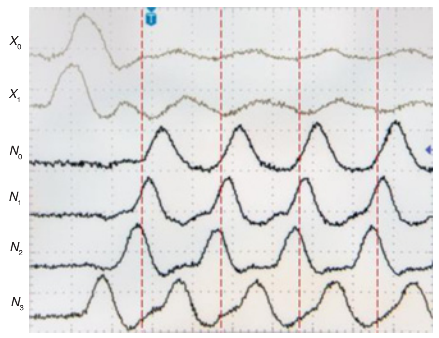

Fig. 5 *Recall of pattern {1, 0, 4, 2} captured on oscilloscope*

Voltage 5 V/div.
Time 10 ns/div.

Results: Twelve simple patterns were tested and correct operation was achieved in all instances both in simulation and in hardware. For example, for the pattern {1, 0, 4, 2} shown in Fig. 1, each spike has a 6 ns pulse width and the period of the pattern is 5 unit intervals, or 30 ns. For the N_0 spike in this pattern, the algorithm makes a 3 connection from each of the preceding context spike neurons N_1 and N_2 to the target neuron N_0 through appropriate configurations of the context muxes. The algorithm then calculates the delay between the two spikes, i.e. d_{total}, and determines the delay mux configuration, s_{dly}, for achieving this delay. The value of d_{total} against s_{dly} was determined from simulation, and the plot is shown in Fig. 4. The context spikes N_1 and N_2 were delayed by 6 ns ($s_{dly} = 0$) and 12 ns ($s_{dly} = 1$), respectively, to achieve coincidence and cause triggering of the N_0 spike. The recall of the pattern can be successfully triggered from a partial representation, e.g. $\{x_0, x_1\} = \{1, 0\}$. Fig. 5 shows the recall of a pattern captured on an oscilloscope. Note that N_3 spike was triggered first in response to the input of its context spikes x_0 and x_1.

The auto-associative memory was implemented and tested on an Altera Cyclone II EP2C35 family FPGA, which consists of 33 216 LEs, with 16 LEs per logic array block (LAB). A 400 MHz PLL-clocked counter and a set of registers are used to capture the timing of the input pattern. A Nios II soft-processor runs the training algorithm, reads the spike timings from the registers, and applies appropriate mux configurations for context neurons connections and delay settings. The auto-associative memory operates asynchronously. The entire system utilises 8% (2764/33 216) of the total LEs available on the FPGA, while the SNN takes up less than 1% (328/33 216). This implementation was intended to demonstrate the feasibility of the design and the capacity can be expanded by replicating the SNN for multiple pattern storage. With the availability of high-density FPGAs such as Stratix IV with 820k LEs, approximately 2500 patterns could be stored.

Conclusion: An FPGA implementation of a compact auto-associative memory is presented. The SNN uses only combinational logic and no sequential clocking elements; it has the potential to process patterns at very high speed and low latency, and could potentially be used for ultra-high performance digital processing designs.

© The Institution of Engineering and Technology 2012
23 November 2011
doi: 10.1049/el.2011.3651
One or more of the Figures in this Letter are available in colour online.

C.H. Ang, A.L. McEwan, C. Jin and P.H.W. Leong (*School of Electrical and Information Engineering, University of Sydney, NSW 2006, Australia*)

E-mail: chong.ang@sydney.edu.au

A.van Schaik (*Bioelectronics and Neuroscience, University of Western Sydney, NSW 2751, Australia*)

References

1 Hopfield, J.J.: ‘Pattern recognition computation using action potential timing for stimulus representation’, *Nature*, 1995, **376**, pp. 33–36
2 Wills, S.A.: ‘Computation with spiking neurons’, PhD Thesis, 2004, University of Cambridge, UK