

Design Method for Unconventional Computing

Lambert Spaanenburgh[†], Benny Åkesson[‡], Andreas Hansson[‡], and Kees Goossens[‡]

[†]Dept. of Information Technology, Lund University, Lund (Sweden)

e-mail: lambert@it.lth.se

[‡]ESAS, Philips Research, Eindhoven (The Netherlands)

e-mail: {benny.akesson, andreas.hansson, kees.goossens }@philips.com

Abstract - Network-on-Chip concepts allow moving away from the classical, centralized computer. The meaningful collaboration of computing units over a well-behaved network creates the infinite state space that underlies the Turing computer concept. The combinatorial state explosion that results from a Network-on-Chip will easily prove more valuable than the mere technological progress in memory storage, when the network nodes become small enough. Embedded Super-Computing has this unconventional goal. This paper introduces a design method for such computing concepts. It discusses how a software simulation is gradually migrated into a Network-on-Chip implementation. The approach is illustrated by the development of a Cellular Neural Network as a typical example of a well-behaved network on small, embedded nodes.

Index Terms - Cellular Neural Networks, Field programmable gate arrays, Network-on-Chip, System-on-Silicon.

I. INTRODUCTION

The architecture of the classical computer is derived from the Turing machine. The all-purpose character of this concept is based on the presence of an infinite state space. In microelectronics this takes usually the form of a central control store, realized on basis of a semiconductor memory. Technological progress has brought quantitative improvements in the size of this memory, but the plannable ease of this development documented by the ITRS RoadMap has stopped fundamentally new solutions from being introduced [1].

Steady miniaturization has gradually migrated the board of old onto the chip, a concept usually called System-on-Silicon (SoS). Each node is a computer on its own, while the combination using a Network-on-Chip (NoC) is again a computer but with a combinatorial enlarged state space. This brings the theoretical concept of the Turing machine closer to reality, assuming that the computing nodes can be kept small.

Analogue networks are small and low power, but lack the flexibility for programming. Analogue networks would be ideal for Embedded Supercomputing, but the amount of design detail usually defies the design on network level. In the past this has seemed the exclusive territory of digital systems. Admittedly, digital logic is an abstraction of an analogue reality, but strangely enough, this is hardly reflected in the conventional design process where the two serve as competing technological targets. Digital systems can be described on a high abstract level and then automatically synthesized using scalable transformations; this has gained them a distinct advantage and pushed analogue away from

many VLSI applications. With the coming of submicron technology, analogue behavioral aspects are coming to the forefront again. Then the digital implementation is maybe the end of a development process, but also an intermediate stage allowing a final analogue realization when possible.

The core of the method presented here is formed by a Network on Chip approach that enables to separate the functions from the communication between them. This will allow systematic detailing the implementation of the individual functions, while leaving the option of digital or analog realization open. Such enables the systematic development of mixed signal, unconventional computers.

Unconventional computers come in different flavors, like polymer, quantum, DNA-based, bio-, molecular, and amorphous machines [2]. This covers also the 'extended analogue computer' in the definition of Lee A. Rubel. Here, a distributed supercomputer is modeled as the collaboration of many, potentially non-digital nodes. This same principle is also mimicked in the Cellular Neural Network (CNN) as originally proposed by Chua & Yang [3]. Harrer & Nossek introduce later the Discrete-Time CNN (DT-CNN) [4]. Both digital or analogue implementations are known [5]-[6].

CNNs can be developed within a MATLAB environment, like many other signal processing systems. But the path from MATLAB model to realization has many thorns. Therefore we propose here an incremental approach that gradually increases in efficiency but can stop at any moment with a viable implementation. The method assumes a Network-on-Chip (NoC) at the heart of the network separating the numerical 'float-to-fix' scaling needs of the computing nodes from the parameter passing & synchronization demands of the inter-nodal communication.

The paper is composed as follows. In section II the background of the methodology is outlined. Then we look into two alternative NoC techniques, discussing the consequences for nodal realization. In section 0 a typical CNN development illustrates the design steps. Finally we draw some conclusions and outline some benefits of the presented approach.

II. THE EMULATION APPROACH

The overall system development procedure is depicted in Fig. 1. The intended behavior is captured by MATLAB code, a software environment that hides the communication needs while the function evaluation is given by the coding order and seemingly detached from any basic clock-like rhythm.

Double-precision floating-point numbers, the largest number container supported on general-purpose computing platforms, represent the values.

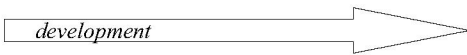
development 			
Phases	Matlab	FPGA	ASIC
Values	float	fixed	fixed
Communication	parameter passing	clocked packets	discrete-time values

Fig. 1. Moving from algorithm to hardware.

The next development stage uses a field-programmable platform, where the functions are created on basis of fixed-point values represented as arbitrary long bit-strings. An explicit network-on-chip will transfer the values between the functions. Gradually the precision of the internal number representation will be decreased. This is not only aimed to make the functional macros smaller, but also to evaluate whether a precision can be achieved that is as low as inherently coupled to analogue implementations.

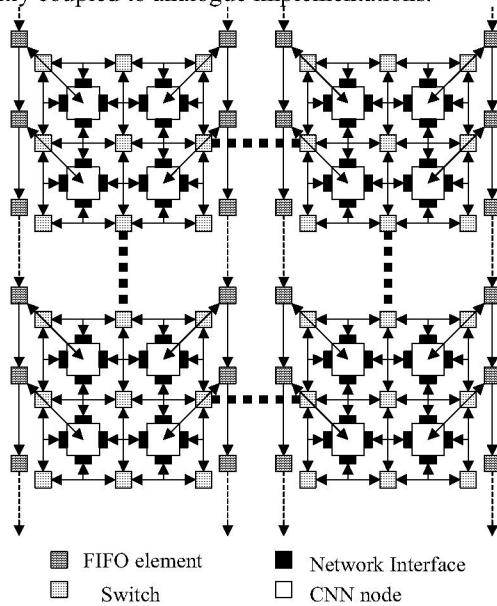


Fig. 2. The Caballero architecture uses a network of CNN nodes with a Network-on-Chip, while pixels are transported over a distributed FIFO.

The precision study can also be done in MATLAB, though at a premium in execution time. The question on the required transfer characteristics that next needs to be answered cannot be handled in MATLAB anymore. The aim is to reduce the traffic density as much as possible. This, in combination with the allowance of medium-size precision, shows that also a realization of analog function macros, embedded in a digital network-on-chip, can be afforded.

At the last stage, we further question the value transfer by the digital network that was originally introduced to allow for a smooth design flow. Having subsequently reduced the computation and the communication requirements, we may

find ourselves in the situation that a fully analog realization proves to be feasible.

Network switching is introduced on the inter-nodal communication for the following advantages. By suitable switching techniques, this can handle an arbitrary neighborhood (though at the expense of decreased precision) and provides more freedom in the network floor plan. Overlapping communication & computation: the map values (or image pixels) are sent to/from the nodal processors over a FIFO, while at the same time the nodal processors are computing and the NoC sends the results to neighboring nodes.

Essentially, a Network on a Chip, NoC, consists of switches and network interfaces, NI (Fig. 2). Network interfaces translate the view that components attached to the NoC have on communication, and the internal view switches have. By using multiple switches a NoC scales both in the number of components (such as cells) that can be attached to the NoC, and in the performance the NoC can deliver. NoCs are therefore modular scalable interconnects [7]. A switch receives data on its inputs and sends it to its outputs, taking care each output is used by only one input at any point in time. Data can be moved around a NoC by circuit and by packet switching.

In circuit switching the entire path between the two cells that communicate is claimed. The data that the source cell passes to the NoC is transported over the circuit to the destination cell without significant processing by the NIs (such as chopping in smaller pieces or packets). Circuit switching is employed in FPGAs, where lookup tables, BS-RAMs, etc. are interconnected by switches. SoCbus [8] is an example of circuit-switched NoCs for systems on a chip (SoC). However, because all links along the path are claimed for the communication, circuit switching becomes costly when high interconnectivity is required.

In packet switching, the sender NI chops data into small packets, which are independently transported by the switches. The receiver NI reconstitutes the data from the packets. The path between sender and receiver can always be shared with other communicating components, and a packet-switching NoC can serve more components than an equivalent circuit-switching NoC. NoCs are currently aimed at SoCs [9], but will be applied to FPGAs too [10].

(Packet-switched) NoCs fit well with digital implementations (or models) of CNNs because they allow an arbitrary (programmable) neighborhood of cells. Moreover, NoCs decouple the communication from computation, by which we mean that rates of computation of the individual cells may differ from each other, as well as the rate of inter-cell data transport between the cells. Hence no global notion of time or synchronization is required between the system components. The Caballero network infrastructure is a modified circuit switch mixed with a guaranteed performance packeting approach. This creates a communication freedom that is functionally related to the circuit switching in an analogue network.

III. DIFFERENT NETWORKS

The feature of local connectivity gives DT-CNNs [4] a first-hand advantage to VLSI implementation with very high speed and complexity. A fully digital implementation relies on the field-programmable gate-array (FPGA) for reasons like explicit parallelism and reconfigurability. Despite its promise, the local connectivity cannot be arbitrary, as signal delay and area usage are wiring dominated. The principal decision is therefore governed by the selection of a wiring strategy.

The number of nodes within the neighborhood r of a CNN node is given by $(2r+1) \times (2r+1)$ and increases rapidly with the increase in neighborhood size. It is clearly impossible to have a direct connection between all the nodes within a neighborhood. A staged approach is advocated in [5]. It is based on a Manhattan routing style, where the messages are passed alternately in horizontal and vertical direction (Fig. 3). Two transfers can handle a 1-neighborhood; four are required for a 2-neighborhood. In general, $2r$ transfers will handle r -neighborhood.

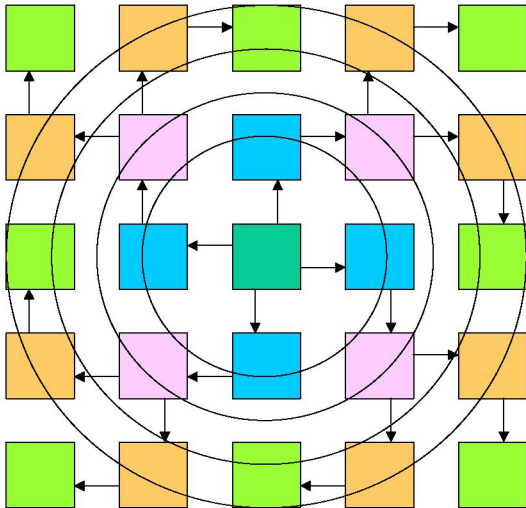


Fig. 3. Spatial communication.

In order to avoid bus conflicts, neighboring nodes cannot be admitted to be active communicating values at the same time. It appears that nodes at a 'Knight Jump' distance can be active simultaneously. Then, activity can be passed clockwise turning along a widget boundary (Fig. 4). This burdens the design with an additional activity controller and severely reduces the amount of potential parallelism. The scheme can easily be extended to larger neighborhoods.

An attractive alternative is formulated in [9]. It proposes a time-multiplexed bus, such that many high-speed serial connections can be served in parallel. The serial transfer of values over the network takes roughly the same time as it takes a node to come with a new value. Under such circumstances it does not pay to have a global network as service can hardly be guaranteed. And best effort cannot be afforded. Apparently a CNN poses a communication problem that can hardly be solved by a general-purpose NoC.

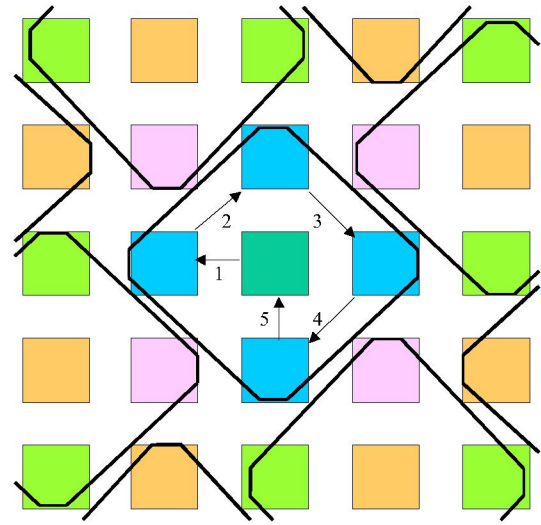


Fig. 4. Node activation for spatial communication.

In the CNN case, the local communication can be used to good fortune. Each intra-node connection is served by one serial transfer line. Consequently 1-neighborhood communication is served by a 8-wide serial network, though irregularly shaped (Fig. 5). Which shape that is, remains to be decided. The good point is that we do not have to concern ourselves with an activation pattern, as all nodes can be active simultaneously. This saves a global controller, but we will see later that it also has its drawbacks.

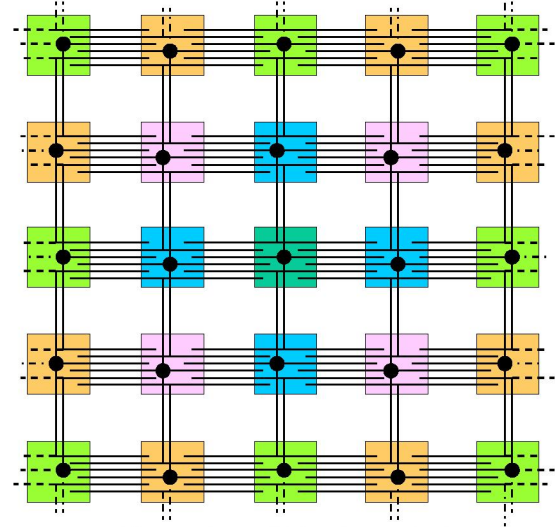


Fig. 5. Time-multiplexed communication.

The essential difference between the spatial and the temporal (i.e. time-multiplexed) scheme is that in the former the network is used by one node only to communicate an n -bit value to all its neighbors, while in the latter case all nodes communicate simultaneously their values bit-wise. The scaling properties are worth noticing. For applications where a large neighborhood r must be spanned, the temporal scheme will just use more time to communicate (latency $\sim nr$ where n means value bit width) while the spatial one will drastically grow in size (latency $\sim r^3$).

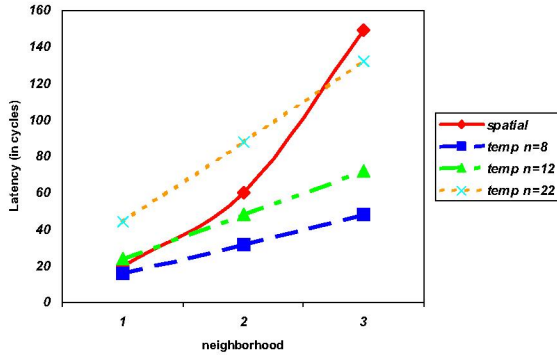


Fig. 6. Neighborhood dependent latency in network communication.

The effective outcome of the relation with neighborhood size as illustrated by Fig. 6 is complicated because it is layout-dependent. It is clearly unfeasible to have a $(2r+1) \times (2r+1)$ wide local bus. But even structured approaches as shown in Fig. 3 and Fig. 5 have their problems. One reason is the explosion in bus width for a larger neighborhood; the other is that the connectivity is local but not disjoint. This overlap may possibly translate into a further aggravation of the spatial presence, as it does not map equally in both dimensions. On the other hand, this agrees very well with the unbalance in wiring dimensions found in contemporary FPGAs.

IV. A NETWORK NODE

The local CNN node is assumed to have an ISO/OSI compliant structure as shown in Fig. 7. It is largely similar to what has been discussed in detail in [11]. The core element is a simple multiplying accumulator (MAC). It operates on data that are either shared over a longer period or currently supplied from other nodes. General constants such as CNN templates and other network initializations come in over a FIFO, while nodal values are received over the network. The Supplier will direct these values with help of the Node and Storage Controllers to their local destination in the Local Storage. The MAC will pick its data independently from this same Local Storage, while signalling the beginning of each computation cycle to the Storage Controller.

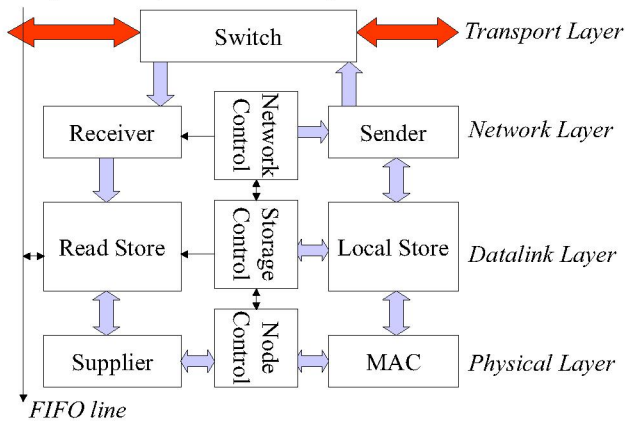


Fig. 7. Cell Overview Schematic.

The Local Storage provides the Kahn-type synchronization between the computation and the communication. It ensures that the local MAC will operate on the available data and not necessarily wait on the inspection of the entire neighbourhood. Conversely, it ensures that the communication does not have to be stalled when the nodal computation has not finished. This allows having communication and computation in different clocking regimes.

The network layer contains the Receiver and the Sender where the cell-addressing scheme for the network unit is handled. In the case of spatial communication this is the source information. From this, the visited cells during transfer can deduce how to move further. In the temporal scheme, each value corresponds with a dedicated line. This still leaves the problem of how to solve inherent series/parallel and parallel/series conversion in the presence of the dual-port RAM. Instead of redesigning the network and datalink layer, we have chosen to add shift registers in the receiver and sender, that will be respectively copied in parallel to the buffer and loaded in parallel from the buffer Fig. 8).

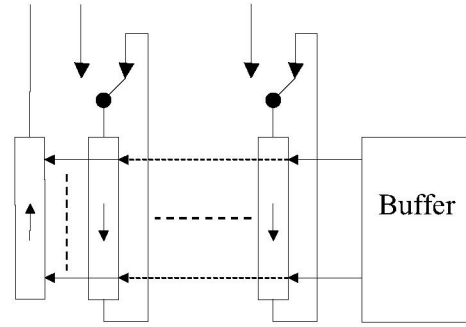


Fig. 8. S/P and P/S conversion in the Network Layer.

The scaling properties are worth noticing (Fig. 9). For applications where a large nodal accuracy is required, the temporal scheme will hardly be affected ($\text{area} \sim r$) while the spatial one will grow quadratic in size ($\text{area} \sim nr^2$ where n means value bit width).

In Fig. 5, a regular communication pattern has been shown to provide for the time-multiplexed communication between the nodes. This suggests a fixed-wiring scheme, but such lacks the required flexibility. A CNN operation is based on the multiplication of nodal values with their templates values with a correspondence that is given on their relative position in the network. This correspondence can be pre-compiled, but it is more elegant to compute this locally on basis of the source address in the message. Consequently we will need packets of source address and value also in this case.

It is unfortunate that this does not easily fit the usual FPGA architecture, but by limiting the neighbourhood we can keep the burden low. The other option is to revert to the principle of delayed activation as introduced for the spatial communication in section II. Doing that, the values will not start to flow at the same clock cycle and therefore will not

complete at the same moment. This makes that that they can be sampled from the serial shift registers in succession to be saved in the intermediate dual-port RAM. Actually this builds the bridge between the two alternatives.

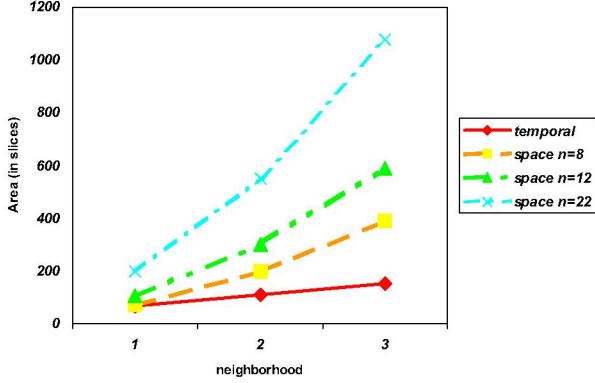


Fig. 9. Area consumption in network communication.

V. DISCUSSION

An example in smart camera design is now introduced to illustrate the methodology outlined in this paper. The principle design is captured by means of morphological expressions. This allows separating the image processing steps from the flow control, but is still within a single MATLAB program. The aim is to bring as much of the morphological expressions together in single functions as this provides a basis to generate an efficient set of CNN templates [13].

The specification of the algorithm can be optimized in terms of potential throughput. This requires a thorough analysis of the data flow in terms of the initialization needs. The end result is a morphological program described in MATLAB and therefore void from implementation data such as synchronization means and word length.

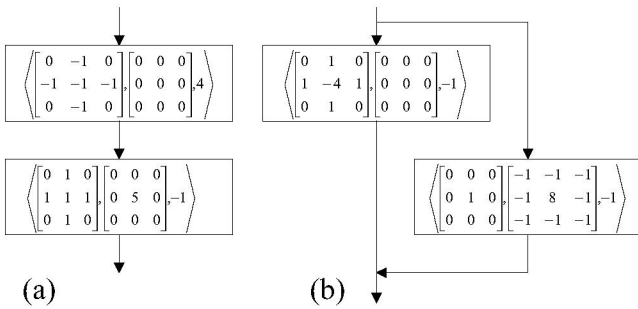


Fig. 10. Optimizing the CNN flow. (from [13])

A typical example is shown in Fig. 10 for the case of inner and outer edge detection ([13], Example 4.2). The basic morphological expression is $F = (X \cap \neg(X \oplus A)) \cup (X \cap (\neg X \oplus A))$. This seems to require 7 operations, but this can be reduced to 2 as shown in Fig. 10a. The traditional solution requires 3 operations, assuming a union of a separate inner- and outer edge extraction (Fig. 10b). The minimal solution does not only requires less

operations, but also does not need more than 1 time loading of the image.

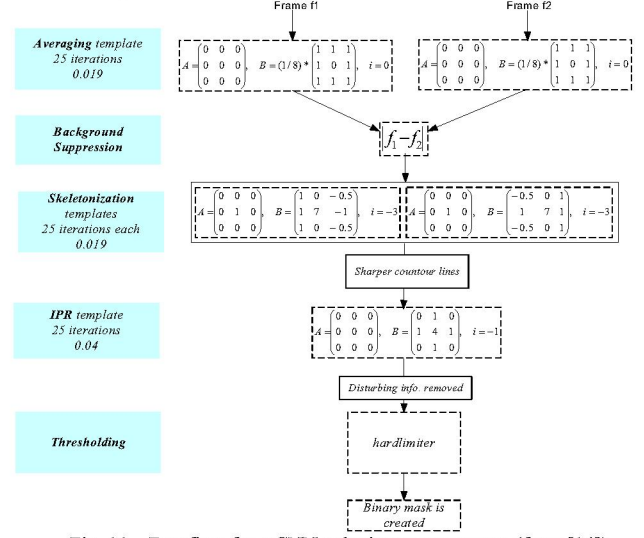


Fig. 11. Dataflow for a CNN velocity measurement. (from [14])

The conversion from a set of morphological expressions to a DT-CNN flow leads to a description based on CNN operations, which can be mapped on a Caballero platform. The MATLAB floating-point values need to be transformed to fixed-point. A spatial communication is used, as this scales best with the large word width. Based on that model, the minimal required word width for nodal and template values can be determined [15].

In the spatial communication scheme, all nodes will communicate in parallel, but the exchange of the values will be not fully parallel. Effectively, all nodes will operate on data from the same computational loop (an iteration) as expressed in the basic CNN nodal formulation

$$x^c(k) = \sum_{d \in N_r(c)} a_d^c y^d(k) + \sum_{d \in N_r(c)} b_d^c u^d + i^c \quad (1)$$

In other words, the system is iteration based, where an iteration will take a number of clocks. Therefore, the next step is to introduce a time-multiplexed communication. Here, not only all nodes are computing in parallel but also they are simultaneously communicating. It is not fully cycle-true but allows investigating this further.

As the bit streams are flowing over the serial Aethereal-like links, such data can be copied into the buffer at judicious intervals. Two policies can be pursued. The first is to cut the transmission after a number of bits. This is similar to word width reduction as has been performed on the previous communication scheme and is therefore superfluous in this context.

The alternative is to update the buffer with even incomplete words but to continue transmission. Consequently the values will be right in the next nodal iteration. In [12] an early 80's idea of Richard Newton was taken up to show how the iterative solution of a higher-order equation could be implemented on a network on computing elements (Fig. 12).

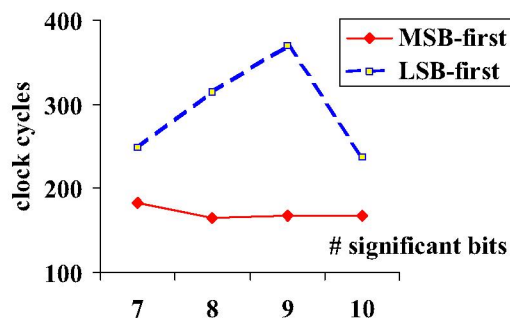


Fig. 12. Newton-Raphson approximation with limited precision using MSB-first and LSB-first arithmetic. (from [12])

Along the same principle of operation we find that convergence in a CNN will still be achieved when the dataflow is not strictly time-scheduled. This allow to have additional gain in network performance using time-multiplexed communication, when the word width has already been minimized (Fig. 13).

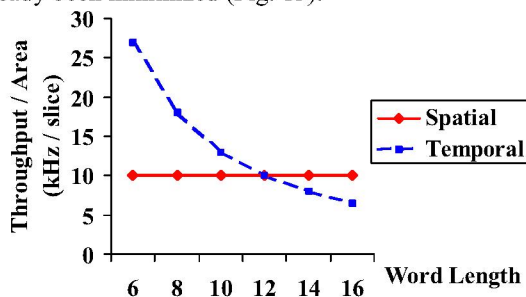


Fig. 13. Throughput Area ration dependence on internal word length for both communication schemes.

VI. CONCLUSIONS

A Rubel system means the return of the analogue computer of old where the digital programming frame is built from a Network-on-Chip. The Aethereal NoC provides excellent opportunities for the system glue in such a large-scale, distributed system. We have studied here the role of Aethereal in the systematic development of a typical Unconventional Application-Specific Integrated Processor (ASIP), a CNN.

From a morphological description of the intended operation, a MATLAB model can be derived with potentially optimal performance in terms of number of operations and amount of external memory access. This model is brought onto a first FPGA structure based on spatial communication within the network doing the CNN iterations in a fixed sequence. This allows an optimal float-to-fix conversion targeting on a minimal internal word length. Subsequently we move to a second FPGA structure based on time-multiplexed communication. This allows handling the communication in arbitrary order. The result is an impressive decrease in hardware capacity demands and increase in network operation speed.

A side benefit of the temporal communication is the compatibility with analogue cells over conventional A/D and

D/A conversion. Here we typically find a Successive Approximation A/D converter that will serially produce an MSB-first bit string. Along similar lines is the discussion about accuracy-driven arithmetic [12].

Another issue that deserves attention is the relation between the composition of complex templates that results from the optimization of morphological expression and the potentially increased size of the local cells that is left after reduction by word size and parallelization of the communication. More practical experience is required to see which of the two effects is dominant and whether a suitable compromise can be formulated.

VII. REFERENCES

- [1] J.W. Mills et alien, "Empty Space Computes: The evolution of an Unconventional Supercomputer", Tutorial at ACM Int. Conf. On Computing Frontiers, Ischia, Italy, 2006.
- [2] C. Teuscher and A. Adamatzky (eds.), "Unconventional Computing 2005: From Automata to Wetware", Luniver Presss, Beckington, England, 2005.
- [3] L. O. Chua and L. Yang, "Cellular Neural Networks: Theory", IEEE Trans. on Circuits and Systems, Vol. 35, pp. 1257-1272 and 1273-1290, 1988.
- [4] H. Harrer and J.A. Nossek, "Discrete-Time Cellular Neural Networks", Int'l Journal of Circuit Theory and Applications, Vol. 20, pp. 453-467, 1992.
- [5] S. Malki, and L. Spaanenburg, "On the packet-switched implementation of a discrete-time CNN", Euromicro Symposium on Digital System Design, pp. 234 - 241, 2004.
- [6] G. Linán et al., ACE16K, "A 128 x 128 focal plane analog processor with digital I/O", Proc. Workshop on CNNs and their Applications, R. Tetzlaff (ed.), World Scientific (Singapore), pp. 132-139, 2002.
- [7] L. Benini, G. De Micheli, "Networks on Chip: A new SoC Paradigm", IEEE Computer, Vol. 35, Nr. 1, pp. 70-80, 2002.
- [8] D. Wiklund, D. Liu, SoCBUS, "Circuit-switched Network on Chip for Hard Real Time Embedded Systems", Proc. IDPDS, 8 pgs, 2003.
- [9] K. Goossens, J. Dielissen, A. Radulescu, "The Aethereal Network on Chip: Concepts, Architectures, and Implementations", IEEE Design & Test of Computers, Vol. 22, Nr. 5, pp. 21-31, 2005.
- [10] R. Manohar and C. Kelly, "Network on a Chip: Modeling Wireless Networks with Asynchronous VLSI", IEEE Communications Magazine, Vol. 39, Nr. 11, pp. 149-155, 2001.
- [11] B. Åkesson et alien, "Sleipner", Proceedings ProRISC (Veldhoven, The Netherlands, November 2004) pp. 201-208.
- [12] R. vanDrunen et alien, "Arithmetic for Relative Accuracy", Proceedings IEEE Symp. on Computer Arithmetic, Bath, England, pp. 239-250, 1995.
- [13] M.H. terBrugge, Morphological Design of Discrete-Time Cellular Neural Networks. Rijksuniversiteit Groningen, November 2005, ISBN 90-367-2394-9, 170 pages.
- [14] S. Malki et alien, "Velocity measurement by a vision sensor", to be presented at CIMSA'06, La Coruna, Spain, July 2006.
- [15] D. Chen and B. Zhou, "Digital Emulation of Analog CNN Systems", M.Sc. Thesis, Lund University (Lund, Sweden), 2005.
- [16] M. Hänggi and G.S. Moschytz, "Analytic and VLSI Specific Design of Robust CNN Templates", Journal of VLSI Signal Processing, Vol. 23, 415-427, 1999.