Nano Communication Networks 1 (2010) 77–85

Contents lists available at ScienceDirect

Nano Communication Networks

journal homepage: www.elsevier.com/locate/nanocomnet

Programmable cellular architectures at the nanoscale

Pritish Narayanan [a], Teng Wang [b], Csaba Andras Moritz [a,*]

[a] 151 Holdsworth Way #301, University of Massachusetts Amherst, Amherst MA 01003, United States

[b] QualComm Inc. 5535 Morehouse Dr, San Diego, CA 92121, United States

ARTICLE INFO

Article history:
Received 4 June 2010
Accepted 9 July 2010
Available online 27 July 2010

Keywords:
NASICs
Cellular neural networks
Parallel architectures
Nanowires
Nanofabrics

ABSTRACT

This paper presents the first fully programmable digital cellular design for nanodevice-based computational fabrics. The system has a fully regular structure and consists of a large number of simple functional units called cells. It is programmable, based on a small number of global signals routed from supporting CMOS and associated nanoscale circuitry. The architecture may be adapted to suit a multitude of information-processing paradigms. One example is shown on a two-dimensional (2D) semiconductor nanowire fabric including corresponding circuit-level aspects. Key metrics such as the density and performance are evaluated. It is seen that this digital cellular design may be up to 22 times denser than an equivalent projected 16 nm CMOS version for image-processing applications. High performance is achieved, with megapixel-size images estimated to require only a few microseconds for processing. Possible manufacturing routes and defect tolerance aspects in the context of image-processing applications are also discussed.

1. Introduction

Reliable manufacturing of large-scale computational systems based on promising nanodevices such as semiconductor nanowires (NWs), carbon nanotubes (CNTs) and molecular devices continues to be challenging. Different fabric architectures for nanoscale systems have been proposed; see, for example, [2,13,18,8]. In all these cases, an important objective of the fabric has been to minimize the underlying manufacturing and device requirements. For example, a microprocessor design built on semiconductor nanowire grids using the NASIC (nanoscale application specific integrated circuit) fabric is shown in [13,14]. Various optimizations, enhancements and defect/fault tolerance techniques have been proposed for NASICs to reduce the manufacturability constraints and improve the performance and projected yield of these designs.

In this paper, we explore a new architecture for nanodevice-based computational fabrics called the NAnodevice-based Programmable Architecture (NAPA). We discuss NAPA in the context of the NASIC computational fabric. However, NAPA and the methodology may be equally applicable to other nanodevice-based fabrics which employ some form of two-level logic, e.g., CMOL, FPNI, NanoPLA, etc. Given constraints on customization in nanoscale systems (i.e., due to their being built at least partially with bottom-up chemical self-assembly), NAPA-like systems are a potentially promising direction for applications such as signal processing, single instruction multiple data (SIMD), and image processing on such fabrics.

The NAPA design is regular and highly parallelized, with a large number of identical functional units or cells performing a given task. Instructions to each cell are transmitted on a limited number of global signals controlled from reliable peripheral CMOS circuitry. The cells themselves are composed of a small number of nanodevices that perform simple computations and are locally interconnected. The collective behavior of a large number of interconnected cells achieves information processing.

* Corresponding author.
E-mail addresses: pnarayan@ecs.umass.edu (P. Narayanan), andras@ecs.umass.edu (C.A. Moritz).
URL: http://www.ecs.umass.edu/ece/andras (C.A. Moritz).

1878-7789/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.nancom.2010.07.003

These sorts of collective computation systems may be a 'natural fit' for nanoscale implementations because (i) high densities are available for a high level of parallelism, (ii) local interconnections with minimal global routing remove the need for complex interconnect structures and (iii) 100% fault-free components are not a requirement since some defective cells in the design may not adversely affect the overall output integrity, and this allows more aggressive manufacturing processes. However, given the high rates of defects in nanomanufacturing, built-in defect/fault tolerance techniques or reconfiguration support (such as assumed in [2,18,19]) will need to be incorporated at various levels of NAPA to ensure that a sufficiently large number of cells are functioning correctly for output integrity.

NAPA may be used to implement a variety of computational paradigms. In this paper one possible implementation based on a computational model used in a discrete-time digital cellular neural network (CNN) for image-processing tasks is discussed. Multiple image-processing tasks are possible by controlling the global instruction signals, which act as the templates for the CNN cells. Hence, the design is fully programmable and can run a variety of processing tasks in a sequence.

The main contributions of this paper are as follows. (i) A new cellular architecture for nanodevice-based digital computational fabrics is presented; (ii) a method to achieve full programmability for this architecture without any additional manufacturing requirements is proposed; and (iii) key metrics such as area and delay are evaluated; the density benefits of the nanodevice-based design over an equivalent projected 16 nm CMOS implementation are discussed. As far as we are aware, NAPA is the first proposed nanodevice-based digital cellular implementation that properly addresses programmability and physical implementation issues such as how to deliver template values to individual nanocells.

The rest of the paper is organized as follows. Section 2 provides an introduction to cellular neural networks. Section 3 discusses the new cell-based architecture. Section 4 evaluates the area and delay on a two-dimensional (2D) nanowire grid. Section 5 discusses defect tolerance aspects. Section 6 concludes the paper.

2. Cellular architectures

A cellular neural network (CNN, also known as a cellular nonlinear network) [1] is a massively parallel computing system made of identical units called cells. Each cell has an input, an output and an internal state that evolves based on certain dynamic rules called templates. Most interconnections are local, and each cell exchanges signals with its nearest neighbors. This system may be used for applications such as image processing, with the values of the templates defining the nature of the particular task being performed.

The original CNN of Chua and Yang [1] is an analog system, in which the cell is an electrical circuit implementing an analog transformation in continuous time. The input, output and state are all analog voltages. Nearest-neighbor interaction between cells is achieved by current flow; the dynamic rule of evolution of the CNN is the Kirchoff current law at the state node. Circuit-level implementations of an analog CNN are shown in [7,11].

A detailed comparison of different approaches to building CNN systems at the nanoscale is presented in [16]. In general, analog implementations of a CNN may be difficult if not impossible to realize at the nanoscale. These designs require arbitrary sizing and customization of devices and precise control of transistor operating points, and may therefore not be achievable with self-assembly-based approaches that favor the formation of regular structures and are not amenable to that degree of nanoscale customization. While the CMOL Crossnet [8] is an analog Hopfield-like neural network, it uses CMOS for computation and signal restoration. Nanowires/nanodevices are used only for interconnects.

Digital MOSFET-based implementations of the CNN include [17,3]. In digital CNN systems, the cell is a digital circuit evaluating a Boolean expression over discrete clock cycles. The input, state and outputs are binary values, and state evolution is through the execution of combinational logic followed by latching at the clock edge. Templates are stored in and retrieved from registers.

In addition to analog and digital implementations, researchers are also exploring the creation of nanoscale architectures using specialized devices. For example, [9,6] show the use of resonant tunneling diodes (RTDs) for CNN architectures. [23] explores a tunneling phase logic (TPL)-based implementation of a CNN in which the electrical phase of the tunneling phenomenon is used to encode the logic state, and nearest-neighbor interaction is achieved by capacitive coupling. While these specialized device-based approaches may possibly achieve high integration densities and fast performance, there are many unresolved issues: for example, the need to address each individual nanodevices, support for template programming, and in general the manufacturability of complete fabrics based on these devices. We therefore believe that these systems, while theoretically very interesting, may only be feasible in the longer term, if at all. By contrast, key manufacturing steps for digital systems such as NASIC have already been demonstrated [15,13]. With improvements in nanomanufacturing, digital nanoscale designs may be aggressively scaled to achieve tremendous benefits in density, performance and other key metrics.

3. The NAPA cellular architecture

3.1. Overview of NASIC

NASIC (nanoscale application specific integrated circuit) fabric is a nanoscale fabric proposed for semiconductor nanowires and targeting datapaths. NASIC designs use crossed nanowire field effect transistors (xnwFETs) on 2D semiconductor NW grids to implement logic functions.[1] NASICs are based on a cascaded two-level logic style, e.g., NOR–NOR, NAND–NAND and heterogeneous two-level (H2L) logic [21]. Microwires provide control

[1] A 3D version of NASICs is under development.

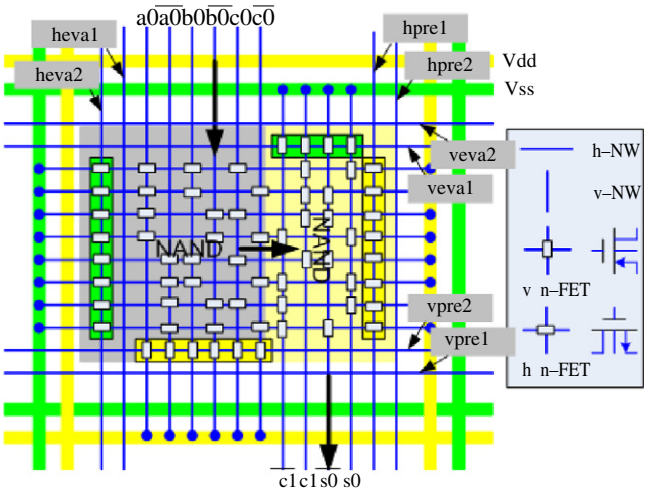

Fig. 1. NAND–NAND-based implementation of a NASIC 1-bit full adder. White boxes are n-type transistors.

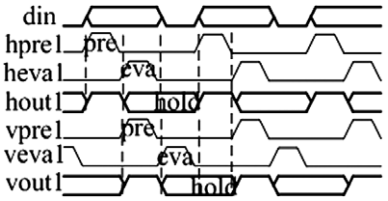

Fig. 2. Timing diagram for NAND–NAND logic showing precharge and evaluate phases of horizontal and vertical NAND planes.

signals generated from outside CMOS circuitry, with dynamic control that channels the flow of data through the nanowire tiles. NASIC circuits require only one type of FET in the logic portions of the design, simplifying the manufacturability and improving performance. NASIC designs are optimized according to specific applications to achieve higher density and on-the-fly defect/fault-masking.

Fig. 1 shows a 1-bit full adder implemented on the NASIC fabric. This tile uses a two-level cascaded NAND–NAND scheme for logic implementation. The tile comprises of n-type nanowires in the horizontal and vertical directions surrounded by a small number of microwires carrying power supplies (*VDD* and *VSS*) and control. *hpre*, *hev*, *vpre* and *veva* are signals used to dynamically control the flow of data through the tile.

The timing waveforms for dynamic control and data propagation are shown in Fig. 2. The outputs of the horizontal plane are first precharged by asserting *hpre*. *hpre* is then switched off and *heva* is switched on to evaluate the output of the horizontal stage based on the inputs. During this time the output vertical nanowires are precharged (*vpre*). The horizontal nanowires then go into *hold* phase with both *hpre* and *heva* switched off, so that the vertical plane can be evaluated (*veva*). This three-phase progression of precharge–evaluate–hold is implemented on all NASIC tiles, and it enables streaming of data through a large multi-tile system. More information about NASIC dynamic control schemes is available in [12–14].

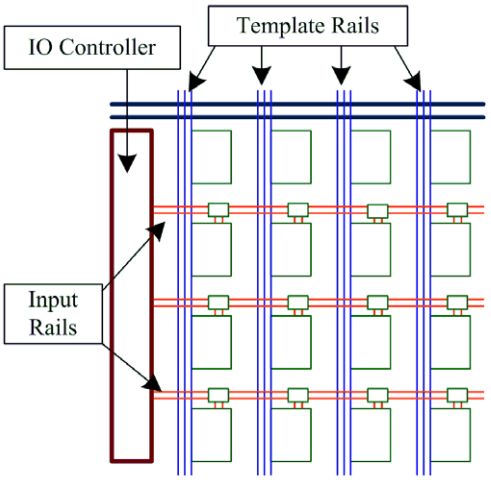

Fig. 3. The NAPA cellular architecture. Boxes are functional units.

3.2. NAPA: cellular architecture

In this section we explore a new cellular architecture called the NAnodevice-based Programmable Architecture (NAPA) in the context of NASICs. The system (Fig. 3) consists of an array of simple, locally interconnected cells, along with a limited number of global signals propagated to every cell in the design. In the specific case of a digital cellular architecture, these global signals would satisfy the role of templates. These signals are driven from peripheral CMOS circuitry, making fully programmable templates possible.

There is a CMOS input–output controller that serially loads the inputs and reads out outputs. Alternatively, systems with a focal plane array of nanosensors placed directly above the cell array may be possible. These may be achieved only with high-density nanodevice-based cells since, with CMOS technology, the area required to process a single pixel may be much larger than the size of the pixel itself [19]. However, the alignment and interfacing of

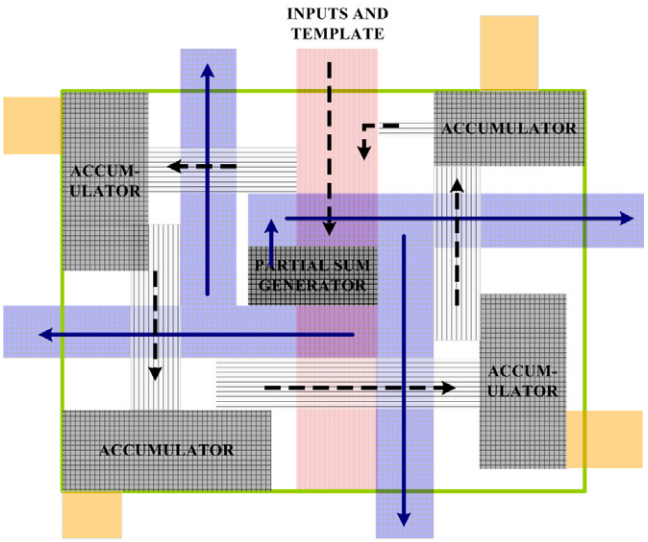

Fig. 4. Layout of a single NASIC cell in NAPA. Solid arrows (clockwise loop) are partial sums transmitted to nearest neighbors. Dashed arrows are the state accumulation datapath.

cells with nanosensors and related concerns will need to be addressed.

Each cell is a collection of NASIC tiles. Nanowires provide local interconnection between tiles in a cell as well as between neighboring cells. Each cell performs a relatively simple computation task, and the collective behavior of a large number of cells accomplishes information processing. Different applications that use cellular architectures such as SIMD, DSP, and CNN may be mapped to a NAPA design. The architecture and capabilities provided in a single cell determine the application, while the framework for programmability and input–output operations is unchanged.

The NAPA system discussed in this paper is based on a computation model similar to digital CNN designs proposed in [17]. However, it achieves this model on a 2D nanowire-based fabric with programmable templates and built-in fault tolerance. The state, templates and input may all be multi-bit values. The output of every iteration is the MSB of the state term. The task performed in each cell is state evolution, through the generation of partial sums, followed by the accumulation of contributions from neighbors. The expression for state evolution of any cell is given by

$$x(n+1) = \sum_{i \in N} a_i y_i(n) + b_i u_i + C$$

where each a_i and b_i are instantaneous template values, $y_i(n)$ and u_i are the outputs and inputs to a particular cell, $x(n+1)$ is the state and C is a constant term. The term $a_i y_i(n) + b_i u_i$ represents the ith partial sum.

The layout diagram of a single NAPA cell is shown in Fig. 4. There is a NASIC tile at the center for generating partial sums and state accumulator tiles at the corners of each cell. The execution inside a cell progresses through the following stages.

1. Generation of partial sums. Initially, the control circuitry for the state accumulators is off and the partial sum generator is on. Each cell then generates five partial sums in sequence, one per nearest neighbor, and one for itself. These are transmitted on the broadcast network (blue clockwise loop).
2. State accumulation. Once all partial sums are available, state accumulation datapaths are activated (anticlockwise loop). This is enabled by switching off the control circuitry for the partial sum and switching on the control for the state accumulator tiles. During this stage, the broadcast networks may be envisioned to be in an extended hold phase, with no precharge–evaluate operations. Thus, by suitably triggering the dynamic control circuitry, selective activation of datapaths and propagation of signals is achieved on a NASIC fabric without the need for arbitrary routing or additional multiplexers.
3. After all partial sums from neighbors have been accumulated, the new value of the output is calculated and the next partial sum generation cycle will begin. The CNN is mathematically proven to converge [1]; after a specified number of iterations, the final output may be readout using the IO controller.

Fig. 5 shows the circuit-level implementation of the CNN cell. White boxes at certain crosspoints represent n-type transistors. The long nanowires through the center of the cell are for templates programmable from outside the cell array. While a two-level logic implementation (NAND–NAND) is used in these circuits, at some points along the state accumulation datapath an additional routing stage is added for turning a signal in a perpendicular direction. The additional stage is needed since arbitrary routing may not be possible on a semiconductor nanowire grid-based system. Arbitrary routing would require nanowire crosspoints that function as T-junctions (which to our knowledge have not been demonstrated) or programmable switches (which increases the manufacturing requirements) rather than as transistors.

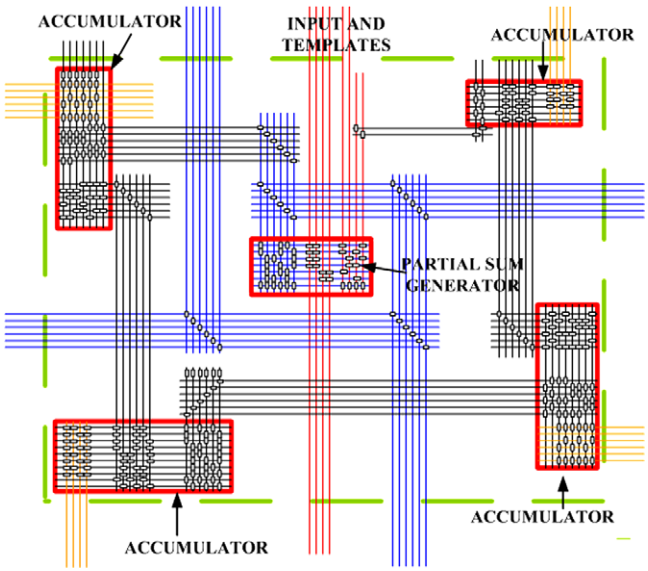

Fig. 5. Circuit-level implementation of a digital CNN cell.

3.3. Implementation of programmable templates

A key feature of the NAPA design is achieving programmability on a 2D grid without introducing new manufacturing requirements. The values on the template rails determine the nature of the image-processing task being performed. To achieve a multitude of tasks it is necessary for these templates to be fully programmable. In NAPA, this is achieved using circuits such as shown in Fig. 6. For illustrative purposes, 1-bit '*a*' and '*b*' templates are shown. The approach is easily scalable to more realistic multiple-bit implementations, using a larger number of micro/nanowires. Microwires (bold lines) act as gates for dynamically controlled n-type semiconductor nanowires. The nanowires are initially precharged to a supply value (*VDD* or *VSS*). During evaluation (*eva* signal on), the output is calculated based on the input value, similar to the functioning of any NASIC stage. The values on the template nanowires are then in *hold* phase, during which time a partial sum is calculated in every cell of the network.

Fig. 6 shows one set of template generation microwires at the top of the design. In practice, multiple sets of template microwires may be required for large array sizes. The periodicity will primarily depend on (a) the ability to reliably manufacture long nanowires without breakage and (b) the performance impact of driving these long nanowires.

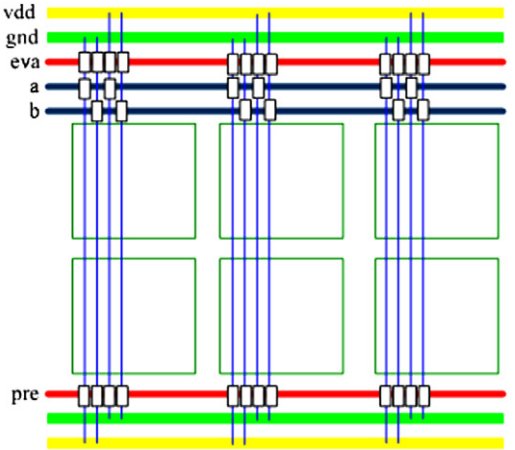

Fig. 6. Implementation of programmable template rails for NAPA. Thick lines represent microwires, thin lines are nanowires.

4. Area and performance evaluation

Potential benefits in density and performance are among the chief motivations for exploration of nanoscale circuits and systems. This section presents a detailed evaluation of these metrics for the NAPA cellular design. Area comparisons against an equivalent projected 16 nm CMOS are also shown.

4.1. Area evaluation

A functional description of an equivalent CMOS-based digital CNN cell was written in Verilog HDL. A 5×5 array of identical interconnected cells was built and synthesized to 180 nm TSMC library using a Synopsys design compiler. The layout and routing optimization with six levels of metal were then carried out to create an equivalent CMOS implementation of the digital CNN at 180 nm. The area of the design was then quadratically scaled to obtain area numbers for aggressive end-of-the-line CMOS technology nodes.

Fig. 7 shows the relative density of a 5×5 NASIC-based design against a design with projected CMOS technology nodes. A 10 nm pitch for semiconductor nanowires is assumed. This may be achievable using *ex situ* nanowire alignment techniques such as Langmuir–Blodgett [5], soft-lithography [4] or crystallographic pattern and etch or

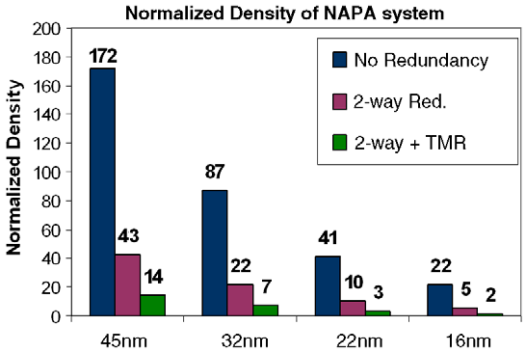

Fig. 7. Comparison of areas of NAPA-based digital CNN implementations against equivalent CMOS implementations at different technology nodes.

diblock copolymer-based catalyst patterning [20] and *in situ* growth and alignment.[2] In general, these results are consistent with previous density evaluations of NASIC microprocessors. It is seen that a NASIC-based cellular architecture would be 22 times denser than a CMOS implementation at 16 nm. However, due to the high defect rates prevalent in nanomanufacturing, some redundancy needs to be incorporated into NASIC designs. A defect rate of up to 15% is assumed for NASICs.

Fig. 7 shows two redundancy schemes based on simple two-way redundancy and triple modulo redundancy (TMR). At 16 nm, with both these techniques incorporated, the density advantage of a NASIC digital CNN is 2 times higher compared to a completely fault-free CMOS implementation.

However, this number is most likely a conservative estimate because: (i) a CMOS design at 16 nm would likely also require some degree of redundancy, (ii) recent advances in NASICs such as heterogeneous two-level (H2L) logic [21] in conjunction with nanoscale voting and error correction techniques that are merged into the logic would improve density and yield further (we are currently working on exploring all these effects), and (iii) if a fault-free 16 nm CMOS process is available, it could be leveraged in some steps of the NASIC manufacturing process, for example, in the creation of a 3D interconnect structure that may save routing area on the grid, reducing the need for as much defect tolerance as assumed here. Based on previous work with NASIC defect-tolerant processors, it is likely therefore that the density improvement will be in the 5–10 times range over CMOS designs assuming the above considerations and a defect rate less than 10% for NASIC fabric. More information on NASIC defect tolerance techniques is available in [13,21].

4.2. Delay estimation for a single digital NAPA cell

The delays of various segments/datapaths of the NAPA cellular design were estimated. All estimations are based on the NAND–NAND logic scheme, which has been shown to be faster than functionally equivalent AND–OR, NOR–NOR schemes. The NAND–NAND scheme uses only n-type nanowire transistors [14].

A nanowire pitch of 10 nm, an oxide layer thickness of 1 nm, and a dielectric constant of 2.2 were assumed. The nanowire transistor length is 5 nm and the width is 4 nm. The ON resistance for these geometries for n-type devices (R_{ON}) has been calculated as 3.75 kΩ based on experimental work reported in [10]. Interconnects are created using a nickel-based metallization process, and the resistivity of the NiSi thus formed is assumed to be 10^{-7} Ωm [22]. A lumped RC model is used for the worst-case delay analysis. Capacitances considered include NW–NW junction capacitances and relatively realistic coupling scenarios based on NASIC logic behavior and control schemes. The coupling capacitance per unit length was found to be 39.04 pF/m. The junction capacitance was found to be 0.652 aF. Table 1 summarizes all parameters.

Table 1
Parameter values for delay calculations.

NW pitch	10 nm
NW shell thickness (t_{sh})	1 nm
NW width (w)	4 nm
Dielectric constant of SiO_2 (ε_r)	2.2
Resistivity of NiSi (ρ_{NiSi})	10^{-7} Ω m
nFET ON resistance (R_{ON})	3.75 kΩ

Table 2
Largest single-phase delay for a single-cell datapath (picoseconds).

Load input	0.172
Template rails	0.249
Partial sum and broadcast	0.733
Accumulate	0.675
Output readout	0.172

In NAPA, multiple datapaths exist. The data flow in this system is summarized in the flow diagram in Fig. 8. Rectangular boxes represent datapaths in the design. Note that the decision boxes are shown only for the sake of clarity and are not decision circuits implemented in the nanowire grid. Different NASIC datapaths are exercised purely based on the control signals coming from external CMOS. As seen from the flow diagram, all datapaths other than input load and output readout are exercised multiple times. As shown in the flow diagram, multiple partial sums are generated (inner loop of Fig. 8) and a single accumulation is done. This process is iterated 100 times (outer loop of Fig. 8), which is considered sufficient for the output to converge.

The maximum single-phase delay is defined as the maximum delay experienced for any one phase (precharge/evaluate) of the dynamic circuit scheme in the design. Table 2 shows the maximum single-phase delay for the different datapaths in a single cell. In all cases, this value corresponds to the evaluate phase with the maximum number of transistors. This is because the transistor's ON resistance is the dominant factor in the calculation of delays. The largest single-phase delay occurs for the partial sum generation and broadcast datapath. Input load and output readout are exercised only once each in the cycle and are controlled independently from the IO controller. Thus, we could have two independent clocking schemes for the system, one for IO and the other for template and

[2] A variety of these nanowire growth and alignment techniques for NASIC crossbars are currently being pursued by various research groups.

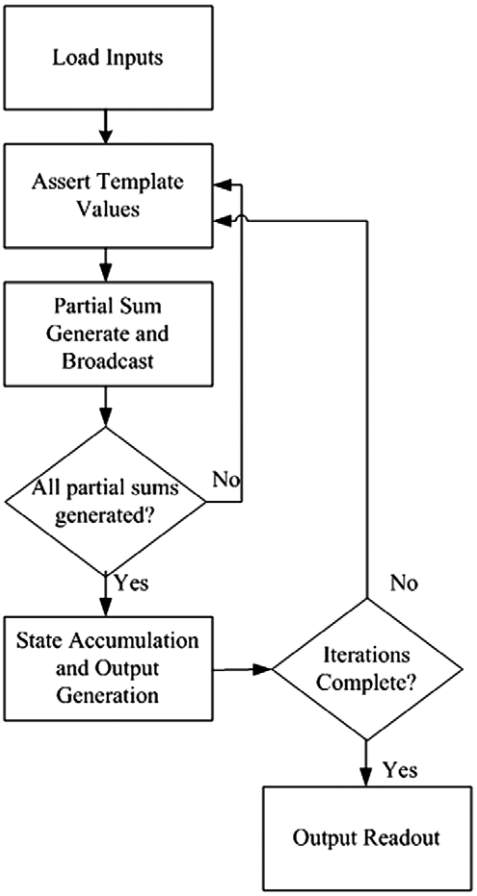

Fig. 8. Flowchart showing the dataflow in the NASIC cellular design.

cell operations. Note that the template delay in Table 2 is the delay for switching a template nanowire whose length is equal to the vertical length of a single cell. In a large-scale array, this number would have to be multiplied by the number of vertical cells in the array.

4.3. Delay estimation for NAPA arrays

Table 3 shows the NAPA datapath delays for different array sizes. The sizes are chosen based on standard image resolutions, since each cell of NAPA processes one image pixel. Input/output delay refers to the total delay in propagating the inputs to all cells. This is the same as the time taken to retrieve the final output from each cell.

Template delay refers to the time taken to drive a long unbroken nanowire whose length is equal to the vertical dimension of the nanoarray. This value, being larger than the partial sum and accumulate phases in a single cell, becomes the maximum single-phase delay in the design. 'Output generation' is an integer multiple of this number, which depends on the number of phases (from the first precharge of template to the generation of output). One hundred iterations are carried out in the cell (for output convergence) before the final output readout. The total delay is therefore the sum of this number and twice the input delay (because one input and one output operation are carried out).

Due to the multiple iterations involved in the other steps, the input–output delay component is relatively small. The total time required including input load, processing and output readout is around one microsecond, which implies ultra-fast processing of megapixel-size images. Initial evaluations show that addition of defect/fault tolerance/redundancy schemes to NASICs may increase delays by up to 5 times without fan-in/out management, i.e. the processing time for a 1920 $\times$ 1600 NAPA will be in the range of 5–6 μs, which is still very fast. However, the actual impact will be much lower assuming that fan-in/out will be constrained (e.g., by splitting the large tiles into a sequence of smaller tiles). Therefore, the NAPA system is expected to easily meet the constraints for processing real-time video streams, where the frame rate usually is less than 100 Hz, even after the addition of defect tolerance.

5. Defect tolerance in NAPA

Permanent defects may be introduced into the NASIC design during manufacturing. The defect model assumed for NASIC designs is fairly generic and assumes three types of permanent defect: the nanowires may be broken; and transistors may be stuck short, or stuck open. Transient faults due to soft-error, process variation, etc., may also be possible and are considered. Various built-in defect/fault tolerance techniques have been proposed for NASICs and they have been shown to significantly improve the yield of microprocessor designs. These techniques (such as two-way redundancy, circuit-level error correction, nanowire interleaving and nanoscale voting), use additional circuitry to mask defects/faults in the design on-the-fly and can be applied to the NAPA system discussed in this paper without limitations.

The key difference for the NAPA system lies in the *amount* of tolerance that is needed. In a general-purpose microprocessor application, enough tolerance must be built in to ensure that the correct output is obtained for all possible input combinations. In an image-processing system the requirements are more relaxed, given that human cognitive ability is limited and some incorrectly processed pixels may not significantly alter the overall output. Consequently, it is sufficient if the built-in defect/fault tolerance techniques ensure that a significant percentage of cells in NAPA produce the correct output. This is illustrated in the following analysis.

We assume to a first order that each cell has a constant probability P_{cell} of producing a correct output, and that cells may be faulty independently of one another. For a cell array of size $M \times N$, let P_{array} (100%) denote the probability that all cells simultaneously produce the correct output.

Then, if the array size is 640 $\times$ 480, and $P_{cell} = 0.999$, P_{array} (100%) $\sim$ 0, i.e. even with very high output correctness probability for individual cells, it may not be possible to get correct outputs on every cell in the array. Only when $P_{cell} = 1$, i.e. only when all the cells in the design are guaranteed to produce a correct output, will we get $P_{array}(100\%) = 1$. This scenario is not likely to be achieved

Table 3
Total delay estimation for NAPA array (nanoseconds).

Picture size	1024 × 768	1280 × 1024	1600 × 1200	1920 × 1600
Input/output	0.704	0.880	0.110	0.132
Template	0.191	0.255	0.299	0.399
Output generation	5.36	7.14	8.37	11.16
Multiple iterations	536	714	837	1116
Total delay	537	716	839	1119

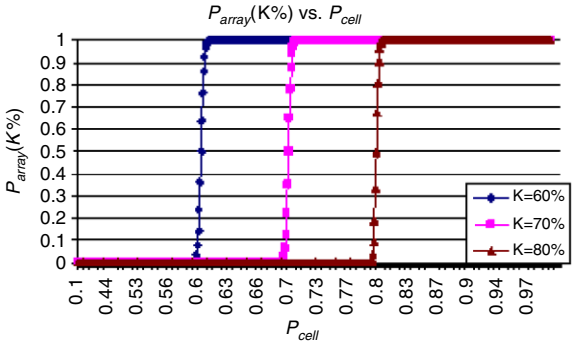

Fig. 9. Probability that at least 'K%' of cells produce a correct output as a function of P_{cell}.

in nanofabrics with device-level defect rates of a few per cent.

Now consider a certain image-processing application where at least 'K%' of individual cells are required to produce a correct output for overall output integrity. Let P_{array} (K%) represent the probability that at least K% of the cells in the design produce a correct output. Ideally, we would like P_{array} (K%) $\sim$ 1.

Based on our first order assumptions, P_{array} (K%) follows a binomial distribution with MxN independent cells, each with a constant probability P_{cell} of producing a correct output. For large values of $M \times N$ the distribution converges to the normal distribution based on the central limit theorem. Therefore, P_{array} (K%) can be determined by the cumulative distribution function of the normal. Fig. 9 plots P_{array} (K%) as a function of P_{cell} for different values of K. For example, when $K = 70\%$ (pink curve), P_{cell} needs to be only slightly larger than 0.7 for P_{array} (70%) $\sim$ 1.

In other words, overall output integrity may be achieved so long as each individual cell has a slightly greater than 0.7 probability of producing a correct output. Tolerance may be built in to achieve this requirement. Thus, the nature of the application relaxes the requirements on the NASIC defect/fault tolerance schemes. Similar trends are seen for K = 60% and K = 80%.

The appropriate 'K%' value may depend on the specific application and size of the design. Also, the above calculation treats cells independently, and does not consider neighborhood interactions. For some applications, depending on the template scaling factors, faulty outputs may propagate to neighboring cells. These considerations are presently being evaluated using mathematical analyses and detailed simulations.

6. Conclusions

NAPA, a massively parallel, template-programmable architecture for nanodevice-based computational fabrics, was discussed. An implementation of a fully programmable cellular architecture for image processing was shown, and density and delays computed. The density of the design was compared with a digital CNN implementation in CMOS projected at 16 nm. The NAPA design achieves a 22 times density advantage. Even at 15% defect rate, NAPA, modified to include considerable redundancy, is estimated to achieve a 2 times density improvement compared to a fault-free 16 nm CMOS design without any redundancy assumed. It is projected that for 5–10% defect rates the NASIC cellular architecture could be an order of magnitude denser than future CMOS and is capable of processing megapixel-size images within a few microseconds. Furthermore, it can enable other applications beyond image processing. For example, we are currently exploring other computational systems, such as based on single instruction multiple data (SIMD) and digital signal processing (DSP). While NAPA was discussed in the context of NASIC, the methodology and overall design can be generalized to implement similar architectures in other nanofabrics such as CMOL and NanoPLA. To our knowledge, NAPA is the first proposed digital cellular architecture with programmable templates on nanodevice-based fabrics.

Acknowledgements

This work was supported in part by awards from the Center for Hierarchical Manufacturing (CHM), and NSF awards CCR:0105516, NER:0508382, and CCR:0541066. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] L. Chua, L. Yang, Cellular neural networks: theory, Circuits and Systems, IEEE Transactions 35 (1988) 1257–1272.
[2] A. Dehon, Nanowire-based programmable architectures, Journal of Emerging Technologies in Computing Systems 1 (2005) 109–162.
[3] M.-D. Doan, M. Glesner, R. Chakrabaty, M. Heidenreich, S. Cheung, Realisation of a digital cellular neural network for image processing, in: Cellular Neural Networks and their Applications, 1994, CNNA-94, Proceedings of the Third IEEE International Workshop, pp. 85 –90.
[4] B.D. Gates, Q. Xu, J.C. Love, D.B. Wolfe, G.M. Whitesides, Unconventional nanofabrication, Annual Review of Materials Research 34 (2004) 339–372.
[5] Y. Huang, X. Duan, Q. Wei, C.M. Lieber, Directed assembly of One-Dimensional nanostructures into functional networks, Science 291 (2001) 630–633.
[6] A. Khitun, K.L. Wang, Cellular nonlinear network based on semiconductor tunneling nanostructure, IEEE Transactions on Electron Devices 52 (2005) 183–189.
[7] P. Kinget, M. Steyaert, A programmable analog cellular neural network cmos chip for high speed image processing, Solid State Circuits, IEEE Journal (1995) 235–243.
[8] K. Likharev, D. Strukov, CMOL: devices, circuits, and architectures, in: Introducing Molecular Electronics, 2005, pp. 447–477.

[9] S.-R. Li, P. Mazumdar, L. Chua, On the implementation of rtd based cnns, in: Proceedings of the International Symposium on Circuits and Systems 2004 (ISCAS'04), pp. 25–28.
[10] W. Lu, C.M. Lieber, Semiconductor nanowires, Journal of Physics D: Applied Physics 39 (2006) R387–R406.
[11] M. Milanova, P. Almeida, J. Okamoto, M. Simes, Applications of cellular neural networks for shape from shading problem, in: Machine Learning and Data Mining in Pattern Recognition, 1999, pp. 51–63.
[12] C.A. Moritz, T. Wang, Latching on the wire and pipelining in nanoscale designs, IN NSC-3 (2004) 1—7.
[13] C. Moritz, T. Wang, P. Narayanan, M. Leuchtenburg, Y. Guo, C. Dezan, M. Bennaser, Fault-tolerant nanoscale processors on semiconductor nanowire grids, Circuits and Systems I: Regular Papers, IEEE Transactions 54 (2007) 2422–2437.
[14] P. Narayanan, M. Leuchtenburg, T. Wang, C.A. Moritz, CMOS control enabled Single-Type FET NASIC, in: Proceedings of the 2008 IEEE Computer Society Annual Symposium on VLSI, IEEE Computer Society, 2008, pp. 191–196.
[15] P. Narayanan, K.W. Park, C.O. Chui, C. Moritz, Manufacturing pathway and associated challenges for nanoscale computational systems, in: Nanotechnology, 2009. IEEE-NANO 2009, 9th IEEE Conference, pp. 119 –122.
[16] P. Narayanan, T. Wang, M. Leuchtenburg, C. Moritz, Comparison of analog and digital nanosystems: issues for the nano-architect, in: Proceedings of the 2nd IEEE International Nanoelectronics Conference, pp. 1003–1008.
[17] M. Sindhwani, T. Srikanthan, K.V. Asari, VLSI efficient discrete-time cellular neural network processor, in: IEE Proceedings — Circuits, Devices and Systems, vol. 149, 2002, pp. 167–171.
[18] G.S. Snider, R.S. Williams, Nano/CMOS architectures using a field-programmable nanowire interconnect, Nanotechnology 18 (2007) 035204.
[19] D.B. Strukov, K.K. Likharev, Reconfigurable hybrid CMOS/Nanodevice circuits for image processing, IEEE Transactions on Nanotechnology 6 (2007) 696–710.
[20] T. Thurn-Albrecht, J. Schotter, G.A. Kastle, N. Emley, T. Shibauchi, L. Krusin-Elbaum, K. Guarini, C.T. Black, M.T. Tuominen, T.P. Russell, Ultrahigh-Density nanowire arrays grown in Self-Assembled diblock copolymer templates, Science 290 (2000) 2126–2129.
[21] T. Wang, P. Narayanan, C.A. Moritz, Heterogeneous Two-Level logic and its density and fault tolerance implications in nanoscale fabrics, IEEE Transactions on Nanotechnology 8 (2009) 22–30.
[22] Y. Wu, J. Xiang, C. Yang, W. Lu, C.M. Lieber, Single-crystal metallic nanowires and metal/semiconductor nanowire heterostructures, Nature 430 (2004) 61–65.
[23] T. Yang, R.A. Kiehl, L.O. Chua, Tunneling phase logic cellular nonlinear networks, International Journal of J. Bifurcation and Chaos (2001) 2895–2911.

Pritish Narayanan received his B.E. (honors) degree in Electrical and Electronic Engineering and his M.Sc. (honors) degree in Chemistry from the Birla Institute of Technology and Science, Pilani, India. He is currently working toward a Ph.D. degree in Electrical and Computer Engineering at the University of Massachusetts, Amherst. Currently, he is a Research Assistant with the Department of Electrical and Computer Engineering, University of Massachusetts. He was previously employed as a Research and Development Engineer at IBM, where he worked on process variation and statistical timing analysis. His research interests include nanocomputing fabrics, computer architecture, and VLSI. Mr. Narayanan was a recipient of the Best Paper Award at ISVLSI 2009. He has served as a reviewer for IEEE Transactions on Very Large Scale Integration (VLSI) Systems and the IEEE Transactions on Nanotechnology.

Teng Wang received the B.S. degree in electronic engineering and information science from the University of Science and Technology of China (USTC), Hefei, China and the M.S. degree from the Chinese Academy of Sciences in 1999 and 2002, respectively. He received the Ph.D. degree in electrical and computer engineering at University of Massachusetts, Amherst in 2009 and is currently with QualComm, Inc. His research interests include nanoscale systems, computer architecture, and VLSI design.

Csaba Andras Moritz received his Ph.D. degree in computer systems from the Royal Institute of Technology, Stockholm, Sweden, in 1998. From 1997 to 2000, he was a Research Scientist at the Laboratory for Computer Science, Massachusetts Institute of Technology (MIT), Cambridge. He has consulted for several technology companies in Scandinavia and held industrial positions ranging from CEO, to CTO, and to founder. His most recent startup company, BlueRISC Inc., develops security microprocessors and hardware-assisted security solutions. He is currently a Professor at the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst. His research interests include computer architecture, compilers, low power design, security, and nanoscale systems.