

Image Processing with CMOL Circuits

Dmitri B. Strukov^{*,†} and Konstantin K. Likharev^{*}

^{*}Stony Brook University, Stony Brook, NY 11794-3800, U.S.A.

[†]Hewlett Packard Laboratories, Palo Alto, CA 94304-1123, U.S.A.

strukov@gmail.com, klicharev@notes.cc.sunysb.edu

ABSTRACT

We have analyzed two options of using hybrid CMOS/nanodevice circuits with area-distributed (“CMOL”) interface for the low-level image processing tasks, on the simplest example of 2D image convolution with a sizable filter window. The first option is to use digital, DSP-like circuits based on a reconfigurable CMOL fabric, while the second one is based on mixed-signal CMOL circuits with the analog presentation of input and output data and the binary presentation of the filter function. Estimates of the circuit performance have been carried out for the 45-nm CMOS technology and the 4.5-nm nanowire half-pitch, and the power consumption fixed at a manageable level. In the digital case, the circuit area per pixel is about $25 \times 25 \mu\text{m}^2$, and the time necessary for convolving a $1,024 \times 1,024$ -pixel, 12-bit-accurate image with a 32×32 -pixel window function of similar accuracy is close to $25 \mu\text{s}$, much shorter than that estimated for purely CMOS circuits with the same minimum feature size. For the mixed-signal CMOL circuit, the corresponding numbers are much better ($\sim 1 \mu\text{m}^2$ and $1 \mu\text{s}$, respectively), but this option requires a very high ($\sim 1\%$) reproducibility of ON currents of the necessary crosspoint devices (programmable diodes), which has not yet been reached experimentally.

1. INTRODUCTION

The rapid progress of focal plane arrays of sensors in such metrics as resolution, frame rate, and dynamic range, as well as the emergence of new urgent needs (e.g., online face recognition), make high-performance image processing a more urgent task than ever. In this context, it is frequently necessary to perform low-level tasks, such as spatial filtering, edge detection, feature extraction, etc., in parallel on all pixels in the image [1], because sequential processing of pixels would be too slow for some applications. However, the parallel processing of pixels requires not only computation-rich hardware but also extremely large aggregate bandwidth of data transfer between the processor and memory subsystems. For example, the first step in hyperspectral imaging [2] for a realistic 12-bit 1024×1024 pixel array with 200 spectral bands requires a processing throughput of $\sim 10^{14}$ operations per second (100 Tops) and an aggregate data bandwidth of $\sim 10^{11}$ bits per second (100 Gbps) [3].

Because of these constraints, general purpose processors are not suited well for low-level image processing. (Even the latest multi-core Cell processor [4, 5], which has been specifically designed for image processing tasks and features a very impressive peak performance of $\sim 2 \times 10^{11}$ add-multiply operations with 32-bit operands per second, falls far short of the prospective needs.) Instead, the most promising computational platforms in this field are digital signal processors (DSP) [6, 7], field programmable gate arrays (FPGA) [8, 9], and most notably, focal-plane processor arrays with digital [10, 11] or analog [12, 13] cores. The basic idea of focal plane

processor arrays is to integrate the signal processing circuitry with sensor devices, so that all the data from sensors are processed locally, with no resources wasted for long-distance data transfer. A major problem with this approach is that for most complex operations (e.g., digital filtering with large window size) the area necessary for processing data from one pixel becomes much larger than the pixel sensor itself. For example, the one-pixel cell of the analog processor described in Ref. 12 was as large as $\sim 100 \times 100 \mu\text{m}^2$, or $\sim 3 \times 10^5$ lithographic squares (F_{CMOS})². This handicap is especially painful now when the conventional CMOS technology is approaching its scaling limits [14, 15].

A possible remedy to this problem might be reconfigurable hybrid CMOS/nanodevice circuits - for reviews, see Refs. 15–19. In such a circuit, a CMOS subsystem with relatively large silicon transistors would be used for signal restoration, long-range communications, input/output functions, and testing/bootstrapping, while an add-on nanowire crossbar with simple (two-terminal) nanodevices at each crosspoint would provide most of information storage and short-range communications, and also facilitate signal processing. This concept has got a strong boost from the recent experimental demonstration [20] of reproducible metal-oxide devices with the functionality (of the programmable diode) necessary for crosspoint nanodevices of the hybrid circuit. Nanowire crossbars with half-pitch F_{nano} down to 15 nm have also been demonstrated [21, 22], and there are good prospects of scaling F_{nano} down to a few nanometers using such advanced patterning techniques as nanoimprint [23] or EUV interference lithography [24].

One challenge faced by the hybrid circuit concept had been the CMOS-to-crossbar interfacing, especially because the high-resolution patterning techniques [23, 24] do not offer an equally high layer alignment accuracy. Recently our group suggested [15, 19, 25–28] a solution to this problem using an area-distributed “CMOL” interface using conic-shaped vias (“pins”) spaced by the CMOS-scale pitch but having nanoscale-sharp tips (Fig. 1). Pins of both types (reaching to the lower and upper nanowire levels) are arranged in square arrays with side $2\beta F_{\text{CMOS}}$, where β is a dimensionless factor larger than 1 that depends on the CMOS cell complexity. Relative to the CMOS pin array, the nanowire crossbar is turned by angle $\alpha = \arctan(1/a) = \arcsin(F_{\text{nano}}/\beta F_{\text{CMOS}})$, where a is an integer. This approach allows a unique access to any nanodevice, even if $F_{\text{nano}} \ll F_{\text{CMOS}}$, with the theoretical 100% fabrication yield even in the absence of any alignment between the CMOS and crossbar subsystems. (For a detailed description of the CMOL interface, see, e.g., Refs. 19, 26.)

Earlier we showed [26, 28] that general-purpose, reconfigurable (FPGA-like) digital CMOL circuits may provide a very substantial advantage (more than two orders of magnitude) in density, at slightly faster speed, in comparison with purely semiconductor FPGA

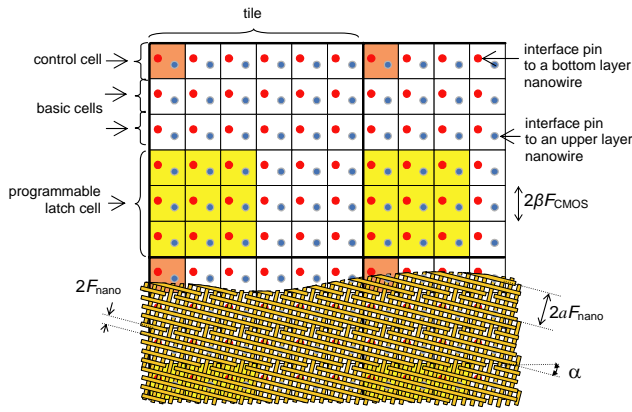


Figure 1: The fragment of the three-cell CMOL DSP fabric for the particular case $\alpha = 4$.

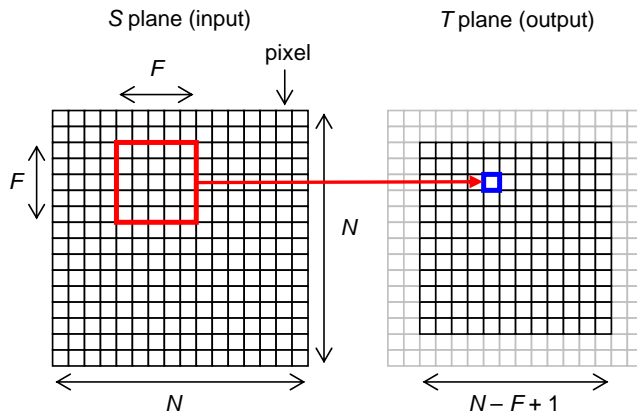


Figure 2: Scheme of the 2D image convolution for particular (impracticably small) sizes of the initial image ($N = 16$), and filter window ($F = 5$).

circuits fabricated with the same design rules, at similar power. It is remarkable that this high performance may be achieved simultaneously with high ($>20\%$) tolerance to some device fabrication defects. Independently, we have shown that mixed-signal CMOL circuits (using analog signals and binary elementary synaptic weights) with bio-inspired “CrossNet” architecture [29] may provide unprecedented performance for some special types of information processing including online image recognition [30]. (The defect tolerance of such circuits is even higher, in some cases reaching $\sim 90\%$ [29].)

The objective of this work has been to explore possible performance of CMOL-based reconfigurable digital and mixed-signal circuits for low-level image processing tasks, on a simple but representative example of 2D image convolution:

$$T_{x,y} = \sum_{i=1}^F \sum_{j=1}^F S_{x+i,y+j} \varphi_{i,j}, \quad (1)$$

where S and T are input and output images, correspondingly, with $N \times N$ pixels each, and φ is a $F \times F$ pixel filter function. Though sometimes special rules for calculating the edge pixels of image T are used [1], we will consider a simplified version of the algorithm where the linear size of output image is smaller by $(F - 1)$ pixels (Fig. 2), so that all output signals T are calculated according to Eq.

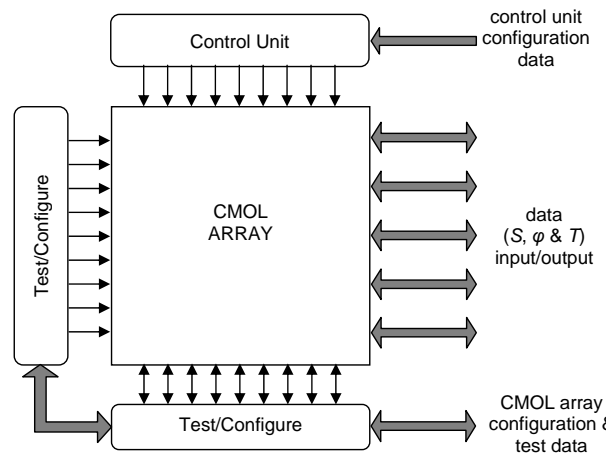


Figure 3: The top-level structure of the CMOL DSP.

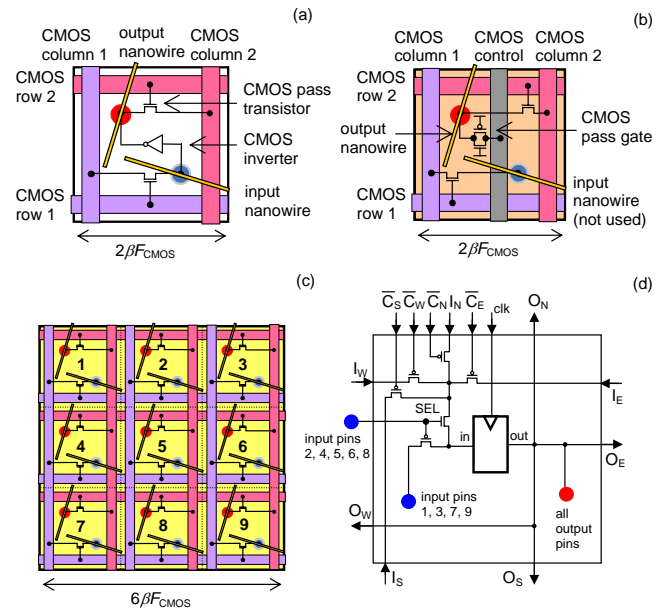


Figure 4: The structure of cells in CMOL DSP: (a) basic cell, (b) control cell, and (c, d) programmable latch cell. For clarity, panel (c) shows only the configuration circuitry, while panel (d) shows the programmable latch implementation.

(1). Such simplification should not affect the performance results for more general case since, typically, $F \ll N$. For example, the baseline parameters used for estimates in this paper are $F = 32$, $N = 1024$, with the similar accuracy ($n_S = n_T = n_\varphi = n = 12$ bits) of the input, output, and filter data.

2. DSP HARDWARE PRIMITIVES

Figure 3 shows the top-level architecture of the proposed CMOL-based DSP. Here we assume the most challenging I/O option when the data are fed to, and picked up from the array from the periphery, e.g., the right side on the Fig. 3. (If an area-distributed, 2D I/O is available, the circuit performance would only be better.) The key part of the architecture, the CMOL array, is similar for each pixel. The pixel area is organized into a uniform mesh of square-shaped “tiles” (Fig. 1). The number of tiles per pixel depends on

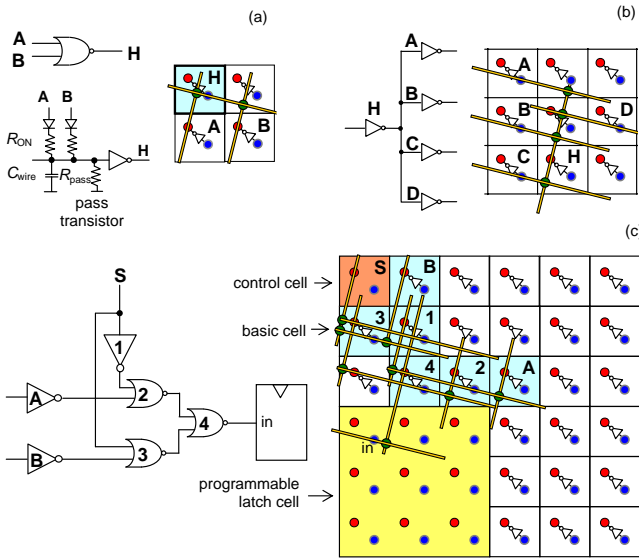


Figure 5: Equivalent circuits and physical implementation of the logic and routing primitives of the CMOL DSP circuits: (a) fan-in-two NOR gate, (b) signal broadcast unit, and (c) 2:1 multiplexer controlled by cell S feeding a programmable latch. For clarity, the physical implementation panels show only the nanodevices set ON and the actively used nanowires. Also, note that for realistic values of the CMOS-to-nano pitch ratio, each nanowire fragment spreads over many more ($\beta F_{\text{CMOS}}/F_{\text{nano}} \gg 1$) basic cells than shown in the figure.

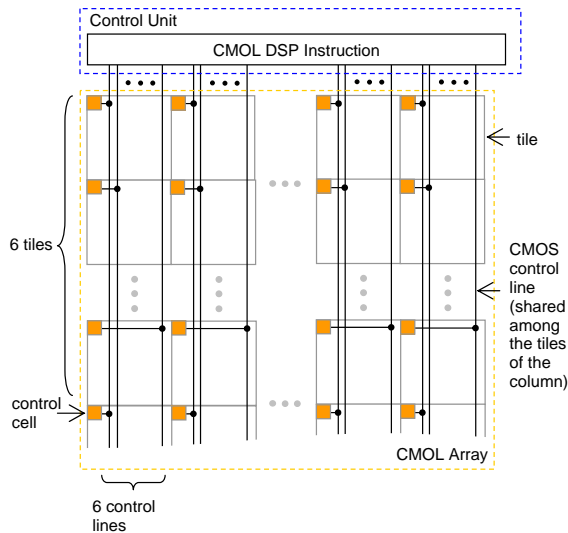


Figure 6: The control signal circuitry for the case $a = 40$, $a' = 36$.

the data word length n ; in our case it is close to 12×12 . Each tile consists of 26 basic cells, one control cell of the similar size, and one programmable latch cell of a larger area (Fig. 1). While the CMOS circuitry of each cell type is different (see Fig. 4), the interface and nanowire levels are the same for all cell types: similarly located pins of each elementary cell contact nanowire fragments of the same length. The CMOS-implemented configuration circuitry,

comprised of a pair of signal lines (shown blue and magenta in Fig. 4) and a pass transistor per pin, is also similar for each cell.

The basic cell (Fig. 4a) has a structure identical to that used in our CMOL FPGA [26]. By setting programmable diodes ON or OFF, each pin of a basic cell may be connected through a nanowire-nanodevice-nanowire link to each of $M = a - 2$ other cells within a nearly-square-shaped “cell connectivity domain”. Figures 5a,b show how the basic cell may be configured to implement a fan-in-two NOR gate and signal broadcast, which are sufficient to implement any Boolean function.

The control cell (Fig. 4b) connects its output nanowire to the CMOS control unit memory (outside the CMOL array) via the designated CMOS control lines - see Fig. 6. Such cells orchestrate the data flow inside the array. For example, they are used to supply control signals to multiplexers and thus select specific inputs to the programmable latch cells - see, e.g., Fig. 5c. There are 6 unique control signals per each tile column in the CMOL array available through corresponding control cells - see Fig. 6. The particular (interleaved) connection pattern of control lines to control cells, as well as the actual number of unique signals, are especially convenient for the typical CMOL parameters $a = \beta F_{\text{CMOS}}/F_{\text{nano}} = 40$ and $a' = 34$ [28]. In this case any cell in a tile column may be connected to any of the 6 control signals in a tile.

The programmable latch cell (Figs. 4c, d) is designed to provide not only temporary data storage but also a fast (CMOS-wire) interconnect between each tile and its four nearest neighbors - see inputs I_S, I_N, I_W, I_E and outputs O_S, O_N, O_W, O_E in Fig. 4d. The latter feature may be used both for window operations and for a fast transfer of data in and out the CMOL array. To implement these functions, each latch cell has a CMOS latch with a programmable input. More specifically, depending on the value of signal SEL, which may arrive from any of five input nanowires (Fig. 4d), the input of CMOS latch can be connected to either any of other four input nanowires or one of the neighboring latch cells. The particular choice of the neighbor is determined by the signals arriving via CMOS-implemented select lines C_S, C_N, C_W, C_E , which are common to all programmable latch cells. The remaining circuitry, i.e. two pairs of orthogonal CMOS lines and two pass transistors in each cell of programmable latch (Fig. 4c), is used for testing and reconfiguration, just like described in Ref. 28. After the reconfiguration the pass transistors pull the voltage on the input nanowires to the ground, so that the default input to the programmable cell comes from the input nanowires rather than from the neighboring latches.

CMOS layout estimates have shown that the basic and control cells can be fit into the $64(F_{\text{CMOS}})^2$ squares each, thus giving $\beta_{\text{min}} = 4$ [28]. (Such compact layout is possible because typical currents in CMOL FPGAs are very small [28], so that all transistors in our design, including p-MOS, may be of the minimum width.) The programmable latch readily fits into an area nine times larger. We also expect that, since the linear size of one basic cell is sufficient to fit four metal lines, CMOS routing should not take more than 6 layers of metal.

3. CASE STUDY: 2D IMAGE CONVOLUTION

3.1 Top Architecture

It is obvious from Eq. (1) that the convolution can be effectively parallelized. For digital CMOL DSP circuitry, only partial parallelization is feasible. At this approach, the convolution process may be broken into F^2 sequential steps, each corresponding to a

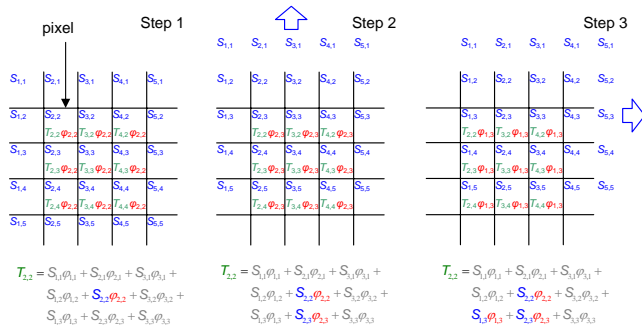


Figure 7: Three sequential time steps of the convolution in the left top corner of the CMOL DSP array for $F = 3$. Colored terms in the formulas below each panel show the calculated partial sums in the pixel 2,2. For the (uncharacteristically small) filter size, it takes just $F^2 = 9$ steps to complete the processing of one frame.

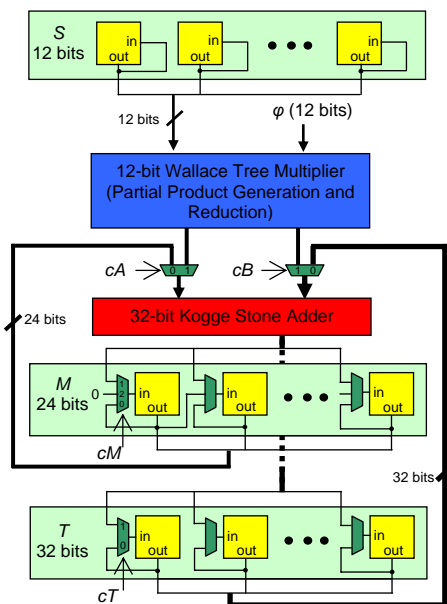


Figure 8: Pixel schematics for the 2D convolution with considered parameters. Note that this picture only shows connections which are implemented with nanowires. In reality there are also connections (between neighbor programmable latches) implemented with CMOS scale lines.

specific pair of indices i, j , the same for all pixels. At each step, every pixel with coordinates x, y of the CMOL array is supplied with one component $S_{x+i, y+j}$ of the input signal matrix, and all pixels are supplied, in parallel, with the same component $\phi_{i, j}$ of the window function. During the step, the pixel circuitry calculates the product $S_{x+i, y+j}\phi_{i, j}$ and adds it to the partial sum of $T_{x, y}$, which is kept in that pixel all the time. These add-and-multiply operations are done in all pixels in parallel (Fig. 7), so that the whole convolution (of one input frame) is accomplished in F^2 steps.

It is evident that the most economical choice of sequential pairs i, j corresponds to the shift of the input image by one pixel in any direction - see Fig. 7.

3.2 Mapping on the CMOL DSP Fabric

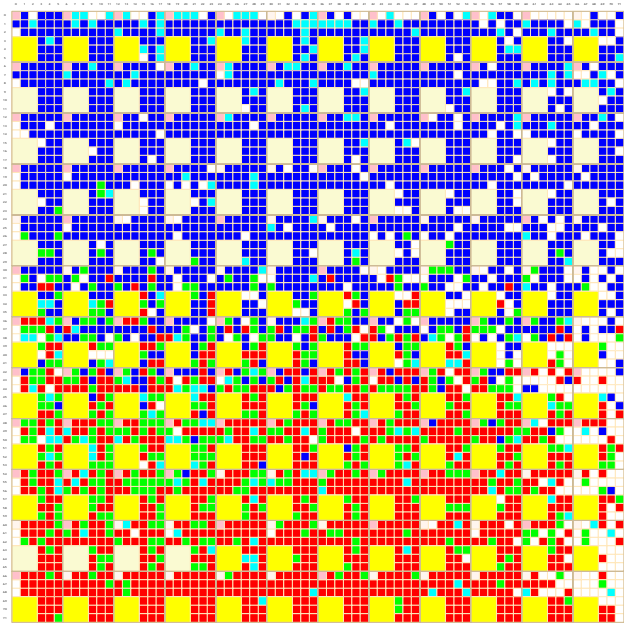


Figure 9: The mapping of the pixel on CMOL DSP (for $F_{CMOS}/F_{nano} = 10$) after its successful reconfiguration of the circuit around as many as 40% of bad nanodevices with random locations. Programmable latches A and B are used for bypass circuitry during the data up and down shift operations.

Figure 9: The mapping of the pixel on CMOL DSP (for $F_{CMOS}/F_{nano} = 10$) after its successful reconfiguration of the circuit around as many as 40% of bad nanodevices with random locations. Programmable latches A and B are used for bypass circuitry during the data up and down shift operations.

One of the advantages of the considered parallelization algorithm is that all pixels may have similar structure (Fig. 8). In this schematics, the two most complex parts are the 12-bit multiplier (for the partial product generation and reduction) and the 32-bit adder. The adder is used both for the last step in the multiplication and for the summation of the products in Eq. (1). Such dual use of the adder requires additional multiplexers (whose CMOL-DSP implementation is described in details in Ref. 31) at the input of the adder (i.e., cA and cB on Fig. 8) and latch M for keeping intermediate values.

We have found [31] that the best performance for the convolution task with the considered parameters can be achieved with a multiplier featuring straightforward partial product generation and the Wallace-tree like reduction scheme [32]. The summation is implemented with a parallel 32-bit Kogge-Stone adder, designed in our previous work [28]. Note that for the considered accuracy $n = 12$ the initial multiplication product is a 24-bit number, and adding all $F^2 = 1024$ products increases the width of output T by yet other 10 bits. This is why at the end of the multiplication step the last 2 digits are dropped (via the right shift operation), so that the subsequent addition produces numbers with 32-bit accuracy. The resulting 32-bit value of T is converted to the 12-bit output merely by dropping the least significant bits.

The remaining pixel hardware (not shown in Fig. 8) includes bypass circuitry [31] which is required to shift S values up and down the pixel array (Fig. 7) while bypassing the programmable latches holding the partial sums of T . (The left and right shifts of S and

T can be done with CMOS lines only, due to the specific mapping of the cells inside the pixel - see Fig. 9.) To accomplish the bypass during, e.g., shift down, the inputs of each programmable latch in a row just below that holding T , are connected to the outputs of the programmable latches in a row just above the T row via two basic cells used as CMOS inverters.

In order to map the pixel circuitry (Fig. 8) on the CMOL DSP fabric, we first created its VHDL structural model and performed its cycle-accurate verification with the help of the ModelSim tool [33]. After that, the VHDL structural file was converted into a flat blif file using the SIS tool [34] and then mapped on CMOL DSP using a semi-manual approach. Figure 9 shows the typical pixel mapping. Note that though the utilization is very high, i.e. about 80% of the whole area of the pixel, there is still some space to add more functions to the pixel (e.g., a more sophisticated rounding scheme, etc.) if necessary, without increasing its area.

3.3 Control

Tables 1, 2 show the full set of instructions (control signals) sent to a pixel for the considered task. The algorithm starts by shifting the initial data S and T (the latter numbers are set to 0 in the beginning) by N pixels, i.e. $12N$ tiles, from the periphery into the pixel array (Fig. 3). This is implemented with $12N$ “shift-all-left” instructions, which set the SEL signals of all programmable latches to high, and thus enable the nearest-neighbor, CMOS-based interconnects. Simultaneously, the same instruction sets the direction of the data movement by setting the signal C_E to high.

After all the necessary data are in place, the first partial product of T (see the left column of Fig. 7) is calculated by executing “multiplication”, “shift- M -right”, and “addition” instructions. To calculate the next partial product, matrix S is shifted by one pixel, using one of “shift- S -up”, “shift- S -down”, “shift- S -left”, “shift- S -right” instructions. For example, the data shifts shown in the next two columns of Fig. 7 are achieved by executing “shift- S -up” and “shift- S -right” instructions 7 and 12 times, respectively. (The reduced number of shifts in the vertical direction is due to bypassing 5 rows of programmable latches holding M and T values.)

Finally, when the matrix T has been calculated, it is shifted out of the array to the periphery using $12N$ “shift-all-right” instructions.

3.4 Defect Tolerance

The defect tolerance of the circuit has been evaluated with the help of the CMOL FPGA CAD algorithm described in Ref. [28]. For each initially mapped pixel, the program has been run 10,000 times with randomly chosen set of defects, formed with the same probability q . Such simulation has shown that in some cases, the pixel circuitry may be successfully reconfigured around as many as 40% defects (Fig. 9). Figure 10 shows the Monte Carlo results for the yield of one pixel, for several typical values of the $F_{\text{CMOS}}/F_{\text{nano}}$ ratio. The linear extrapolation of these results (in the log-log scale shown in Fig. 9) indicates that for the typical case $F_{\text{CMOS}}/F_{\text{nano}} = 10$ ($a = 40$, $a' = 34$) the whole array with 1024×1024 pixels can have an acceptable circuit yield of 90% at the nanodevice defect rate about 20%.

3.5 Performance

We have estimated the performance of the 2D image convolution mapped on CMOL DSP chip for a particular choice of parameters, $F_{\text{nano}} = 4.5$ nm and $F_{\text{CMOS}} = 45$ nm, which might be typical for the initial stage of the CMOL technology development [35]. Using the cell area estimates made in Sec. 2, the size of one pixel is about $25 \times 25 \mu\text{m}^2$, while the size of full CMOL array with $N = 1024$ pixels is $\sim 25 \times 25 \text{mm}^2$.

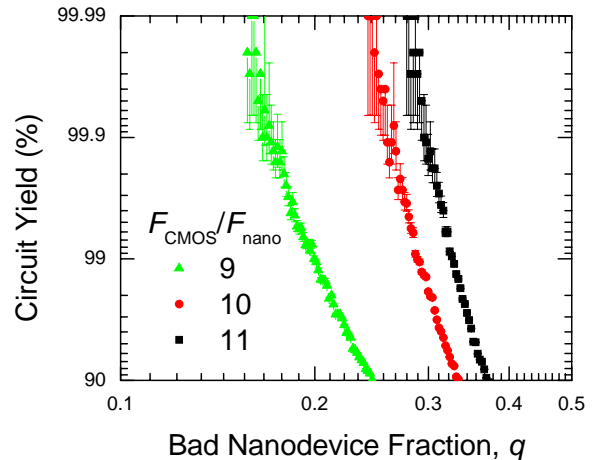


Figure 10: The final (post-reconfiguration) defect tolerance of the convolver for several values of $F_{\text{CMOS}}/F_{\text{nano}}$. The figure shows the defect tolerance of the circuits on the log scale which makes visible the results for the most interesting (high) values of yield.

Our latency calculations followed those of previous works on digital CMOL circuits [26, 28]. Assuming a plausible power supply voltage $V_{\text{DD}} = 0.3$ V [26], we first find the smallest acceptable nanodevice resistances, and consequently the lowest delay of CMOS gates, such that the power consumption density does not exceed the ITRS-specified value of 200 W/cm^2 and the voltage swing on the input of CMOS inverters is sufficiently larger than the corresponding shot and thermal noise of nanodevices [26, 28]. From such optimization the nanodevice resistance in the ON state is about $20 \text{ M}\Omega$, while delay τ_0 of a NOR-1 gate is about 100 ps.

In general, it would be beneficial to choose a clock cycle time almost as small as the NOR-1 gate delay, since some operations (e.g., moving data from one programmable latch to the nearest neighbor latch via CMOS lines), can be completed in one such cycle. On the other hand, building a few-gigahertz clock distribution network for a 625 mm^2 CMOL DSP chip might be a problem. For example, it would take roughly 300 ps for signal to propagate from one end of the chip to the other. This is why for our estimates we have used a conservative 1-ns clock cycle, which can be readily implemented.

Table 3 summarizes the latency of the instruction execution. For example, bypassing of the occupied programmable latch can be done in less than 200 ps and therefore all shift operations can be completed in one cycle. The addition latency is determined by the select-output delay of the adder multiplexers ($8\tau_0$), the delay of the 32-Kogge Stone adder ($33\tau_0$), the input-output delay of multiplexers connected to programmable latches holding the M (or T) values ($4\tau_0$). These operations together take 45 NOR-1 gate delays, or 4.5 ns, so that the actual delay may be calculated from the smallest integer number of clock cycles which is larger than the physical delay, i.e. 5 clocks or 5 ns. Similarly, the physical delay of multiplication, which is comprised from that of partial product generation ($3\tau_0$), 12-bit Wallace tree partial product reduction ($48\tau_0$), input-output adder multiplexer ($4\tau_0$), 32-Kogge Stone adder ($33\tau_0$), and input-output programmable latch multiplexer ($4\tau_0$), is 92 NOR-1 gate delays, or 10 clocks.

Using the notation from Table 3, the full delay of convolution is therefore can be calculated as

$$\tau = 7F(F - 1)\tau_S + 12F\tau_S + F^2(\tau_M + 2\tau_S + \tau_A). \quad (2)$$

Operation	Control and input bit values																										
	C _E	C _W	C _N	C _S	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	23	24
shift all left	1	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
shift all right	0	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
shift S left	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
shift S right	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
shift S down	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
shift S up	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
shift M right	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
addition	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
multiplication	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1: The full set of control and input signals.

Bits	Control and input signal connection
1	to SEL inputs of all latches except cells S, A, B
2	to SEL inputs of cells S (0-11 bits)
3	to SEL inputs of cells A (0-11 bits)
4	to SEL signal of cells B (0-11 bits)
5	to multiplexer cA
6	to multiplexer cB
7	to multiplexer cM (0 bit)
8	to multiplexer cM (1 bit)
9	to multiplexer cT (0 bit)
10	1/0 constant
11	1/0 constant
12-24	φ value (0-11 bits)

Table 2: Destinations of control and input signals.

Operation	Delay (in NOR1 units)	Delay (in cycles)	Notation
shift all left	< 5	1	
shift all right	< 5	1	
shift S left	< 5	1	T _S
shift S right	< 5	1	
shift S down	2	1	
shift S up	2	1	
shift M right	2	1	
addition	45	5	T _A
multiplication	92	10	T _M

Table 3: Instruction latency calculation results.

In this equation, the first and second terms account for vertical and horizontal movement of S data, respectively, while the last term includes calculation and addition of partial sums (with rounding). Plugging in the numbers discussed above, the full delay for our parameters may be estimated as 25 microseconds.

4. MIXED-SIGNAL OPTION

The digital signal processing discussed in two previous sections, implies that the analog outputs from the sensor array are first digitized. However, there is another option: first carry out the front-end signal processing (in our example, convolution) in the analog form and only then digitize the output signals for further processing.

Figure 11 shows the general idea how the convolution may be done using a mixed-signal CMOL circuit. Analog inputs S , presenting signals from each pixel of the sensor array, are fed into the interface pins (shown red in Fig. 11) leading to nanowires of the CMOL crossbar. The latching switches, contacted by the input (“red”) nanowire carrying an input signal S_{r+k} , are set to either ON or OFF states to represent binary digits $\varphi_k^{(l)}$ of numbers of the corresponding window multiplier $\varphi_k (l = 1, 2, \dots, n)$. (Here r and k are just a shorthand for the index pairs $\{x, y\}$ and $\{i, j\}$, respec-

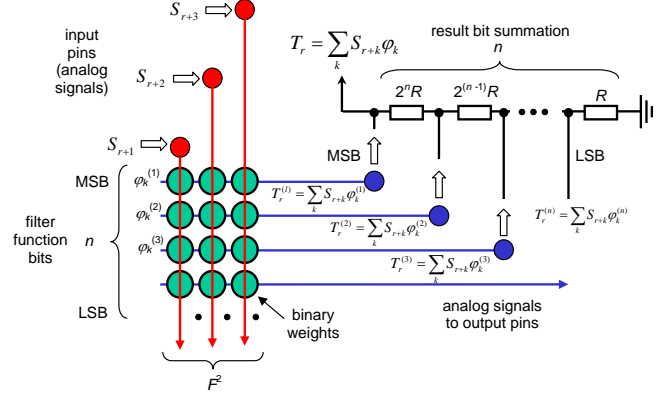


Figure 11: The basic idea of the CMOL mixed-signal convolution.

tively.) As a result, current injected through all F^2 switches into the output (“blue”) nanowire is proportional to the sum

$$T_r^{(l)} = \sum_{k=1}^{F^2} S_{r+k} \varphi_k^{(l)}. \quad (3)$$

Now, these signals are summed up, in a resistor circuit (see the right part of Fig. 11), with the corresponding weights proportional to 2^l . If the weights are exact, the output analog signal is proportional to

$$T^{(l)} = \sum_{l=1}^n 2^l \sum_{k=1}^{F^2} S_{r+k} \varphi_k^{(l)} = \sum_{k=1}^{F^2} S_{r+k} \sum_{l=1}^n 2^l \varphi_k^{(l)} = \sum_{k=1}^{F^2} S_{r+k} \varphi_k, \quad (4)$$

i.e. to the required result (1).

This procedure may be extremely area-efficient, involving just $F^2 n$ crosspoints for each add-multiply operation. For our sample values ($F = 32, n = 12$) this is just $\sim 12,300$ crosspoints, with the total area (for $F_{\text{nano}} = 4.5 \text{ nm}$) below $1 \mu\text{m}^2$. This means that the necessary crossbar would fit on a minor fraction of one pixel area, i.e. the operation may be carried out in parallel for all pixels of the output image, making it extremely fast as well. For example, with the same average power dissipation density ($P_0 = 200 \text{ W/cm}^2$) as used for the digital option estimates above, and a pixel size of $A = 10 \times 10 \mu\text{m}^2$, we can use crosspoint devices with ON current as high as $I_{\text{ON}} = 2P_0 A / F^2 n V \sim 100 \text{ nA}$. (The factor 2 is due to the fact that on the average only a half of the latching switches, representing binary digits, is in the ON state. In this estimate, the power supply voltage V is, as before, 0.3 volt.) This means that the worst-case time constant of the output wire recharging, $\tau_{\text{max}} \approx (2F_{\text{nano}} F^2 C_0) V / I_{\text{ON}}$, which is the main component

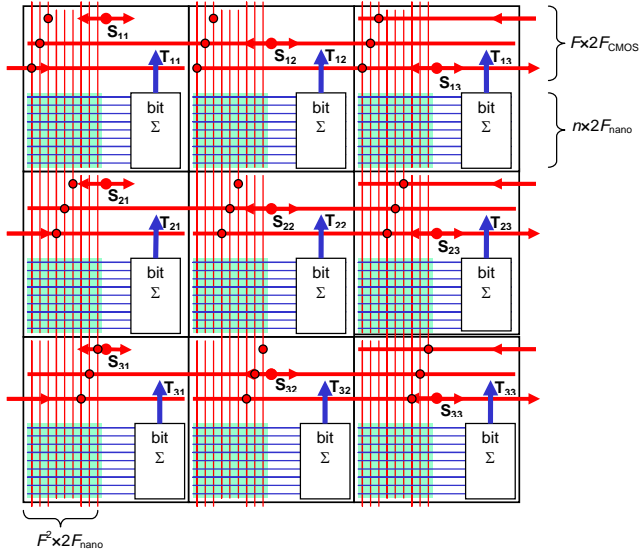


Figure 12: Signal transfer for convolution, for clarity shown for the (unrealistically low) window size $F = 3$. Green rectangles denote the location of nanowire crossbar arrays. (In reality, the bit summation blocks, implemented in CMOS, may be located below these arrays.)

of the convolution time, is very small. Using the typical value of the specific capacitance of the crossbar nanowire, $C_0 \approx 0.2 \text{ fF}/\mu\text{m}$ [28], we get for τ_{max} max an estimate of less than 10 nanoseconds, while the typical time $\tau_{\text{ave}} \approx 2(2F_{\text{nano}}C_0)V/I_{\text{ON}}$ is of the order of 10 picoseconds (!).

However, the mixed-signal approach faces several tough challenges, starting with the necessary interconnects. Indeed, in order to carry out the convolution for all pixels in parallel, each single-pixel convolver (Fig. 11) should be fed by F^2 input signals from each pixel of the surrounding window (Fig. 2). Implemented in the CMOS subsystem, these interconnects would require two much real estate. Indeed, the total width of the wires leading to or from one pixel would be at least $F^2 \times 2F_{\text{CMOS}}$, in our example $\sim 65 \mu\text{m}$, i.e. substantially larger than the pixel size.

Figure 12 shows an architecture which solves this problem. In this approach, each input signal is fed into a CMOS-level bus (in Fig. 12, presented with a bold red line) which is tapped by F “red” interface pins leading to input nanowires (thin red lines). Each nanowire reaches F elementary convolvers (Fig. 11), feeding them with the same input signal S_i . In this way, the additional interconnect width is just $F \times 2F_{\text{CMOS}}$ per pixel - in our numerical example, only $\sim 2 \mu\text{m}$, comfortably below the linear pixel size ($10 \mu\text{m}$).

Another problem of the mixed-signal approach is the shot noise of the latching switches, whose spectral density $S_I(f)$ cannot be much less than the Schottky value $2eI_{\text{ON}}$, because of a relatively small length of these devices. In order for the noise not to decrease the output signal accuracy, the total r.m.s. noise of $F^2/2$ open devices feeding each output line, within the output bandwidth Δf , should not exceed the least significant bit of the total signal current $F^2 I_{\text{ON}}/2$. This gives us the following condition on Δf :

$$[eI_{\text{ON}}F^2\Delta f]^{1/2} \leq I_{\text{ON}}F^2/2^{n+1}. \quad (5)$$

For our sample parameters ($n = 12$), Eq. (5) yields $\Delta f \leq 50 \text{ MHz}$, corresponding to $\sim 20 \text{ ns}$ averaging time. Although this time is longer than even the maximum value of the signal delay τ ,

it is still much shorter than the full convolution time of the digital option - see Sec. 3 above. Note also that Δf is an exponential function of the required accuracy of the output signal, so that if n is less than the 12 bits accepted in our estimate, the bandwidth limitation would be substantially softer.

Last but not least, the mixed-signal implementation imposes hard requirements on random variations of the main parameter of the programmable diodes, current I_{ON} . Indeed, with the same requirement as for the shot noise, the relative r.m.s. spread $\Delta I_{\text{ON}}/I_{\text{ON}}$ should not exceed $F/2^{n+1}$, in our example $\sim 0.5\%$. In the most advanced experiments we are aware of [20], the statistical spread of the current was much higher, about 20%, even with the additional stabilization of I_{ON} with a MOSFET incorporated in each crosspoint. Probably the only feasible way to reach the necessary stability is the further improvement of the incorporated devices.

5. DISCUSSION

In general, a fair comparison between both CMOL options we have explored and more traditional ways of image processing is hard to make. The main reason is that our target parameters (most importantly, the filter size and operand accuracy) are substantially higher than those reported in literature. Most implementations (except for general purpose microprocessors, like Cell [4], and FPGAs [9]) are very parameter-specific so that it is not quite clear how their performance would scale to a larger image size. Most of the reported focal array processors could only perform convolution with a very small filter size $F \leq 5$, so they could keep all components of matrix φ within one pixel circuit area. Keeping all 1024 values of φ with 12-bit precision in each pixel, as required for the filter size we have considered, is not an option for the current or even rationally envisioned CMOS technologies. Moreover, there is some inconsistency in the internal computation accuracy: while some implementations have the same internal accuracy as that of the input values [10], others have much wider functional units than necessary, e.g., 32-bit adders used for 8-bit pixel operations in Ref. 11. For this reason we will only compare our results to the performance of the Cell processor, whose performance can be estimated for a wide range of parameters.

Using the data from Ref. 5, it will take approximately 30 ms for a 90-nm 3.2 GHz Cell processor to calculate convolution with the parameters we have considered (with the 32-bit internal accuracy). Even assuming a very optimistic (linear) delay scaling and possible increase in the number of cores (from 8 to 32), the corresponding latency of a hypothetical 45-nm 6.4 GHz Cell processor would be about 3.5 ms. This number is at least 100 times larger than that of proposed CMOL DSP. This is not surprising since the CMOL DSP has a much higher peak performance. For example, it can theoretically perform 250×10^{12} 32-bit additions per second or about 100×10^{12} 12-bit multiply-add operations per second, i.e. above two orders of magnitude higher than Cell. Moreover, the theoretical data bandwidth of a CMOL DSP could be as high as 10 Tbit/s, which should be enough for even very demanding applications.

It is worth noting that a major advantage of the Cell-type processors for the low-level image processing tasks is a very fast (nanosecond-scale) time necessary for changing the running task (e.g., the filter size). CMOL DSP can almost certainly have a sub-100-microsecond time of switching from one task to another. (This estimate follows from the experimental sub-50-ns switching times of the metal-oxide-based programmable diodes [20] and the fact that nanodevices in different connectivity domains can be programmed simultaneously [28]). Optimizing the structure of CMOL DSP to reduce the reconfiguration time is one of the possible future research directions.

The range of urgent hardware development tasks is much broader, including in particular: (I) the design, fabrication, and characterization of programmable diodes with incorporated semiconductor diodes, with the ON current reproducible better than $\sim 1\%$ (which would allow to pursue the ultrafast mixed-signal option described in Sec. 4), (II) scaling of reproducible crosspoint nanodevices below 10 nm (which may require the transfer from the metal-oxide-based programmable diodes to single-electron-based SAM junctions [19]), and (III) experimental demonstration of an area-distributed CMOL interface, which may radically change the industrial perception of the hybrid CMOS/nanodevice circuits.

Acknowledgment

Useful discussions of various aspects of digital CMOL circuits with S. Das, A. DeHon, S. Dirk, J. Ellenbogen, P. Franzon, S. Goldstein, D. Hammerstrom, A. Kaloyeros, R. Karri, R. Kiehl, P. Kuekes, J. Lukens, A. Mayr, A. Orailoglu, G. Snider, M. Stan, D. Tennant, R. van Zee, K. Wang, R. S. Williams, T. Zhang, and N. Zhitenev are gratefully acknowledged. The work has been supported in part by AFOSR, DTO, and NSF.

6. REFERENCES

- [1] R. Lewis, *Practical Digital Image Processing*. New York: Ellis Horwood, 1990.
- [2] P. H. Swain and S. M. Davis, *Remote Sensing. The Quantitative Approach*. New York: McGraw-Hill, 1978.
- [3] S. M. Chai, *et al.*, "Focal-plane processing architectures for real-time hyperspectral image processing," *Appl. Optics*, vol. 39, pp. 835–849, 2000.
- [4] D. C. Pham, *et al.*, "Overview of the architecture, circuit design, and physical implementation of a first-generation cell processor," *IEEE J. Solid-State Circuits*, vol. 41, pp. 179–196, 2006.
- [5] *Benchmark: Algorithm Performance on the Cell Broadband Engine Processor - White Paper 2006*, available online at http://www.mc.com/literature/literature_files/Cell-Perf-Simple.pdf.
- [6] S. Agarwala, *et al.*, "A 600-MHz VLIW DSP," *IEEE J. Solid-State Circuits*, vol. 37, pp. 1532–1544, 2002.
- [7] C. Basoglu, *et al.*, "Single-chip processor for media applications: The MAP1000 (TM)," *Int. J. Imaging Syst. Technol.*, vol. 10, pp. 96–106, 1999.
- [8] N. K. Ratha and A. K. Jain, "Computer vision algorithms on reconfigurable logic arrays," *IEEE Trans. Parallel Distrib. Syst.*, vol. 10, pp. 29–43, 1999.
- [9] C. Torres-Huitzil and M. Arias-Estrada, "FPGA-based configurable systolic architecture for window-based image processing," *EURASIP J. Appl. Signal Process.*, vol. 2005, no. 7, pp. 1024–1034, 2005.
- [10] P. Foldesy, *et al.*, "Digital implementation of cellular sensor-computers," *Int. J. Circuit Theory Appl.*, vol. 34, pp. 409–428, 2006.
- [11] G. Linan, *et al.*, "ACE16K: An advanced focal-plane analog programmable array processor," in *Proc. ESSCIRC 2001*, 2001, pp. 201–204.
- [12] P. Dudek and P. J. Hicks, "A general-purpose processor-per-pixel analog simd vision chip," *IEEE Trans. Circuits and Syst. I - Regul. Pap.*, vol. 52, pp. 13–20, 2005.
- [13] R. Etienne-Cummings, Z. K. Kalayjian, and D. H. Cai, "A programmable focal-plane mimd image processor chip," *IEEE J. Solid-State Circuits*, vol. 36, pp. 64–73, 2001.
- [14] *International Technology Roadmap for Semiconductors. 2005 Edition*, available online at <http://public.itrs.net/>.
- [15] K. K. Likharev, "Electronics below 10 nm," in *Nano and Giga Challenges in Microelectronics*. Amsterdam: Elsevier, 2003, pp. 27–68.
- [16] M. R. Stan, *et al.*, "Molecular electronics: From devices and interconnect to circuits and architecture," *Proc. IEEE*, vol. 91, no. 11, pp. 1940–1957, 2003.
- [17] A. DeHon, "Nanowire-based programmable architectures," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 1, no. 2, pp. 109–162, 2005.
- [18] P. J. Kuekes, G. S. Snider, and R. S. Williams, "Crossbar nanocomputers," *Sci. Am.*, vol. 293, no. 5, pp. 72–80, 2005.
- [19] K. K. Likharev and D. B. Strukov, "CMOL: Devices, circuits, and architectures," in *Introducing Molecular Electronics*, G. Cuniberti, G. Fagas, and K. Richter, Eds. Berlin: Springer, 2005, pp. 447–478.
- [20] A. Chen, *et al.*, "Non-volatile resistive switching for advanced memory applications," *IEDM Tech. Digest*, p. 31.3, 2005.
- [21] G.-Y. Jung, *et al.*, "Circuit fabrication at 17 nm half-pitch by nanoimprint lithography," *Nano Lett.*, vol. 2, no. 3, pp. 351–354, 2006.
- [22] J. E. Green, *et al.*, "A 160-kilobit molecular electronic memory patterned at 10(11) bits per square centimetre," *Nature*, vol. 445, no. 7126, pp. 414–417, 2007.
- [23] M. Bender, *et al.*, "Status and prospects of uv-nanoimprint technology," *Microel. Eng.*, vol. 83, pp. 827–830, 2006.
- [24] H. H. Solak, "Nanolithography with coherent extreme ultraviolet light," *J. Phys. D -Appl. Phys.*, vol. 39, pp. R171–R188, 2006.
- [25] K. K. Likharev, "CMOL: A silicon-based bottom-up approach to nanoelectronics," *Interface*, vol. 14, pp. 43–46, 2005.
- [26] D. B. Strukov and K. K. Likharev, "A reconfigurable architecture for hybrid CMOS/Nanodevice circuits," in *FPGA'06*. New York, NY: ACM Press, 2006, pp. 131–140.
- [27] —, "Defect tolerant architectures for nanoelectronic crossbar memories," *J. Nanosci. Nanotechnol.*, vol. 17, pp. 151–167, 2006.
- [28] —, "CMOL FPGA: A reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices," *Nanotechnology*, vol. 16, no. 6, pp. 888–900, 2005.
- [29] O. Türel, *et al.*, "Neuromorphic architectures for nanoelectronic circuits," *Int. J. Circ. Theory App.*, vol. 32, no. 5, pp. 277–302, 2004.
- [30] J. H. Lee and K. K. Likharev, "CMOL CrossNets as pattern classifiers," in *Proc. of Int. Work-Conf. on Artificial Neural Networks*, Barcelona, Spain, June 2005, pp. 446–454.
- [31] D. B. Strukov and K. K. Likharev, "Reconfigurable hybrid CMOS/nanodevice circuits for image processing," *submitted to Nanotechnology*, 2007.
- [32] M. J. Flynn and S. F. Oberman, *Advanced Computer Arithmetic Design*. New York: Wiley, 2001.
- [33] *ModelSim XE III 6.1e Software*, available online at <http://www.model.com>.
- [34] E. M. Sentovich, *et al.*, "SIS: A system for sequential circuit synthesis," Tech. Rep., available online at citeseer.ist.psu.edu/sentovich92sis.html.
- [35] K. K. Likharev and D. B. Strukov, "CMOL technology development roadmap," 2007, submitted to NanoArch.