

# Wire-Streaming Processors on 2-D Nanowire Fabrics

Teng Wang, Mahmoud Ben-Naser, Yao Guo and Csaba Andras Moritz  
 Electrical and Computer Engineering Department  
 University of Massachusetts Amherst, MA 01003, USA  
 {twang, mbennase, yaoguo, andras}@ecs.umass.edu

**Abstract**—Most of the research in the field of nanoelectronics has been focused on nanodevice and nano-fabrication aspects. By contrast, very little work has been reported on the design or capabilities of circuits and computational architectures that can be built out of nanodevices. A key challenge in any nanoscale system is to preserve the density advantage of the underlying nanodevice. Topological, interconnect, and fault-tolerance constraints can, however, severely impact the effective system-level density that can be achieved. This paper explores designs on silicon nanowire based 2-D computing fabrics that, while also programmable and hierarchical, are more tuned towards an application domain than PLA style approaches. Our goal is to achieve denser designs with better fabric utilization and efficient cascading of circuits. We call these designs *NASICs: Nanoscale Application-Specific Integrated Circuits*. In this paper we demonstrate possible NASIC optimizations, within the constraints of sub-lithographic fabrication, that improve fabric utilization. Finally, we design and evaluate a simple nanoscale stream processor and compare its density to an equivalent synthesized CMOS implementation scaled to 30-nm CMOS. Our results show that, despite this small design being dominated by microwires, a 12.5X density advantage can be achieved on a defect-free fabric. We estimate that larger NASIC designs could be 100X denser compared to CMOS before fault tolerance techniques are applied.

## 1. INTRODUCTION

The most promising underlying nano-technologies today are semiconductor nanowires (NWs) and arrays of crossed carbon nanotubes (CNTs). Researchers have built FETs and diodes out of NWs [6] and CNTs [11]. Fluidic alignment, has been used to construct arrays[13].

In this paper we use 2-D grids of NWs, with the grid junctions acting, as required, as FETs or diodes, or be disconnected. The work presented here is part of our effort to build *NASICs: Nanoscale Application-Specific Integrated Circuits* and builds on our previous work on static-style NASICs [25]. The elemental units in NASICs are the *tiles*. A tile contains circuits for adders, multiplexers, and flip-flops, etc. Individual tiles can be connected with nanowires or microwires to form a larger multi-tile structure performing an application-specific computation. Microwires also permit the tiles to be programmed by appropriately setting the type of each crosspoint[15].

NASICs face design challenges not encountered in the world of traditional microelectronic devices. For example, the defect levels in nano-fabrics tend to be quite high: we have to design enough fault-tolerance to sustain functionality in the face of a substantial fraction of the circuits being faulty. The

overhead of microwires used for global communication and programming can also be quite high. Moreover, density tends to be less when two-level, rather than multi-level, logic is used. Additionally, there are topological and device constraints which must be taken into account.

Unfortunately, as we will show, it is hard to design high-density sequential circuits at the nanoscale, based on traditional MOS-like approaches. For example, when logic is cascaded or when sequential circuits are used, only the diagonal portion of the logic area tends to be utilized: most of the rest is essentially wasted.

To address this problem, we propose *implicit nano-latches*, which provide high-density temporary storage at the nanoscale, without requiring explicit latching. Our approach preserves the density advantages of nanodevice-based fabrics, while permitting the design of complex, high-density, pipelined structures.

We present the design of a complete but simple stream processor, called *Wire-Streaming Processors (WISP-0)*, that exercises many of the different NASIC circuit styles and optimizations. In WISP-0, in order to preserve the density advantage of nanodevices, data are streamed through the fabric with minimal control/feedback paths. Intermediate values produced during processing are often stored on the nanowires using implicit nano-latches. A simple ISA is defined to illustrate how the design works. We rely on the compiler providing the necessary padding between instructions to avoid resolving dependencies at runtime. This allows us to significantly increase the effective density by avoiding feedback paths and control logic as much as possible.

Finally, the paper shows a comparison between the density of a fully implemented CMOS version and the NASIC WISP-0. Our evaluation shows that WISP-0 has 12.5X density advantage compared to our synthesized CMOS design scaled to 30-nm CMOS, assuming a defect-free fabric. We estimate that the density ratio can be much higher for larger scale NASIC designs because of lower overall microwire overhead. These ratios do not yet account for fault-tolerance techniques. We are currently pursuing various approaches for circuit-level fault tolerance and mapping CAD tools.

The rest of this paper is organized as follows. Background information on nanodevices and static NASICs is presented in Section 2. Section 3 explores some of the challenges faced when designing sequential circuits. The proposed dynamic solutions are shown in Section 4. The Wire-Streaming Processor design is presented in Section 5. Section 6 provides the density evaluation and comparison with a synthesized CMOS version scaled to 30-nm BPTM technology. We conclude with

## 2. TECHNICAL BACKGROUND

## 2.1. Nanoscale Devices

Nanowires (NWs) and Carbon Nanotubes (CNTs) have sizes of the order of a few nanometers, and their density can be as high as  $10^{12}$  switches/cm<sup>2</sup> [8]. The electrical characteristics of nanowires can be more reliably controlled than those of nanotubes [6].

Current control in NWs or CNTs is exerted using gates formed in various ways, or by forming diode junctions. FET behavior has been achieved using metallic gates [11],[18], and crossing NWs or CNTs [6]. By varying the amount of oxide grown at their intersection, crossing CNTs or NWs can be made such that one NW forms a diode with the other, or one acts as a FET gate to the other, or they do not couple [6].

Rapid progress is being made in the development of feasible logic devices. Diode resistor logic was demonstrated. At the same time restoring logic was introduced with nanowire FET-resistor logic [6]. Avouris from IBM made important progress toward low power logic by developing complementary devices on the same NT and demonstrated a CMOS-like nano-inverter [14]. Hewlett-Packard Research has developed a molecular crossbar latch (Kuekes, patent #6,586,965).

While there are many practical challenges remaining, it looks like that it will be possible to build regular nanoarrays from uniform length CNTs or NWs. For example, p-doped horizontal NWs and n-doped vertical NWs can form a nanoarray. At the junctions of NWs, p-n diodes or FETs can be produced. Fluidic alignment, which involves suspending nanowires or nanotubes in a solution (e.g., ethanol) and then making the fluid flow along pre-cut channels on a surface, allows one to construct arrays. Both Samsung (Choi, patent #6,566,704) and Iljin Nanotech (Lee, patent #6,350,488) have patented methods of growing vertically-oriented arrays of nanotubes with separations of less than 10 nm.

## 2.2. Static NASIC Circuits

We present two NASIC adder designs based on static ratioed logic. Our designs are based on a programmed grid-structure of CNTs and NWs similar to [15]. Other examples of NASICs are shown in our previous paper [25].

Figure 1 shows a FET-based 1-bit full adder. This design is somewhat similar to a PLA architecture, but shows the use of a buffer plane in between the NOR planes and a circuit-specific layout, and the sizes of the various planes, including the position of the pullups and pulldowns. The thicker wires represent microwires. The thin wires are nanowires: the horizontal wires are P-wires and the vertical ones are N-wires. The doping of the wires along the source-drain of a FET transistor determines the type of the transistor.

The programming of the grid junctions can be realized through a technique described in [15]. During nanoscale addressing, if we apply a voltage on the horizontal and vertical wires, we can program the state of the device at their intersection.

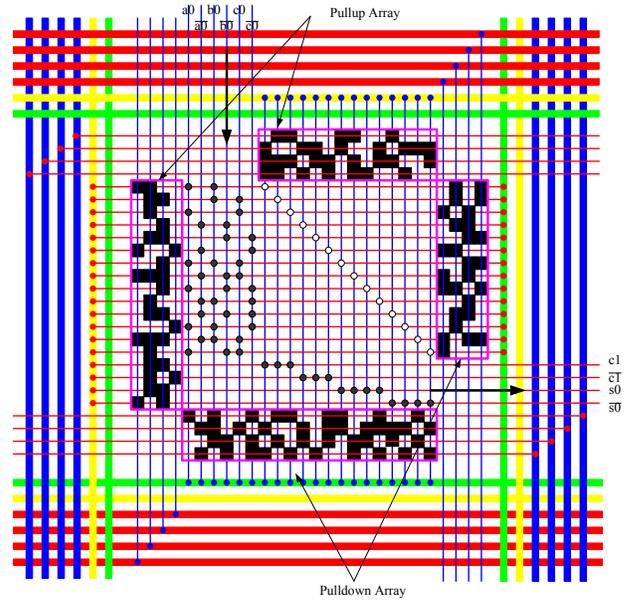


Fig. 1. 1-bit adder realized with NW FETs. Note the impact of the buffer that moves the design into a diagonal shape.

The interface between nanowires and microwires as well as the addressing of the junctions assumes a nanoscale decoder block similar to [24]. The decoders require at least  $O(\log(N))$  microwires (or more if redundancy is incorporated), where  $N$  is the number of the nanowires in one dimension, assuming a square NASIC tile.

An additional requirement is the decoder imprint pattern, as shown around the adder circuit (part of the pullup and pulldown arrays that are also used to connect to  $V_{dd}$  and  $Gnd$ ) in Figure 1. Other schemes that do not require direct patterning (a weakness of this interfacing), e.g., based on stochastic self-assembly, are under development by several research groups [3], [16].

This circuit demonstrates the difficulty of obtaining good fabric utilization due to the presence of buffers. These buffers are needed to turn the corner in order to couple the PFET-based NOR plane that derives the initial product terms in the adder, and the NOR plane that generates the sum of the product terms.

The NOR logic on the left side of Figure 1 composes product terms using PFET NW transistors. The right side of the design uses NFET transistors to turn the corner and directs the terms to a PFET-based NOR plane to calculate the sum of the product terms for the addition logic. With NOR-NOR logic, we can express arbitrary logic functions.

Despite the high density of NW devices, the NW-FET based logic arrays require buffers between the NOR planes. This is because it is impossible to provide ground lines interleaved with the nano-wires (for pull-down evaluation). As a result, large portions of the circuit, perpendicular to the diagonal defined by the NOR planes, cannot be utilized for active devices. This “diagonal” effect, however, can be avoided by replacing the second NOR plane with diode-based OR logic and avoiding the buffers previously needed to turn the corner.

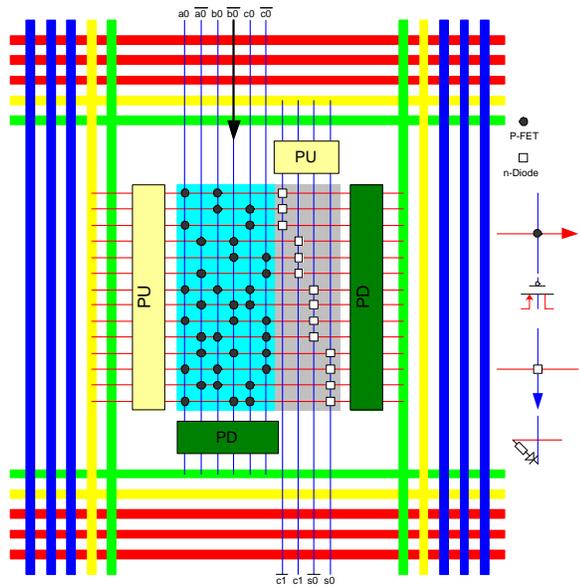


Fig. 2. Optimized 1-bit adder realized with NOR-OR NW logic.

The optimized adder shown in Figure 2, replaces the NFET buffer and PFET logic that were used previously to calculate the sum of the product terms, with an equivalent diode-logic based plane. The input NOR part of the circuit remains the same as in the previous design. The use of the diode-logic plane effectively removes the area overhead of the NFET buffer and moves the final sum calculation into a vertically positioned logic block that significantly improves fabric utilization. While this is a non-restoring logic, the outputs eventually drive the next stage input NOR plane where the product terms are fully restored by a pullup or a pulldown network.

### 3. CHALLENGES IN DESIGNING SEQUENTIAL CIRCUITS

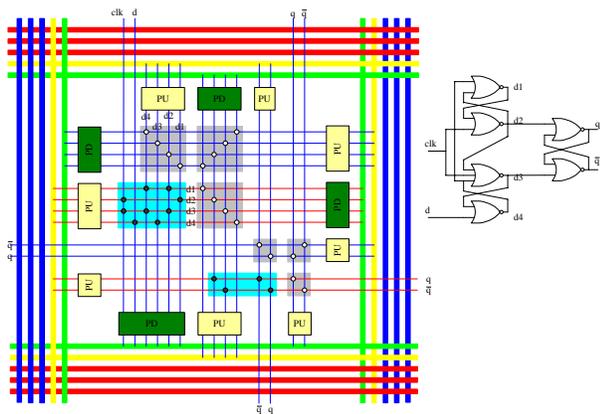


Fig. 3. 1-bit Flip-flop design with NWs.

Figure 3 presents an example of a flip-flop circuit that could be used to provide small storage locally. The design illustrates the challenges when incorporating sequential circuits with NW/CNT-based devices in a grid-based fabric. The

relatively low area utilization is due to a feedback path that is realized with three non-inverting NFET blocks that take the  $d1, d2, d3, d4$  signals back to the input of the flip-flop. The figure also shows an additional area (see logic in the right corner) that is required to route the outputs  $q, \bar{q}$  (bottom-right corner) of the flip-flop in each dimension – often necessary in practice. One insight from this is that using latches or pipelined circuits is difficult.

One of the most common design elements in any processor design is the datapath. Figure 4 shows a simple datapath which combines an optimized adder circuit (left-upper corner) and the flip-flop (middle) together, and shows the routing of the outputs in the four directions, to enable inter-tile cascading. Note that the area is defined primarily by the flip-flop. There is an additional area (right-bottom corner in the figure) required to realize the wiring necessary to route the outputs in several directions. In practice, a better utilization is possible if the combinational part of the tile is more complex.

As seen, the flip-flop causes the design to move in a diagonal shape and significantly reduces the effective density of the design. Any additional logic in the tile would be pushed into a diagonal shape with much of the nanofabric density lost.

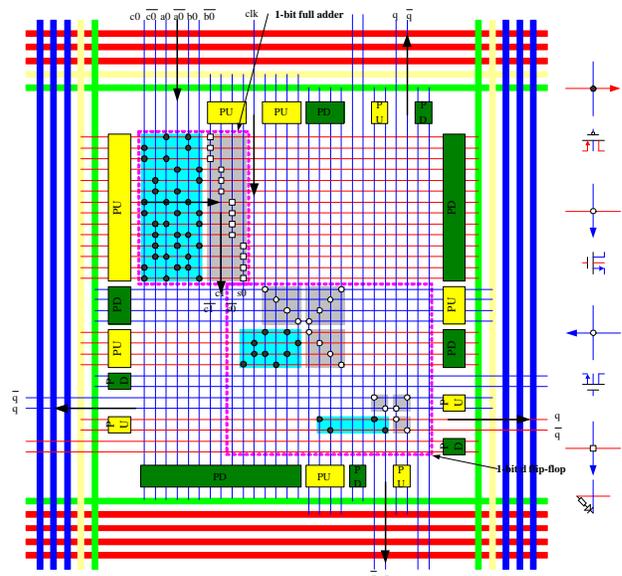


Fig. 4. 1-bit data-path with NOR-OR NW logic.

### 4. PROPOSED DYNAMIC DESIGNS

A key issue when building NASIC tiles, is the low effective density of latch circuits. This is due to the difficulty of building sequential circuits on a nano-grid. Moreover, latches require regions of differently doped nanowires along a dimension. While scientists have demonstrated differential doping [12], this could still be a limiting aspect.

Moreover, the circuits shown earlier are all based on static ratioed logic. There are several problems with these circuits. First, static ratioed circuits require careful sizing of devices for correct logic, that sets additional constraints on manufacturing. In addition, static power consumption due to direct-path currents would limit the scaling of NASIC tiles to arbitrary sizes.

In this section, we show how to use dynamically cascaded circuits adapted to the nano-grid to alleviate these problems.

Conventional MOS designs apply various techniques, such as adding an inverter between the cascaded dynamic circuits, also called Domino logic, to guarantee functional correctness. However, we have found that the solutions that have been proposed for MOS (see [1] for an overview) are not suitable or difficult to realize once the constraints of the nano-grid (e.g., the two-level logic, the grid layout and doping related) are taken into consideration.

We propose a dynamic circuit style which requires only one type of doping in each dimension. This design also enables register-like behavior on the nano-wires, without explicit latching, reducing the need for using the latch circuit presented earlier and enabling pipelined circuits.

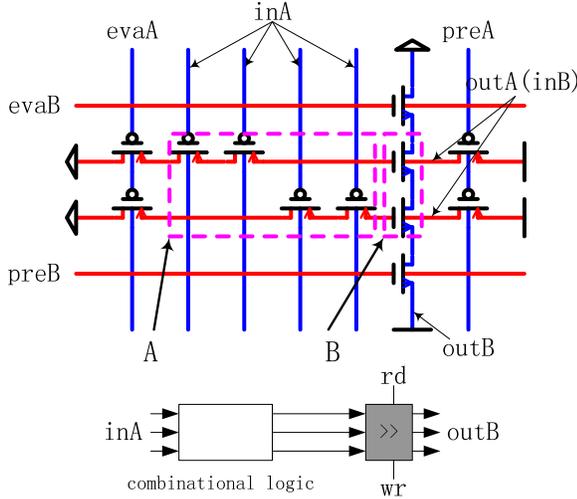


Fig. 5. Dynamic combinational circuit and nano-latch

As shown in Figure 5, instead of using static pullup and pulldown arrays, we add dynamic precharge and evaluate transistors to control a precharge and an evaluate phase for each clock cycle. A logic function is implemented with a combination of an OR-NAND logic that are evaluated successively. The OR part corresponds to A and the NAND part to section B in Figure 5. Note that both A and B are using a precharge-evaluate sequence to generate an output, but they together form a pipeline consisting of two stages. A pipelined NASIC circuit (see Figure 7) can be built by cascading several circuits in this fashion. A novel aspect of our design for dynamic NASIC circuits, is the addition of the *hold phase* that is used to enable correct cascading. In addition, this phase also enables temporary storage of output values between circuits in NASIC tiles. Figure 6 shows a waveform that includes the precharge-evaluate-hold phases for both A and B. For example, the hold phase for A requires both preA and evaA to be high to switch-off the corresponding transistors used for precharging and evaluation.

In the precharge phase, the value of outA (that is the same as inB) is precharged to high voltage. In the evaluate phase, the circuit is discharging depending on the logic inputs inA. In the hold phase, the value of outA (or inB) is kept, because

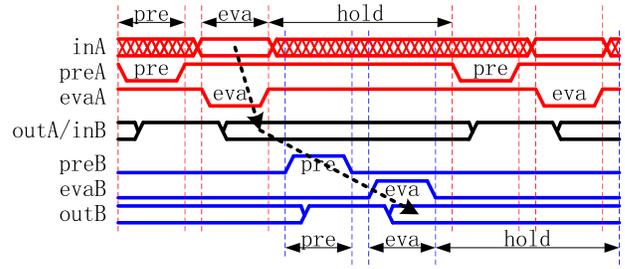


Fig. 6. Waveform showing the signals necessary to control a dynamic NASIC circuit and the nano-latch.

the input of the successive circuit is capacitively loaded, and preA and evaA are switched off.

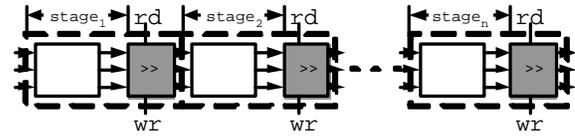


Fig. 7. Block diagram of a pipelined circuit with stages separated by nano-latches.

There are several advantages to our proposed dynamic style for NASIC designs. First, we can implement an arbitrary 2-level logic with a simpler doping process. In these designs all horizontal and vertical nanowires will have the same doping. Second, and perhaps the most interesting aspect is that *we can latch data on nano-wires* without explicitly adding sequential circuits. We call this temporary storage on the wire the *Nano-Latch*. This is very useful as it can provide a way to realize a pipelined structure without the problems we mentioned earlier with static and sequential circuit styles.

As a matter of fact the proposed dynamic circuit has the benefit that it can keep a value in a certain node for several cycles. Referring to Figure 6, if setting both preA and evaA to high, section A could stay in a hold phase storing a previously computed output for several cycles. The output of A is preserved. We can read this value by driving the precharge and evaluate transistors again. We can note that the precharge and evaluate signals of section A are write-enable controls and those of section B are read-enable controls for the nano-latch between A and B.

Of course the value stored in a nano-latch (e.g., outA) could be lost due to leakage currents if section A stays in a hold phase for too long. In conventional MOS technology, keeper devices are used to prevent the values stored in DRAM cells to be lost. We are currently investigating such circuits for the nano latches. An alternative, is to realize the keeper devices at microscale. Since a keeper device can serve a large number of nano devices, this strategy doesn't impact much on the area density.

#### 4.1. Pipelined Adder

The example in Figure 8 shows a 2-bit pipelined full adder that avoids using sequential circuit based latches. Our

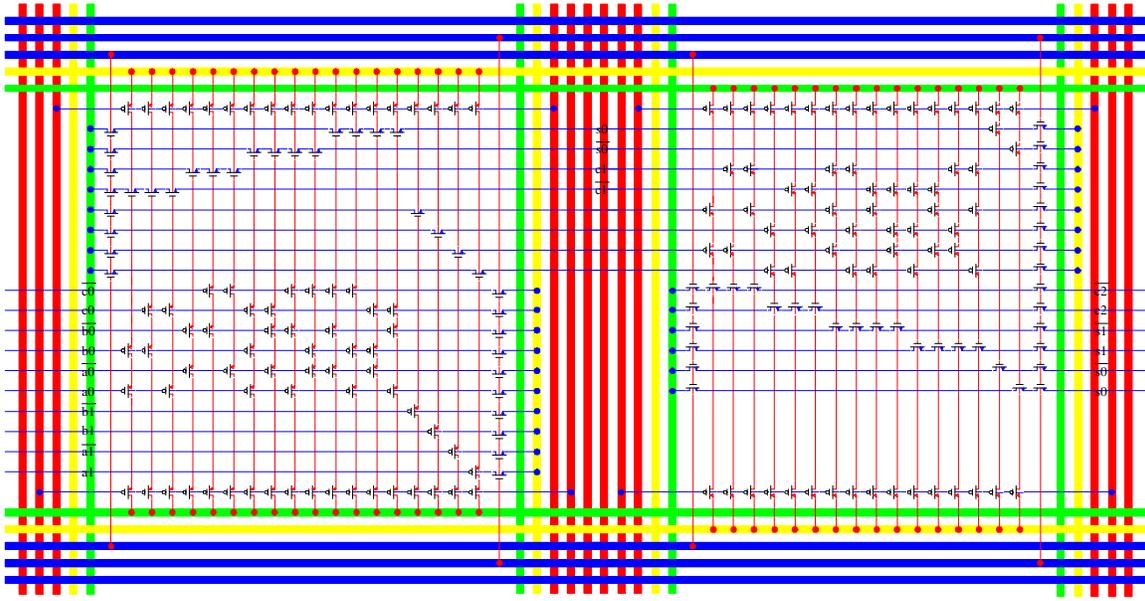


Fig. 8. Pipelined 2-bit Full Adder.

objective is to show a simple example of a pipelined circuit with the proposed dynamic design. Referring to Figure 8, we demonstrate how to split a 2-bit ripple adder into 2 pipeline stages by using the Nano Latch concept and dynamic circuit style at nano scale. Signals  $a_0, b_0, c_0$  are calculated first in the first stage, but  $a_1, b_1$  are shifted instead to the next pipeline stage. In the second stage,  $a_1, b_1$  and  $c_1$  (from stage 1) are calculated.  $s_0$  is also pipelined and input to the second stage. In this way, complex computations can be split into several simpler stages.

## 5. WIRE-STREAMING PROCESSORS

To apply the proposed techniques above, we design a simple but complete stream processor developed on 2-D nanowire fabrics called WISP-0. WISP-0 implements a typical 5-stage pipeline architecture including *fetch*, *decode*, *register file*, *execute* and *write back*. WISP-0 consists of five nanotiles, as shown in the floorplan in Figure 9. A nanotile is shown in the figure as a box surrounded by dashed lines. NWs are used to provide communication between adjacent nanotiles. Each nanotile is driven by the surrounding MWs, which are not shown in the figure. These nanotiles are designed in dynamic style and are cascaded together with nanowires to form a 5-stage pipeline. The proposed implicit nano-latches are used to support pipelining in WISP-0.

The five nanotiles work as follows. The *PC* block implements the program counter, which generates a 4-bit address each cycle. The *ROM* tile is a read-only memory structure that can store up to 16 7-bit instructions. One instruction is fetched from the *ROM* every cycle based on the address generated by the *PC*. The instruction then is decoded into opcode and operands in the *DEC* (decoder) block. Next, the values of the operands are read from the 4-entry register file (the *RF* block in the figure), and sent to the *ALU* for the execution of the

instruction. The result will be written back to the register file if necessary.

Currently, WISP-0 supports five instructions: *nop*, *mov*, *movi*, *add* and *mult*. The instruction format is shown in Figure 10: each instruction contains a 3-bit opcode and two 2-bit operands. The first operand could also be used as the destination address if necessary. The *nop* instruction is included to avoid RAW (read after write) data hazards, because control logics and bypass networks are difficult to implement on 2-D fabrics and would have very significant overall density impact due to the diagonal problem shown earlier. In WISP-0, we assume that the compiler has the ability to create codes without data hazards by inserting *nop*'s.

Next, we will present the detailed design of each block in WISP-0. For each nanotile, the surrounding MWs are not shown in the following figures to improve the readability. The pull-up/down networks are omitted as well.

The Program Counter (*PC*), as shown in Figure 11, implements a 4-bit accumulator. It is composed of a 4-bit incremter (bottom half in the layout) and a 4-bit nano-latch (top half in the layout). The output of the incremter is delayed for one cycle in the nano-latch and fed back to the incremter in the next cycle. The instruction address, which is the output of the incremter, is therefore increased by one in each cycle. This block shows an example of how to implement feedback networks in a single nanotile. We can implement an arbitrary finite state machine this way.

Program codes are stored in the *ROM* block, in which programs can not be modified during runtime. Figure 12 shows an example of the instruction ROM; the corresponding codes stored are shown on the right side. As we mentioned before, many *nop*'s are inserted in the code to avoid data hazards. The *DEC* block takes instructions from *ROM* and decodes them into opcodes and operands. Figure 13 shows the layout of the decoder.

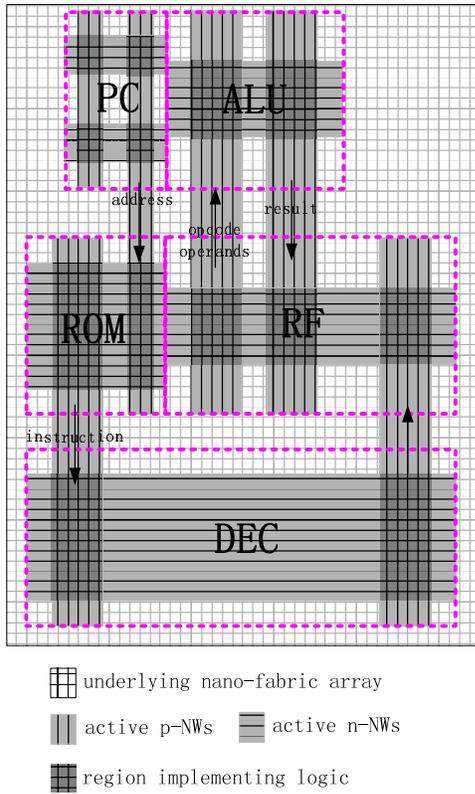


Fig. 9. The floorplan of WISP-0

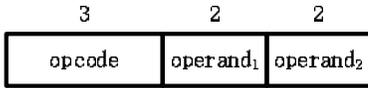


Fig. 10. The instruction format in WISP-0. Each instruction has a 3-bit opcode and two 2-bit operands.

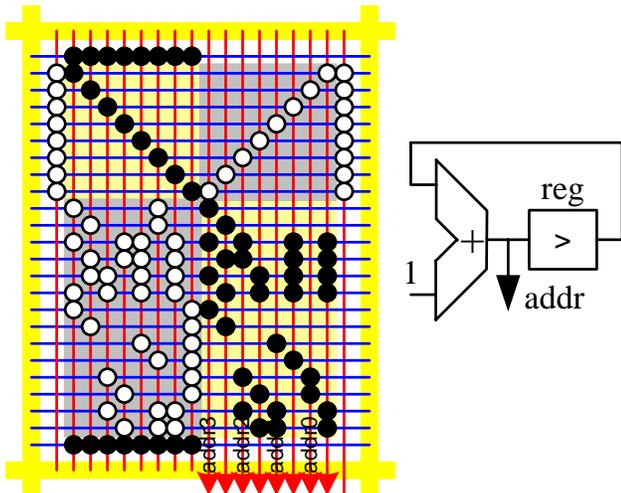


Fig. 11. The layout and schematic of the program counter in WISP-0. Black dots represent p-FETs and white dots represent n-FETs.

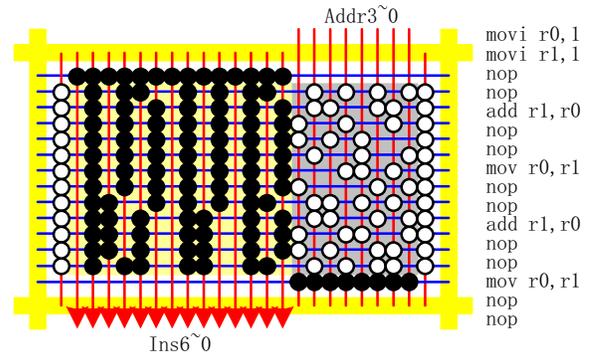


Fig. 12. An example of ROM layout in WISP-0. Program codes stored are shown on the right side.

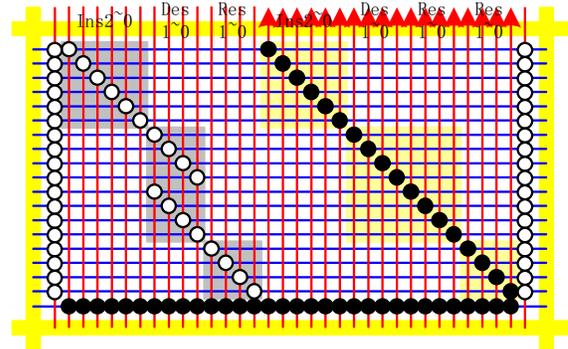


Fig. 13. The layout of the decoder in WISP-0.

Figure 14 shows the design of a 4-entry 2-bit register file. The data (in both true and complementary formats) are stored on the bottom 16 horizontal NWs in the figure, which implements a 16-bit implicit nano-latch. The values of the registers are selected by the 2-bit 4-to-1 multiplexer (*2-bit MUX41* in the figure), and another multiplexer (*2-bit MUX21*) is used to choose between immediate values (for *movi*) and register values. At the same time, *opcode* and *dest* (destination register address) are pipelined to *ALU*. If *ALU* needs to write results back to the register file, the data and register address will enter from the top right corner of the tile and the corresponding register value is updated by the logic in the bottom right.

Figure 15 shows the layout and schematic of *ALU* that implements both addition and multiplication functions. The arithmetic unit integrates an adder and multiplier together to save area. It takes the inputs (at the bottom) from the register file and produces the write-back result. At the same time, the write-back address is decoded by the 2-4 decoder on the top and transmitted to the register file along with the result. The result will be written in the corresponding register in the next cycle.

## 6. DENSITY EVALUATION

The key advantage of nanoscale devices is their density. However, as we mentioned before, without new circuit and architecture approaches, this density advantage could be lost due

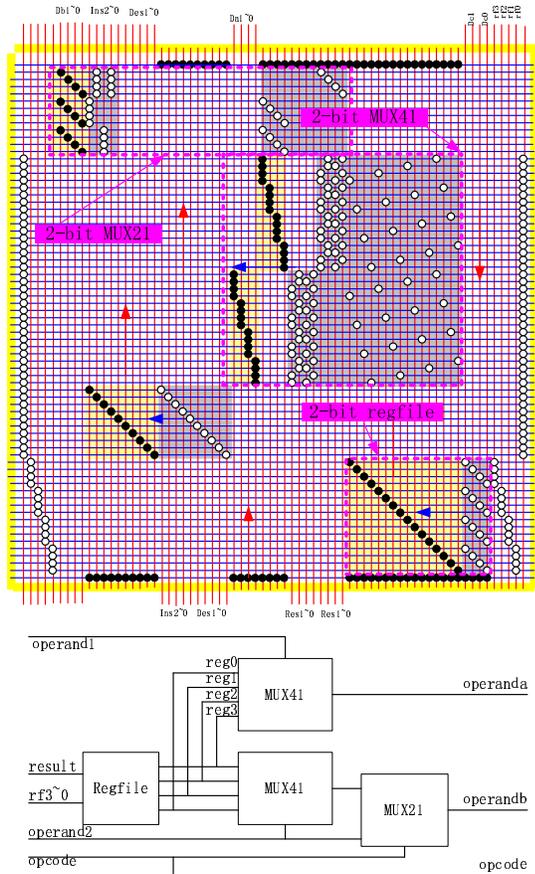


Fig. 14. Register file in WISP-0. It has four 2-bit registers.

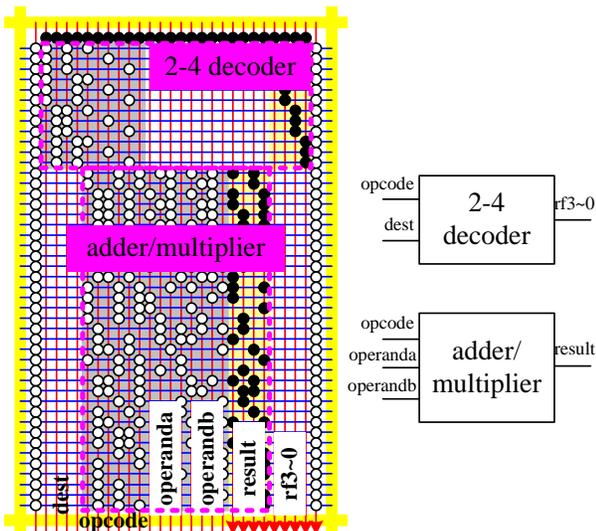


Fig. 15. The layout and schematic of the ALU in WISP-0.

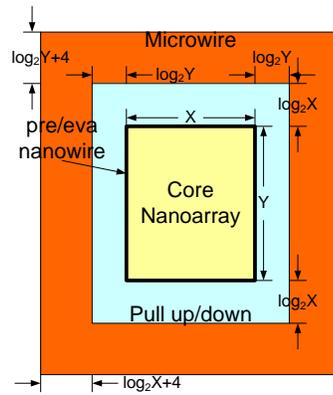


Fig. 16. Area breakdown of a typical nanotile. This figure shows the minimum size of pull-up/down networks and MWs.

Table 1. Area breakdown of each tile in WISP-0 ( $\mu m^2$ ).

Area	Core	PU/PD	MW	Total	Area Efficiency
PC	0.05	0.05	1.16	1.26	4.12%
ROM	0.04	0.04	1.01	1.09	3.81%
DEC	0.07	0.08	1.35	1.49	4.58%
ALU	0.14	0.10	1.54	1.78	7.93%
RF	0.53	0.22	2.66	3.41	15.60%
Total	0.84	0.49	7.72	9.04	9.24%

to manufacturing, fabric and device constraints when building nanoscale systems [25]. To validate our new architecture and techniques working around these constraints, we make an initial density evaluation on WISP-0 and compare its density to a 30-nm CMOS design.

We implemented a CMOS prototype with the functionality of WISP-0 in Verilog. We use the Design Compiler from Synopsys to synthesize it with 180-nm technology. The total area obtained is  $4,098 \mu m^2$ . We scale this value down to 30-nm (might be expected in 10 years based on the semiconductor industry roadmap) to make a fair comparison. The area is  $113 \mu m^2$  in 30-nm CMOS technology.

To calculate the area of WISP-0, each nanotile is partitioned into three parts: core nanoarray, pull up/down networks and microwires, as shown in Figure 16. Assuming that the size of a core nanoarray is  $X * Y$ , we use  $2 \log_2 Y$  vertical and  $2 \log_2 X$  horizontal NWs as pull-up/down networks plus  $2 \log_2 X + 4$  vertical and  $2 \log_2 Y + 4$  horizontal MWs (4 MWs as power supply and ground). The total area for a  $X * Y$  nanotile is:

$$[X + 2 \log_2 Y + s * (\log_2 X + 4)] * [Y + 2 \log_2 X + s * (\log_2 Y + 4)].$$

In the expression above,  $s$  represents the pitch ratio between MWs and NWs. We assume that the pitch of NWs is 10nm and  $s = 9$ . This is a relatively conservative assumption for the rapidly improving nanodevice technology [15].

We calculate the area for each of the nanotiles in WISP-0, as shown in Table 1. The total area of WISP-0 is  $9.04 \mu m^2$ . Compared to the area of CMOS design we have shown above, the density ratio between WISP-0 and the CMOS prototype is  $113/9.04 = 12.5$ .

This ratio is not that exciting because nanoarrays take only

$0.84\mu\text{m}^2$  of the total area while the area of MWs is almost ten times larger. The area efficiency for the core nanoarray ( $Area_{Core}/Area_{Total}$ ) is only 9.24%. The reason is that most of the tiles in WISP-0 have very small sizes, for example, the nanoarray size of PC is only 18X24. With larger nanotiles, however, this efficiency can be improved significantly because the number of MWs is logarithmic to the number of NWs. As we can see from the table, the largest tile RF (70X72) has a much higher area efficiency (more than 15%) compared to other smaller tiles.

In practice, a typical nanotile would have around  $1,000 \times 1,000$  NWs inside, and the area efficiency would go up to about 76% with the corresponding density ratio increasing to 103X.

Note that this still assumes a defect-free fabric. We expect that this ratio will decrease once fault tolerance techniques are added. Currently, we are exploring built-in fault tolerance at the circuit level. In WISP, since every signal is generated in both original and complementary forms, the system already provides some redundancy.

## 7. CONCLUSION

With CMOS technology approaching fundamental limits, the focus will increasingly shift to nanoelectronics based architectures. Recent academic and industrial activities and successes at the device level support this trend. This paper presents several ideas to improve the density of NW based designs.

We show a complete design of a simple stream processor called WISP-0 and compare its density with an implementation in 30-nm CMOS technology. Our initial evaluation shows that WISP-0 achieves a 12.5X density advantage on a defect-free fabric. We project that larger designs, with lower microwire overhead, could however be 100X denser.

## REFERENCES

- [1] Jan M. Rabaey, "Digital Integrated Circuits - A Design Perspective", Prentice Hall, 1996.
- [2] P. Wong, "Panel Session on Nanotechnology," *IEEE International Symposium on Signal Processing*, San Diego, 2002.
- [3] [http://www.hpl.hp.com/2002/oct-dec/beyond\\_silicon.html](http://www.hpl.hp.com/2002/oct-dec/beyond_silicon.html).
- [4] R.F. Service, "Molecules Get Wired," *Science* **294**, 2442 (2001).
- [5] G.Y. Tseng and J.C. Ellenbogen, "Toward Nanocomputers," *Science* **294**, 1293 (2001).
- [6] Y. Huang, X. Duan, Y. Cui, L.J. Lauhon, K-Y. Kim, and C.M. Lieber, "Logic Gates and Computation from Assembled Nanowire Building Blocks," *Science* **294**, 1313 (2001).
- [7] J.A. Hutchby, G.I. Bourianoff, J.V. Zhirmov, and J.E. Brewer, "Extending the Road Beyond CMOS," *IEEE Circuits and Devices* **18**, 28 (2002).
- [8] T. Rueckes, K. Kim, E. Joselevich, G.Y. Tseng, C-L. Cheung, and C.M. Lieber, "Carbon Nanotube-Based Nonvolatile Random Access Memory for Molecular Computing," *Science* **289**, 94 (2000).
- [9] A. DeHon, "Array-Based Architecture for Molecular Electronics," *First Workshop on Non-Silicon Computation NSC-1*, (2002).
- [10] S.C. Goldstein and M. Budi, "Nanofabrics: Spatial Computing Using Molecular Electronics," *Proceedings of the 28th Annual International Symposium on Computer Architecture* (2001).
- [11] R. Martel, V. Derycke, J. Appenzeller, S. Wind, and Ph. Avouris, "Carbon Nanotube Field-Effect Transistors and Logic Circuits," *DAC 2002 New Orleans*, ACM (2002).
- [12] L.J. Lauhon, M.S. Gudiksen, D. Wang, and C. M. Lieber, "Epitaxial core-shell and core-multi-shell nanowire heterostructures", *Nature* **420**, 57, 2002.

- [13] D. C. Duffy, J. C. McDonald, O. J. A. Schueller, and G. M. Whitesides, "Rapid Prototyping of Microfluidic Systems in Poly(dimethylsiloxane)", *Anal. Chem.* **70**, 4974, 1998.
- [14] Ph. Avouris R. Martel, V. Derycke, J. Appenzeller, "Carbon Nanotube Transistors and Logic Circuits," *PhysicaB*, Vol. 323, pp6-14, October 2002.
- [15] A. DeHon, "Array-Based Architecture for FET-Based, Nanoscale Electronics," *IEEE Transactions on Nanotechnology*, Vol. 2, No. 1, March 2003.
- [16] A. DeHon, "Stochastic Assembly of Sublithographic Nanoscale Interfaces" *IEEE Transactions on Nanotechnology*, Vol. 2, No. 3, Sept 2003.
- [17] Yi Cui, X. Duan, and C. Lieber, "Doping and Electrical Transport in Silicon Nanowires", *The Journal of Physical Chemistry*, Vol. 104, June, 2000.
- [18] A. Bachtold, P. Hadley, T. Nakanishi, C. Dekker, "Logic Circuits with Carbon Nanotube Transistors," *Science* **294**, 1317, 2001.
- [19] M. Mishra, and S.C. Goldstein, "Scalable Defect Tolerance for Molecular Electronics", *First Workshop on Non-Silicon Computations (NSC-1)*, 2002.
- [20] C. A. Moritz, D. Yeung, A. Agarwal, "SimpleFit: A Framework for Analyzing Design Tradeoffs in Raw Architectures," *IEEE Transactions on Parallel and Distributed Systems*, June 2001.
- [21] J. C. Eble III. "A Generic System Simulator with Novel On-Chip Cache and Throughput Models For Gigascale Integration," *PhD thesis, Georgia Institute for Technology*, November 1998.
- [22] A San Francisco Consulting Group Report. "Seven of the Most Promising Nanotechnology Companies -Report," Voted by Attendees Nanotech Venture Fair 2002 2002
- [23] H. B. Bakoglu, "Circuits, Interconnects and Packaging for VLSI," *Reading, MA: Addison Wesley*, 1990.
- [24] A. DeHon, "Array Based Architectures for Molecular Electronics," *First Workshop on Non-Silicon Computations (NSC-1)*, 2002.
- [25] T. Wang, Z. Qi, and C. A. Moritz, "Opportunities and challenges in application-tuned circuits and architectures based on nanodevices", in *First ACM International Conference On Computing Frontiers*, pp.503-511, Apr 2004.
- [26] Y. Cui, Z. Zhong, D. Wang, WU Wang, and CM Lieber, "High Performance Silicon Nanowire Field Effect Transistors" *Nano Letters*, Vol.3, pp.149-152, 2003.
- [27] Z. Zhong, D. Wang, Y. Cui, M. M. W. Bockrath, and C. M. Lieber, "Nanowire Crossbar Arrays as Address Decoders for Integrated Nanosystems" *Science*, Vol. 302, 2003, p. 1377.