# Wire-Streaming Processors on 2-D Nanowire Fabrics

Teng Wang, Mahmoud Ben-Naser, Yao Guo, Csaba Andras Moritz

Electrical and Computer Engineering Department
University of Massachusetts Amherst, MA, USA
{twang, mbennase, yaoguo, andras}@ecs.umass.edu

## ABSTRACT

Most of the research in the field of nanoelectronics has been focused on nanodevices and fabrication aspects and as a result a variety of nanodevice technologies have been demonstrated. By contrast, very little work has been reported on the design and evaluation of circuits and computational architectures using nanodevices. There is similarly not much work on the impact of device and fabric (e.g., the 2-D nanowire array) properties on computing. In this paper, we focus on computing architectures based on silicon nanowires. We explore a simple stream processor developed on 2-D nanowire fabrics and compare its density to a 30nm CMOS implementation. We also identify techniques to work around fabric-specific constraints. Our initial evaluation shows that this stream processor has great density advantage compared to CMOS technology.

***Keywords***: nanoscale circuits, architecture, NASIC, silicon nanowires, carbon nanotubes

## 1 INTRODUCTION

Nanotechnology is one of the most promising replacements for CMOS technology. Perhaps the most exciting nanodevices today for nanoscale integrated circuits are semiconductor nanowires (NWs) and arrays of crossed carbon nanotubes (CNTs). Researchers have already built FETs and diodes out of NWs [4]–[6] and CNTs [8]. While there are many practical challenges still remaining, it seems that it will soon be possible to build regular nanoarrays from uniform-length CNTs or NWs [7]. By contrast, we have seen very little work on the design and evaluation of circuits and computational architectures using nanodevices. Similarly, little work on the impact of device and fabric properties on computing has been made so far.

Our previous work [1]–[3] has addressed some of the challenges and technical constraints when building computing systems on 2-D nanowire fabrics. These challenges include:

- The 2-D regular NW array, where doping is fixed in each direction, significantly impacts the density of circuits due to the diagonal problem, when the logic is cascaded, only the diagonal portion of the nanotile is utilized [1].
- Having 2-level logic instead of multi-level on 2-D nanoarrays reduces the density further.
- Control circuits and bypass networks are difficult to implement. This is because building feedback paths on 2-D nanoarray is challenging.
- Similarly, it is hard to design high-density sequential circuits using traditional MOS-like approaches, because they require feedback paths.

In this paper, we present a complete design of a simple stream processor developed on 2-D nanowire fabrics, called WISP-0. WISP-0 is the first version of our *Wire-Streaming Processors* (WISP). In WISP, in order to preserve the density advantages of nanodevices, data are streamed through the fabric with minimal control/feedback paths. Intermediate values produced during processing are often stored on the nanowires without requiring explicit latching.

We also compare the density of a fully implemented CMOS design and the NW-based WISP-0. Our evaluation shows that it is possible to preserve the density advantages of nanodevices in WISP despite the fabric constraints.

## 2 OVERVIEW OF NASICS

The WISP architecture proposed here is a key part of our effort to build NASICs: *Nanoscale Application-Specific Integrated Circuits* [1]–[3]. NASICs are based on extensive research on understanding emerging device and fabrication constraints. A NASIC design has a hierarchical and tiled architecture and it is optimized to deal with various manufacturing, fabric and device constraints. Next, We briefly describe the key components of NASICs. More details can be found in our previous papers [1]–[3].

### 2.1 Nanotiles

Nanotiles are the basic building blocks of NASICs. Figure 1 shows a typical nanotile. The orthogonally crossed nanowires form a nanoarray. The junctions can be programmed as FETs or detached [7]. Nanoarrays

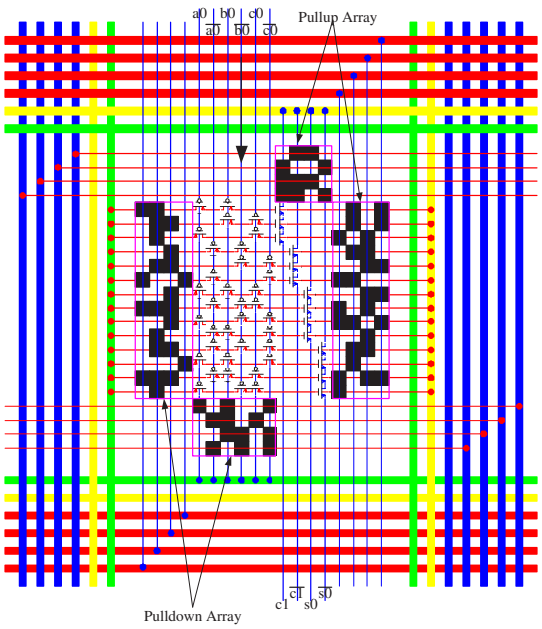


Figure 1: A nanotile for a 1-bit full adder. The thicker wires are MWs and the thinner wires are NWs. NWs in different directions have different doping types.

are surrounded by microwires (MWs). Each signal is expressed in its both original and complementary forms. Microwires provide signals and power supply. Pull-up/down arrays act as the interface between MWs and NWs. To program each NW junction, the number of MWs must be at least logarithmic to the number of NWs [7].

## 2.2 Dynamic Nanotiles and Pipeline

One of the most common components in any processor design is the datapath. Registers or latches are required to pipeline the data flow. Due to topological and doping constraints, latch circuits are however very difficult to implement on nanoarrays. In NASICs, we use a new dynamic circuit style to achieve temporary storage in stead of using explicit flip-flops. A pipelined NASIC circuit can be built by cascading dynamic nanotiles without explicit latching of signals. Use of latch circuits would have affected density considerably due to the diagonal problem [1]–[3].

## 2.3 Interconnect and Multi-tile Designs

A nanoscale processor may have thousands of nanotiles. Achieving efficient communication between nanotiles is a critical design aspect. In NASICs, local communication between adjacent nanotiles is provided by NWs for area efficiency. MWs are used for global communications only.

# 3 WISP-0

WISP-0 is an initial version of our wire-streaming processors. It exercises many of the design strategies

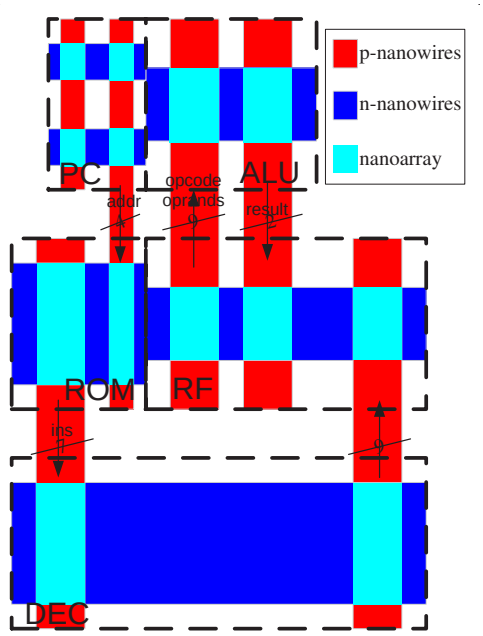


Figure 2: The floorplan of WISP-0

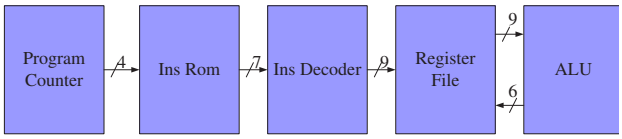


Figure 3: The schematic of WISP-0

and optimizations in NASICs. WISP-0 contains a 2-bit datapath and a 3-bit opcode. Figure 2 shows the overall layout of WISP-0. In this figure, each box surrounded by dashed lines represents a nanotile. All adjacent nanotiles are connected by a set of NWs. These nanotiles are all designed in dynamic style and are cascaded together "on the wire" to form a 5-stage pipeline: fetch, decode, register file, execute and write back. No explicit latches are used. Figure 3 is the schematic of WISP-0.

The *PC* block implements the program counter. It generates a 4-bit address for each cycle. *ROM* is the instruction ROM. It fetches one instruction according to the address from *PC*. The instruction goes to *DEC* (decoder) and is decoded into opcode and operands. Next, the opcode and operands enter *RF* (register file) stage and read the value of operands from the registers. The instruction will be executed in *ALU* and the result will be written back to the register file.

Control logics and bypass networks are difficult to implement on 2-D fabrics. Carefully selected ISA minimizes these circuits. Currently WISP-0 supports the following instructions: *nop*, *mov*, *movi*, *add* and *mult*. All fields in these instructions have fixed lengths (like RISC) in order to simplify the design of the decoder and ALU.

Due to limited space, we only include three key blocks: *PC*, *RF* and *ALU*. MWs are not shown in the following

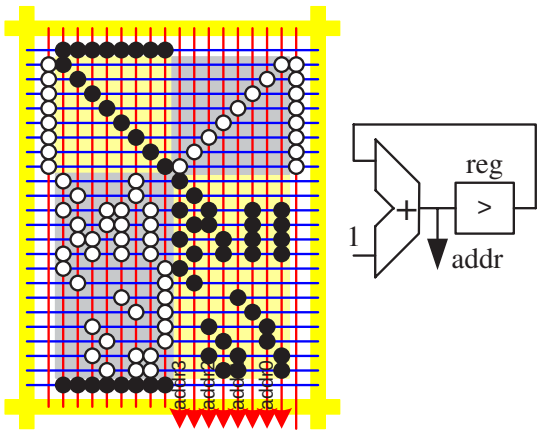


Figure 4: The layout and schematic of the program counter in WISP-0. Black dots represent p-FET and white dots represent n-FET.

figures to improve the readability. For simplicity, the pull-up/down networks are also omitted.

Figure 4 is the layout and schematic of the program counter. This block implements a 4-bit accumulator in a nanotile. It has two components: a 4-bit incrementer (bottom half in the layout) and a 4-bit latch (top half in the layout). In each cycle, the output of the incrementer is delayed and fed back to the input. The address is therefore increased by one in each cycle.

The design of the register file is shown in Figure 5 and Figure 6 is its schematic. In this block, data are stored on the 16 horizontal NWs at the bottom. They are selected by the 2-bit 4-to-1 multiplexer (*2-bit MUX41* in the figure) by $operand_1$ and $operand_2$ respectively. The data ($operand_a$) read out by $operand_1$ is sent directly to "ALU". The data read out by $operand_2$ is sent to another multiplexer (*2-bit MUX21*). This data and $operand_2$ are selected by *opcode* to produce $operand_b$. The reason is that some instructions (e.g., *movi*) will use the immediate data provided by the instruction instead of the values from registers. 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 control signals will enter from the top right corner of the tile and update the values on the bottom 16 horizontal NWs.

Figure 7 shows the layout of ALU in WISP-0. This block executes the instructions and generates the result (*result* in the figure) and control signals (*rf3~0*). The top part (*2-4 decoder*) is a 2-4 decoder. It selects the register to be written back according to the destination address (*dest*). The bottom part (*adder/multiplier*) represents an arithmetic unit. It calculates the summation or product (decided by *opcode*) of $operand_a$ and $operand_b$.

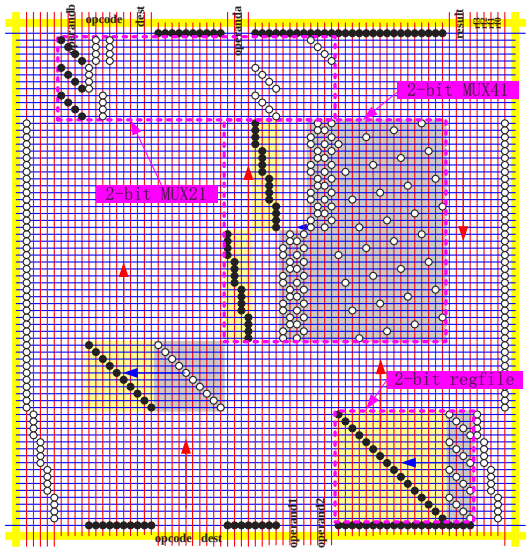


Figure 5: Register file in WISP-0. It has four 2-bit registers.

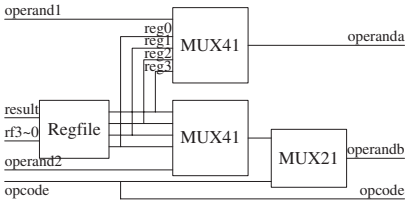


Figure 6: The schematic of register file.

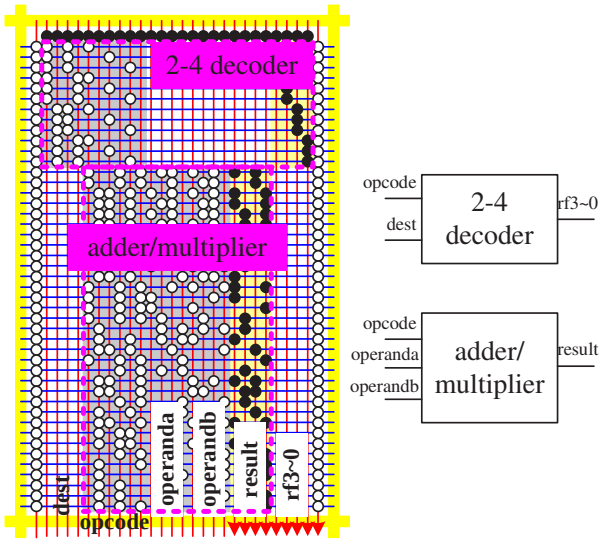


Figure 7: The layout and schematic of the ALU in WISP-0.

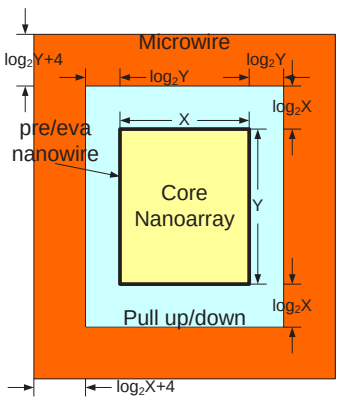


Figure 8: Area breakdown of a typical nanotile. This figure shows the minimum size of pull-up/down networks and MWs.

## 4 DENSITY EVALUATION

The key advantage of nanoscale devices is their density. However, we have found that without new circuit and architecture approaches, this density advantage could be lost due to manufacturing, fabric and device constraints when building nanoscale systems [1]–[3]. In this section, we make an initial density evaluation on WISP-0.

First let us analyze a typical nanotile in NASIC. Figure 8 shows the area breakdown. Assuming that the size of a nanoarray is $X * Y$, we need at least $2log_2Y$ vertical and $2log_2X$ horizontal NWs as pull-up/down networks plus $2log_2X+4$ vertical and $2log_2Y+4$ horizontal MWs (4 MWs as power supply and ground). The total area becomes: $[X+2log_2Y+s*(log_2X+4)]*[Y+2log_2X+s*(log_2Y+4)]$.

In this expression, $s$ is the pitch ratio of MWs to 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. We use this expression to calculate all blocks in WISP-0. The total area is $9.04\mu m^2$. Among this area, nanoarrays take only $0.77\mu m^2$ while the overhead of MWs dominates. The area efficiency ($Area_{nanoarray}/Area_{total}$) is only 8.5%. With larger nanotiles, however, this efficiency can be improved significantly as will be shown next.

To compare with CMOS technology, we implemented a CMOS prototype in Verilog. We use the Design Compiler from Synopsys to synthesize it with 180nm technology. The total area is $4{,}098\mu m^2$. We scale this value down to 30nm (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 30nm CMOS technology.

The density ratio between WISP-0 and the CMOS prototype is $113/9.04 = 12.5$. But, as we mentioned before, MWs take up most area in WISP-0. Because the number of MWs is logarithmic to the number of NWs, when the design is larger, the area percentage of MWs goes down. In practice, a typical nanotile would have around 1,000×1,000 NWs inside, and the area efficiency would go up to 76% and the corresponding density ratio to 115.

## 5 CONCLUSION

In this paper, we have shown a complete design of a simple stream processor and compared its density with an implementation in 30nm CMOS technology. Our research indicates that we successfully preserve the density advantage of nanodevices in WISP despite fabric constraints.

Our future work includes fault tolerance on nanoarrays. This is especially important since nanodevices will likely have a higher defect rate than CMOS devices. Currently we are focusing on built-in fault tolerance at circuit and architecture levels. In WISP, since every signal is generated in both original and complementary forms, the system already provides some redundancy.

## REFERENCES

[1] 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.

[2] C. A. Moritz and T. Wang, "Latching on the Wire and Pipelining in Nanoscale Designs", in Non-Silicon Computing Workshop (NSC-3), ISCA-31, pp.39-45, Jun 2004.

[3] T. Wang and C. A. Moritz, "NASIC: Nanoscale Application-Specific ICs and Architectures", Boston Area Architecture Workshop, BARC'04, Boston, MA, Jan 2004

[4] 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).

[5] 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.

[6] 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.

[7] A. DeHon, "Array-Based Architecture for FET-Based, Nanoscale Electronics" IEEE Transactions on Nanotechnology, Vol. 2, No. 1, pp.23-32, Mar 2003.

[8] 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).