MATRIX: A Reconfigurable Computing Device with Configurable Instruction Distribution (Extended Abstract)

Ethan Mirsky
(eamirsky@ai.mit.edu)

Andre DeHon†
(andre@ai.mit.edu)

MIT AI Lab
545 Technology Square, Cambridge, MA 02139

The MATRIX chip represents a novel, reconfigurable computing architecture which supports configurable instruction distribution. Device resources are allocated to controlling and describing the computation on a per-task basis. Application-specific regularity and parallelism allows us to compress the resources allocated to instruction control and distribution, in many situations yielding more resources for datapaths and computations. This flexibility is made possible by a multi-level configuration scheme, a unified configurable network supporting both datapath and instruction distribution, and a coarse-grained building block which can serve as an instruction store, a memory element, or a computational element.

MATRIX (Figures 1 and 2) is composed of an array of 36 identical basic functions units (BFUs) surrounded by 12, 10-bit configurable I/O ports (8 bits of data, 2 bits of control). Each BFU (Figure 3) contains a 256x8-bit memory, an 8-bit ALU and multiply unit, and reduction control logic that allows data-dependent decision-making. The BFU can be configured to operate as a memory for instruction or data storage, a datapath compute unit, or an ALU-register file combination. In addition, the BFU can be chained together to produce 16, 24, 32 or wider datapaths.

The input ports (see Figure 3) select network data values to feed and control the BFU. Note that the BFU function control (e.g. ALU operation selection), memory addresses, and data input ports all take their inputs from the same unified data network. This allows a data byte on any network line to serve as instruction or data.

The configurable network overlaying the BFUs consists of three mesh-like structures, resembling traditional FPGA interconnect: A local network, a length-4 bypass interconnect, and global lines spanning the columns and rows. Unlike FPGA interconnect, however, these connections are all 8 bits wide, and are dynamically controllable - an application can alter its interconnect on a cycle-by-cycle basis during run-time. The network can thus be controlled by a long instruction like a VLIW processor or statically configured like an FPGA. All of the network lines support both instruction and data distribution allowing the interconnect wires to be allocated as needed.

For example, Figure 4 shows three possible implementations of a convolution filter on the MATRIX chip. In the first example, the BFUs are configured as a systolic datapath with no dynamic control. This runs at the highest possible throughput - 1 result every other clock cycle (each multiply takes 2 cycles). However, it also consumes most of the chip area in the process (4 BFUs per TAP).

The second example shows some of the same BFUs configured as a small microcoded engine, with an 8-bit ALU/register file, program counter, and instruction memories. This is a general microcoded execution engine capable of executing any sequence of less than 256 microinstructions with full, cycle-by-cycle control

†. Andre DeHon may now be contacted at: Soda Hall #1776, U.C. Berkeley, Berkeley, CA 94720-1776

over the BFU's resources. When performing the convolution filter, this engine requires 1/30 as much area as the systolic implementation (it can run up to 60 TAPs), but it runs much slower - 1 result every 57 clock cycles. However, if that throughput is sufficient for the application, this implementation frees up a great deal of area that can be used for other operations.

The third example shows the same BFUs configured in a VLIW-like structure. In this example, we're exploiting some of the parallelism inherent in the algorithm by allocating separate units to perform the multiply, the add, and the pointer manipulations. In this manner we can trade off a small increase in area (3 additional BFUs) for much higher throughput - 1 result every 13 cycles, running up to 64 TAPs.

Many other variations on these theme are possible. The power of the MATRIX architecture is its ability to deploy these resources based on application regularity, throughput, and space available. In contrast, traditional microprocessors, VLIW or SIMD machines fix the assignment of control resources, memory, and datapath flow at fabrication time, while traditional programmable logic does not support the high-speed reuse of functional units needed to perform irregular tasks.

The initial prototype, currently being fabricated on Hewlett-Packard's CMOS14TB process (0.6μm), is expected to run at 50 MHz, providing 1.8 billion 8-bit operations per second. Based on our experience with MATRIX, we believe that the architecture can easily scale to 100-200 MHz clock speeds and 10x10, or larger, arrays.

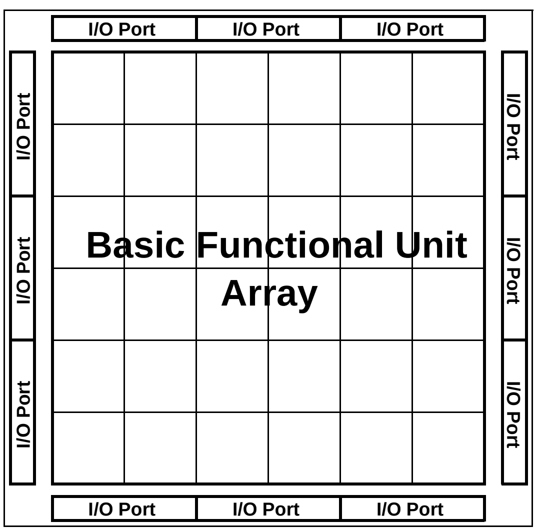

Figure 1: MATRIX Chip Block Diagram

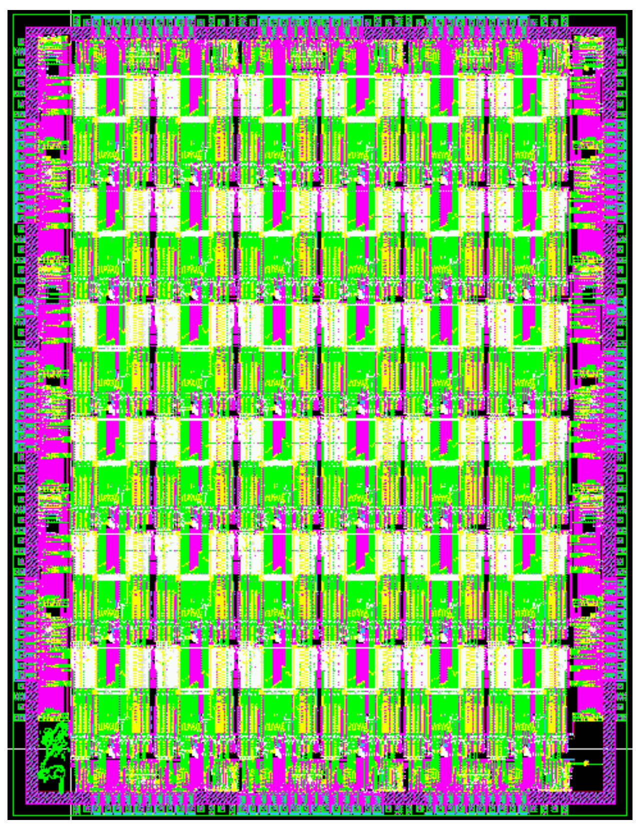

Figure 2: MATRIX Chip Layout

Figure 3: MATRIX Basic Functional Unit (BFU)

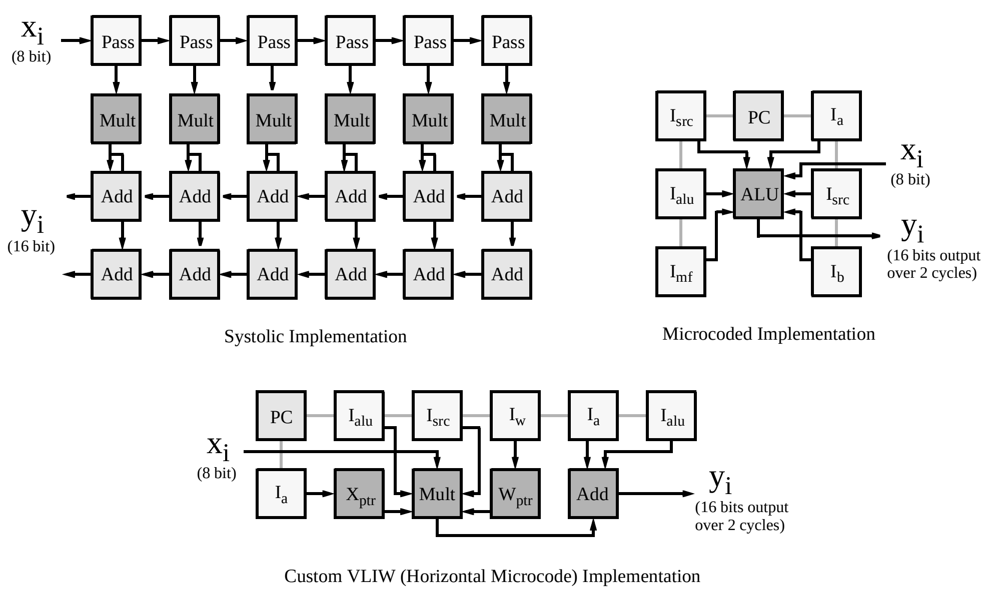

Figure 4: MATRIX Application Examples (Convolution)