VIRAM-1 Vector Datapath

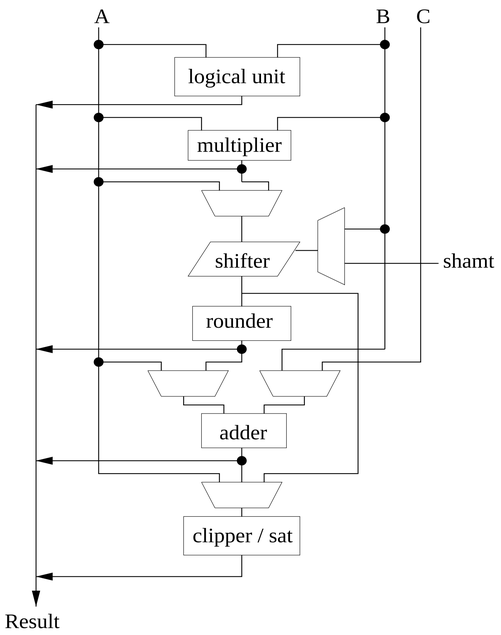
University of California, Berkeley
Joseph Gebis
gebis@cs.berkeley.edu

Description

VIRAM-1 is a vector microprocessor currently being designed at the University of California, Berkeley. The vector datapath contains the arithmetic and logical units that are used to perform vector calculations.

This poster contains information about the vector datapath and its current status.

Datapath Block Diagram



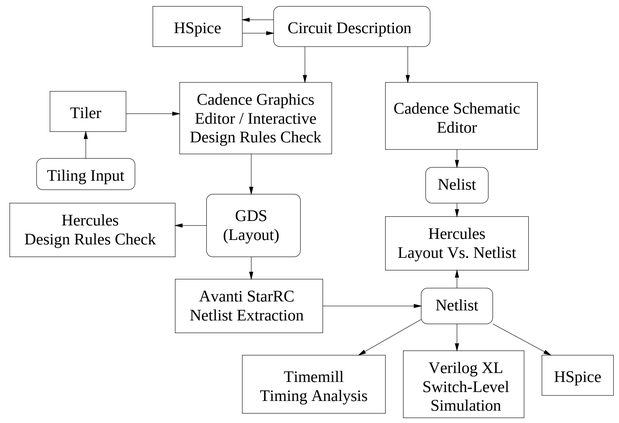
The datapath contains the following modules:

- Logical unit, for performing AND/OR-type logical operations
- Multiplier
- Shifter
- Rounder, capable of rounding in the following modes:
  - Truncate
  - Round Up
  - Jam
  - Round to Nearest Even
- Adder
- Clipper / Saturation

The VIRAM-1 datapath is fully pipelined and partitionable down to 16 bits. This means it is capable of operating on one set of 64-bit operands, two sets of 32-bit operands, or four sets of 16-bit operands.

Design Steps

Simplified Custom Layout CAD Flow



The design process comprises many steps, all of which must be completed to ensure the the design will perform the necessary operations and operate as intended. This poster describes the process for only the vector datapath, which is being designed with custom logic; other parts of the microprocessor are being designed using macro blocks, synthesized logic, and other methods of design. The process is, in many steps, the same, but there are some significant differences as well.

Design Input

The process begins with a description of a datapath block. A circuit is designed to implement the functionality of the block; a switch-level simulator (such as Verilog) may be used to test the design of complicated blocks, or even design larger blocks using smaller blocks (such as full adders) as primitives.

The bottom-level circuit design is an iterative process; various logic families and circuit designs are tested and compared for power, size, and speed. Finally, a circuit with sized transistors is chosen that satisfies the requirements of the circuit.

Once the circuits are designed, they can be layed out. A graphics editor, such as Cadence, is used to do this; repetitive segments may use tiled versions of smaller blocks to build large circuits. The graphics editor has a built-in tool to perform Design Rules Check that ensure that the design is able to be fabricated.

After the layout is completed, the design data is exported as a GDS file for use with other tools. At this point, the design is theoretically complete (the GDS file is used to generate the masks that are actually used to fabricate the chip), but many verification steps still remain.

Verification

Verification occurs at every stage in the design process.

The first verification step in design input occurs when the design is being captured in the graphics editor. An online design rules checker is used to check that no rules are being violated. Once the design is complete and exported from the graphics editor, another design rules checker (such as Hercules) is used; this additional check is performed using a rule set from the fabrication plant.

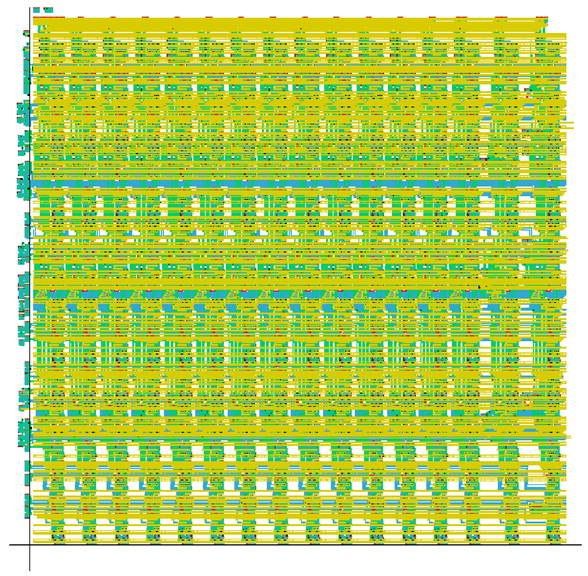
The original circuit description is also used to generate schematics of the circuits. A netlist from the schematic is compared to a netlist generated from the layout to ensure that they are the same, and the captured circuits match the intended design.

The netlist is used to execute many switch-level simulation cycles using a tool such as Verilog. This is used to show that the functionality of the designed circuit matches the original description.

Besides switch-level functionality, timing behavior needs to be verified. A timing tool such as Timemill is run on the entire circuit to verify timing operations. Further analysis can be performed using HSpice on critical paths (to get more accurate information).

Finally, tools such as Powermill can be used to make sure that power consumption does not cause any problems in the design.

Current Layout



The layout of the basic block of the multiplier (which is repeated sixteen times to create the full multiplier) is complete and shown above. The circuits for all other blocks are complete, as is the layout of the gates and cells they comprise.