PipeRench: Power and Performance Evaluation of a Programmable Pipelined Datapath

Benjamin A. Levine and Herman H. Schmit

Dept. of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA USA 15213
blevine@cmu.edu, herman@ece.cmu.edu
Reconfigurable Computing Device

A Computing Device which can be reconfigured for each different application that it runs, by changing the functionality of its hardware and the way that its hardware is connected.

PipeRench was developed by students and faculty at Carnegie Mellon.
Why Reconfigurable Computing?

More Design Effort

PERFORMANCE

ADAPTABILITY

ASIC
FPGA
DSP
CPU

RECONFIGURABLE COMPUTING
Why Reconfigurable Computing?

We want performance and adaptability:
– Performance of an ASIC

Implement application as custom datapath to:
  • Increase parallelism.
  • Decrease memory traffic (through locality).
  • Increase performance.
  • Use less power.

– Adaptability of a CPU.

  Completely reprogram as needed for new applications.
Why NOT Reconfigurable Computing?

• FPGA design is more like HW than SW
  – No real C to FPGA yet, so must use HDL

• FPGA configuration is fixed to one FPGA
  – Must redesign to gain performance on larger FPGAs
  – Can't use design on FPGA with fewer resources.
  – Compares poorly to SW for microprocessors:
    • No portability
    • No scalability
Solution: Use A Virtual Architecture

Compile ONCE!

Run Everywhere!

Through Ultra-high speed reconfiguration.

Different:
- Technology
- $$$
- Performance
Virtual Architecture

• Compile to virtual machine
  – Makes compilation easier
  – Compile from high-level language (DIL)
  – Binaries decoupled from specific hardware
  – Scalable / Re-usable

• Restrict the model of computation to pipelined datapaths
  – Makes virtual architecture possible
  – Simplifies compilation and programming
Pipelined Datapaths

```c
for (i=0; i<maxIn; i++)
{
    y[i]=0;
    for (j=0; j<Taps; j++)
    {
        y[i] += x[i+j]*w[j];
    }
}
```

Lots of apps fit:
- DSP
- Image Processing
- Cryptography
PipeRench Fabric

A programmable, pipelined data path containing:

Processing elements
Local interconnect
Pass Registers
Unbounded Depth
Pipeline Virtualization

Virtual Pipeline
Pipeline Virtualization

Since stripes are connected in a ring, data can always pass between adjacent virtual stripes in the physical fabric.
Performance Scaling

Real Device

2 Stripes
2 Outputs / 6 Cycles

4 Stripes
4 Outputs / 6 Cycles

6 Stripes
6 Outputs / 6 Cycles
Chip Structure

- Ring Structure
  - Interleaved

- Global Buses
  - Inputs
  - Outputs
  - Configuration
  - State Storage

- 16 Stripes
  - 16 PEs each
PE Architecture

Eight-bit Buses

This vertical bus connects to one horizontal wire, depending on which PE it is.

From Previous Stripe

Global Busses

Global Input Bus
State Restore Bus
Global Output Bus
State Store Bus

To Next Stripe

All wires are 8-bits unless otherwise noted
PE Architecture

All wires are 8-bits unless otherwise noted.
Flexible, Cascadable Shifters

All wires are 8-bits unless otherwise noted
PE Architecture

Eight-bit Cascadable Functional Units

This vertical bus connects to one horizontal wire, depending on which PE it is.

From Previous Stripe

Global Busses

To & From other PEs

To PE

To PE

From PE

To PE

All wires are 8-bits unless otherwise noted
Functional Unit Architecture
PE Architecture

All wires are 8-bits unless otherwise noted.

Pass Register File

From Previous Stripe

To Next Stripe

Global Busses

To & From other PEs

This vertical bus connects to one horizontal wire, depending on which PE it is.

Functional Unit

Register File

State Store Bus

State Restore Bus

Global Output Bus

Global Input Bus
Pass Register File

- Two values can be read in each stripe.
- PE can write one new value to a single register.
- Otherwise: each register writes value from previous stripe.
Pass Register File Operation

1. FU
2. FU
3. FU
4. FU
5. FU

last use

1  2  3  4  5
Pass Register Problem

Old register values cycle through fabric endlessly. Extra switching consumes power.
Solution: Register Kill

Configuration word

Configuration bits control which registers are read….

From Previous Stripe

Crossbar

FU

To Next Stripe
Solution: Register Kill

Configuration word

Crossbar

Configuration bits control which registers are read….

From Previous Stripe

To Next Stripe

FU
Solution: Register Kill

...they can also control which registers are passed...

From Previous Stripe

...and which are "killed" by clearing the register.

Crossbar

Variable in this register used later.

Last use for variable in this register

To Next Stripe
Solution: Register Kill

"Last use" bit controls whether read register are killed or passed

This requires only two more bits per PE and minimal additional HW.
Implemented Hardware Design

• Industrial Partner: ST Microelectronics
• Process Technology: 0.18 micron, 6 metal
  
  3.65 million transistors
  49 sq. mm die
  120 MHz fabric operation
  60 MHz I/O frequency
  < 3W power

Reconfigure entire fabric in 133ns.
Switch applications in 8ns.
Fabric Layout
Chip Die Shot
Performance on Filtering

• 40 Tap 16-bit FIR Filter
  – 41.8 MSPS

• Comparable to high-end DSPs
  – Much lower clock rate
  – Without a full multiplier
    (taps are compiled into hardware)
Performance on Encryption

• IDEA Encryption: 450 Mbps
  – Key is compiled into hardware
  – Compilation (including P&R) takes less than one minute

• Comparison:
  – 800 MHz Pentium III Xenon: 75.4 Mbps
Power Consumption – FIR Filter

- Before 14 taps, near constant power
- At 14 taps, virtualization causes step
Conclusions

• A practical virtual machine for pipelined programmable datapaths is possible.
• Virtual hardware $\Rightarrow$ physical hardware:
  – Completely self-managed on chip at run-time.
  – Enabled by fast incremental reconfiguration.
• Virtual architecture allows:
  – Easier compilation
  – Forward compatibility / Scalability
• Implemented chip has high performance and low power requirements