Designing Languages for Designing Hardware
Adrian Sampson
Cornell
How do we harness the power of computing?

What are the secrets of human intelligence?

How do computers work?

What is computation, really?

What is a computer?
What is computation?

What is a computer?
“The complexity for minimum component costs has increased at a rate of roughly a factor of two per year.”

Gordon Moore
co-founder of Intel
1965
The size of a single transistor decreases by half every 18 months.

Gordon Moore
co-founder of Intel
1965
and cost, and power...

The size of a single transistor decreases by half every 18 months.

Gordon Moore
co-founder of Intel
1965
EXPOENTIAL!
CPU core frequency vs year of introduction

- 1 MHz
- 10 MHz
- 100 MHz
- 1 GHz
- 10 GHz

Year of Introduction:
- 1970
- 1975
- 1981
- 1986
- 1992
- 1997
- 2003
- 2008
- 2014
free lunch

exponential single-threaded performance scaling!
free lunch

multicore era

we’ll scale the number of cores instead
FIGURE S.1  Processor performance from 1986 to 2008 as measured by the benchmark suite SPECint2000 and consensus targets from the International Technology Roadmap for Semiconductors for 2009 to 2020. The vertical scale is logarithmic. A break in the growth rate at around 2004 can be seen. Before 2004, processor performance was growing by a factor of about 100 per decade; since 2004, processor performance has been growing and is forecasted to grow by a factor of only about 2 per decade. An expectation gap is apparent. In 2010, this expectation gap for single-processor performance is about a factor of 10; by 2020, it will have grown to a factor of 1,000. Most sectors of the economy and society implicitly or explicitly expect computing to deliver steady, exponentially increasing performance, but as these graphs illustrate, traditional single-processor computing systems will not match expectations. Note that the SPEC benchmarks are a set of artificial workloads intended to measure a computer system's speed. A machine that achieves a SPEC benchmark score that is 30 percent faster than that of another machine should feel about 30 percent faster than the other machine on real workloads.
free lunch  multicore era  who knows?

time  2005  2015

immemorial  ?  ?  ?
The performance returns from Moore’s Law ended in 2015!

The only way forward is to trade off generality for efficiency!

A New Golden Age for Domain-Specific Hardware,
Enhanced Security, Open Instruction Set,
Agile Chip Development

John L. Hennessy and David A. Patterson
A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services

Andrew Putnam  Adrian M. Caulfield  Eric S. Chung  Derek Chiu
Kypros Constantinides  John Demme  Hadi Esmaeilzadeh  Jeremy Fowers
Gopi Prashanth Gopal  Jan Gray  Michael Haselman  Scott Hauck
Amir Hormati  Joo-Young Kim  Sitaram Lanka  James Larus
Simon Pope  Aaron Smith  Jason Thong  Phillip Yi Xiao  Doug Burger

Microsoft

Abstract

Datacenter workloads demand high computational capabilities, flexibility, power efficiency, and low cost. It is challenging to improve all of these factors simultaneously. To advance datacenter capabilities beyond what commodity server designs can provide, we have designed and built a composable, reconfigurable fabric to accelerate portions of large-scale software services. Each instantiation of the fabric consists of a 6x8 2-D torus of high-end Stratix V FPGAs embedded into a half-rack of 48 machines. One FPGA is placed into each server, accessible through PCIe, and wired directly to other FPGAs with pairs of 10 Gb SAS cables.

In this paper, we describe a medium-scale deployment of this fabric on a bed of 1,632 servers, and measure its efficacy in accelerating the Bing web search engine. We describe the requirements and architecture of the system, detail the desired function into the available reconfigurable area. One could virtualize the FPGA by reconfiguring it at run-time to support more functions than could fit into a single device. However, current reconfiguration times for standard FPGAs are too slow to make this approach practical. Multiple FPGAs are needed to suitably accelerate portions of large-scale software services. Each instantiation of the fabric consists of a 6x8 2-D torus of high-end Stratix V FPGAs embedded into a half-rack of 48 machines. One FPGA is placed into each server, accessible through PCIe, and wired directly to other FPGAs with pairs of 10 Gb SAS cables.

In this paper, we describe a medium-scale deployment of this fabric on a bed of 1,632 servers, and measure its efficacy in accelerating the Bing web search engine. We describe the requirements and architecture of the system, detail the improvements from general-purpose systems.
RTL
register-transfer level

Verilog  VHDL  Bluespec  Chisel

FPGA

Actual Silicon
FPGA
RTL (register-transfer level)

C

Assembly
CPU
C

High-Level Synthesis

RTL
register-transfer level
Image and video processing, financial analytics, bioinformatics, and scientific computing applications. Since RTL programming in VHDL or Verilog is unacceptable to most application software developers, it is essential to provide a highly automated compilation/synthesis flow from C/C++ to FPGAs.

As a result, a growing number of FPGA designs are...
Verilog is unacceptable \[\not=\] we must program FPGAs in C
HLS

An enormous series of ad hoc consistency checks, hacks, and workarounds to compile some C programs to Verilog.

Seashell

A new language for hardware accelerator design with a type system that defines which programs are realizable on FPGAs.
int A[10];
int B[10];
for (int i = 0; i < 10; i++) {
    int x = A[i];
    int y = x * 5;
    B[i] = y;
}
int A[10]

int B[10]

* loop body
int A[10]

int B[10]
#pragma HLS ARRAY_PARTITION variable=A factor=5
#pragma HLS ARRAY_PARTITION variable=B factor=5

int A[10];
int B[10];

for (int i = 0; i < 10; i++) {
    #pragma HLS UNROLL factor=5
    int x = A[i];
    int y = x * 5;
    B[i] = y;
}
int A[10]

int B[10]
```c
#include <stdio.h>

int A[10];
int B[10];

int main() {
    for (int i = 0; i < 10; i++) {
        #pragma HLS UNROLL factor=5
        int x = A[i];
        int y = x * 5;
        B[i] = y;
    }
    return 0;
}
```
Memory types

```c
memory A : int[10];
for (...) {
    access A[i];
    access A[i+1];
}
```
memory A : int[10];
for (...) {
    access A[i];
    access A[i+1]; // error: A already used in this context
}

Affine memory types
Affine types and linear types, as made famous recently by Rust.
Banked memory types

```c
memory bank(5) A : int[10];
```
Banked memory types

```python
memory bank(5) A : int[10];
for (let i in 0..1) {
    access A[0][i];
    access A[1][i];
    access A[2][i];
    access A[3][i];
    access A[4][i];
}
// one access to each A[j] allowed here
```

<table>
<thead>
<tr>
<th>A[0][0]</th>
<th>A[0][1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>A[1][0]</td>
<td>A[1][1]</td>
</tr>
</tbody>
</table>
Hybrid indices for unrolling

```r
memory bank(5) A : int[10];
for (let i in 0..9) unroll 5 {
  access A[??][??];
}
```

i : idx<0..5, 0..2>

A pair of a static index from 0 through 4 and a dynamic index that’s either 0 or 1.
Seashell

RTL
register-transfer level

FPGA

C
Assembly
github.com/cucapra/seashell