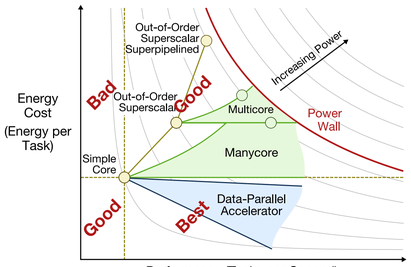
The Maven Vector-Thread Architecture

Yunsup Lee[1], Rimas Avižienis[1], Alex Bishara[1], Richard Xia[1], Derek Lockhart[2], Christopher Batten[2], Krste Asanović[1]

[1]Parallel Computing Laboratory, UC Berkeley
[2]Computer Systems Laboratory, Cornell University

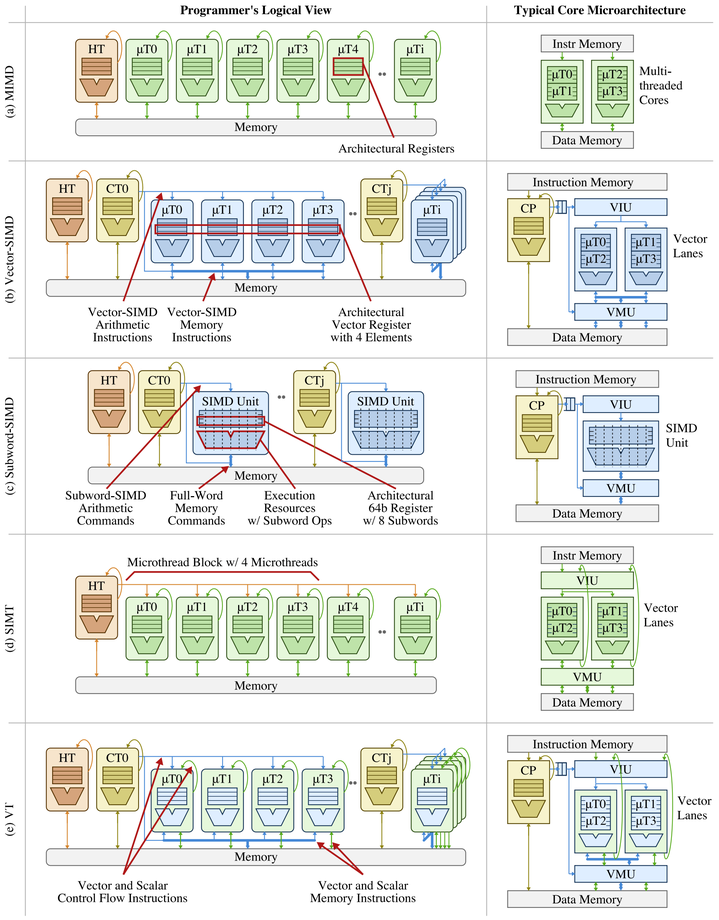
1 Motivation

Future manycore processors will be energy-constrained, and thus the primary metric for evaluating these architectures will be their **energy-efficiency**. In this work, we investigate new architectural and microarchitectural mechanisms which enable a wider array of applications to be mapped to energy-efficient vector units.



2 Architectural Patterns

Three different architectural patterns (excluding subword-SIMD and SIMT) were evaluated in terms of their performance, energy efficiency and area. Maven is a new vector-thread architecture, which is based on a vector-SIMD architecture, adds minimal hardware to support irregular DLP well, and is considerably simpler to implement than previous vector-thread designs.



3 Maven Programming Methodology

```
void vvadd_vt( int dest[], int src0[],
               int src1[], int size )
{
 int vlen = vt::set_vlen( size );
 for ( int i = 0; i < size; i += vlen )
 {
   vlen = vt::set_vlen( size - i );

   vt::HardwareVector<int> vsrc0, vsrc1;
   vsrc0.load( &src0[i] );
   vsrc1.load( &src1[i] );

   vt::HardwareVector<int> vdest;

   VT_VFETCH( (vdest), (vsrc0,vsrc1),
   ({
     vdest = vsrc0 + vsrc1;
   }));

   vdest.store( &dest[i] );
 }
}
```

Compiles to instruction which abstracts hardware vector length

C++ class that uses compiler vector types to enable automatic vector register allocation

Unit-stride vector loads

Compiles to separate function which is then vector fetched

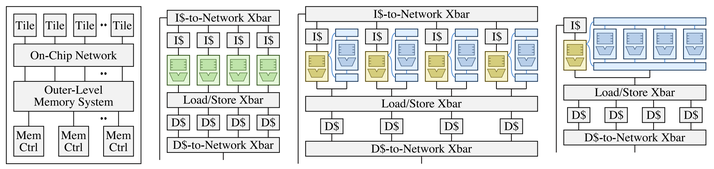
Inside vector fetched block vector registers appear as scalar values

Unit-stride vector store

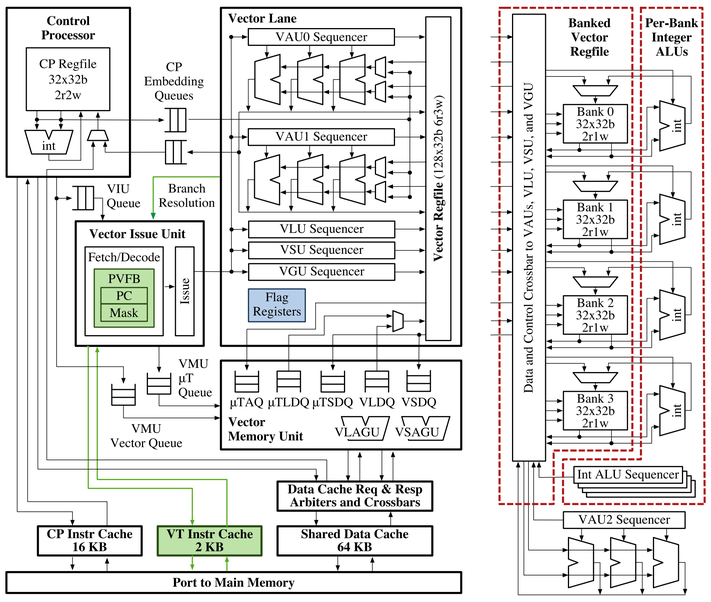
4 Maven Tile Microarchitecture

We focus on comparing the various architectural design patterns with respect to a single data-parallel tile. Example tiles are a MIMD tile, a vector tile with four single-lane cores, or one four-lane core. To manage complexity of many design points, we developed a library of parameterized synthesizable RTL components.

Tile Configurations

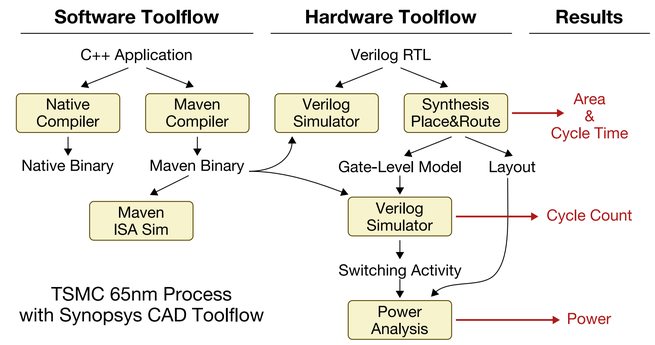


Maven Core Microarchitecture

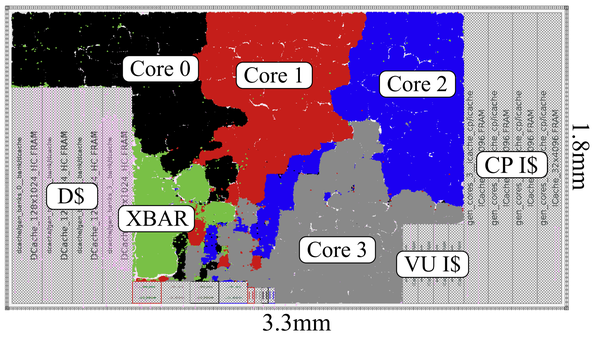


5 Evaluation Framework

Toolflow



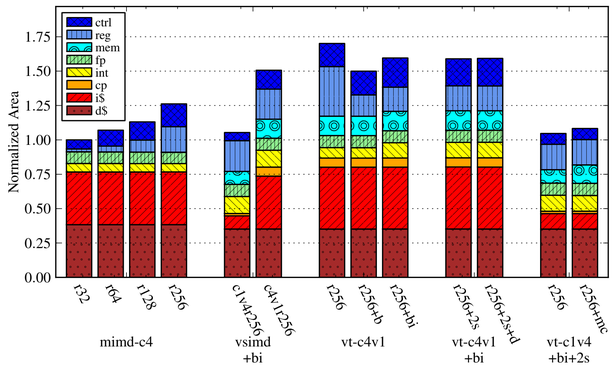
An Example VLSI Layout



6 Evaluation Results

We first compare tile configurations based on their cycle time and area before exploring the impact of various microarchitectural optimizations. We then compare implementation efficiency and performance of the Maven VT pattern against the MIMD, and vector-SIMD patterns for the six application kernel.

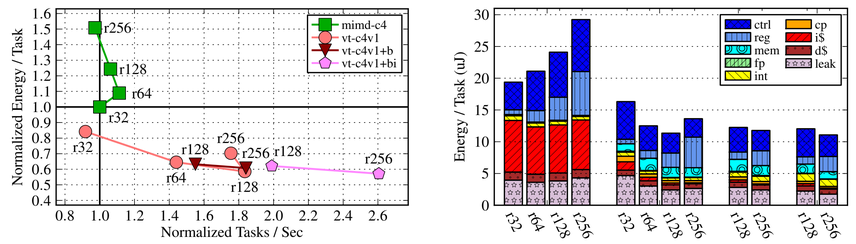
Area & Cycle Time



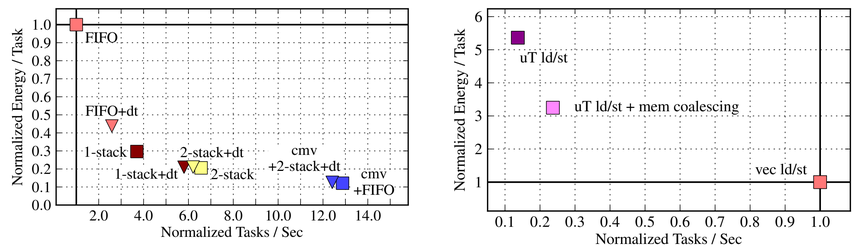
Configuration	Power Statistical /Simulated (mW)	Total Area (mm²)	Cycle Time (ns)
mimd-c4r32	149 /137-181	3.7	1.10
mimd-c4r64	216 /130-247	4.0	1.13
mimd-c4r128	242 /124-261	4.2	1.19
mimd-c4r256	299 /221-298	4.7	1.27
vsimd-c4v1r256+bi	396 /213-331	5.6	1.37
vsimd-c1v4r256+bi	224 /137-252	3.9	1.46
vt-c4v1r256	428 /162-318	6.3	1.47
vt-c4v1r256+b	404 /147-271	5.6	1.31
vt-c4v1r256+bi	445 /172-298	5.9	1.32
vt-c4v1r256+bi+2s	409 /225-304	5.9	1.32
vt-c4v1r256+bi+2s+d	410 /168-300	5.9	1.36
vt-c1v4r256+bi+2s	205 /111-167	3.9	1.42
vt-c1v4r256+bi+2s+mc	223 /118-173	4.0	1.42

Performance and Energy Efficiency

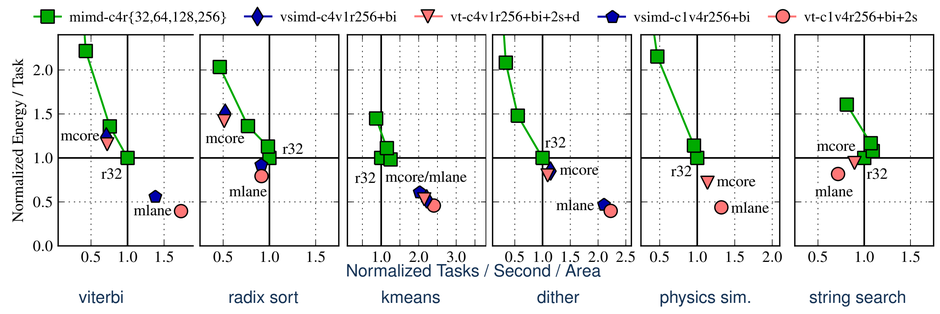
Impact of Additional Physical Registers, Intra-Lane Regfile Banking, and Additional Per-Bank Integer ALUs



Impact of Density-Time Execution and Stack-Based Convergence Schemes / Impact of Memory Coalescing



Implementation Efficiency and Performance for MIMD, vector-SIMD, and VT Patterns Running Application Kernels



7 Conclusions

1. The Maven vector-thread architecture is more area and energy efficient than MIMD architectures on regular DLP and (surprisingly) on irregular DLP
2. The Maven vector-thread architecture is a promising alternative to traditional vector-SIMD architectures, providing greater efficiency and easier programmability
3. Using real RTL implementations and a standard ASIC toolflow is necessary to compare energy-optimized future architectures

For more details, see our ISCA '11 paper below or use the QR code on the right with a barcode reader.
"Exploring the Tradeoffs between Programmability and Efficiency in Data-Parallel Accelerators"

This work was supported in part by Microsoft (Award #024263) and Intel (Award #024894, equipment donations) funding and by matching funding from U.C. Discovery (Award #DIG07-10227). The authors acknowledge and thank Jiongjia Fang and Ji Kim for their help writing application kernels, Christopher Celio for his help writing Maven software and developing the vector-SIMD instruction set, and Hidetaka Aoki for his early feedback on the Maven microarchitecture.

Yunsup Lee — 577C Soda Hall, Berkeley, CA 94720 — yunsup@cs.berkeley.edu — http://www.cs.berkeley.edu/~yunsup