The T0 Vector Microprocessor

Krste Asanovic
James Beck
Bertrand Irissou
Brian E. D. Kingsbury
Nelson Morgan
John Wawrzynek

University of California at Berkeley
and the
International Computer Science Institute

Primary support for this work was from the ONR, URI Grant N00014-92-J-1617, the NSF, grants MIP-8922354/MIP-9311980, and ARPA, contract number N0001493-C0249.

Additional support was provided by ICSI.

Talk Outline

Why Vector Microprocessors?

T0 Microarchitecture

T0 Implementation and Packaging

Summary

6.3-03

Goal:

High-Performance, Programmable, Scalable DSP Architecture.

Many new applications require high performance DSP, for example, multimedia and human-machine interface.

Algorithms are constantly changing, and not all applications warrant custom hardware development, so need programmable DSP engine.

Software development is a major expense. Desire object code compatibility while scaling parallelism up for performance, or scaling parallelism down for cost.

6.3-04

Solution:

Vector Instruction Set Architecture

Many compute-intensive DSP operations are vectorizable and vector architectures are the most efficient way to run vector code.

Low control complexity, even for highly parallel implementations.

High arithmetic performance through multiple parallel and pipelined functional units.

Sustainable high main memory bandwidth with vector memory operations.

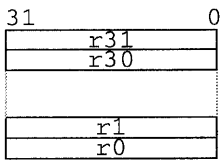
Can be added to conventional scalar instruction set to preserve software investment.

Vector instruction set architecture scalable between low cost and high performance implementations while remaining object-code compatible.

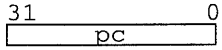
T0 User Programming Model

CPU

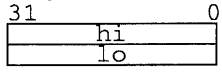
General Purpose Registers



Program Counter

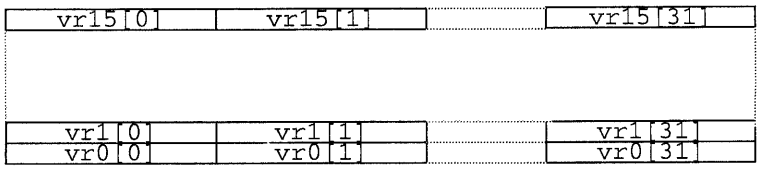


Multiply/Divide Registers

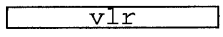


VU (COP2)

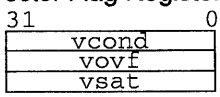
16 Vector Registers, each holding 32 x 32-bit elements.



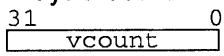
Vector Length Register



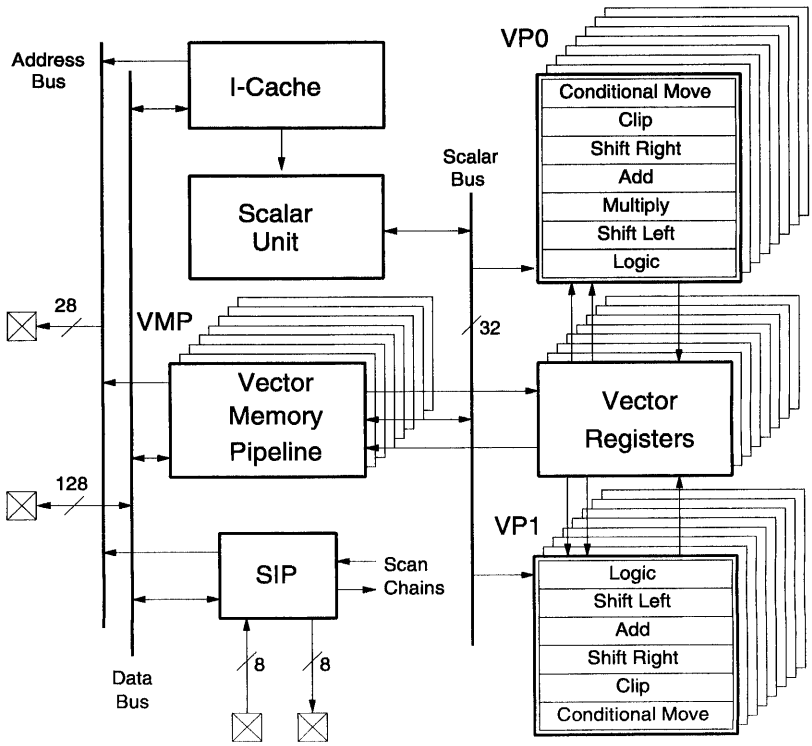
Vector Flag Registers



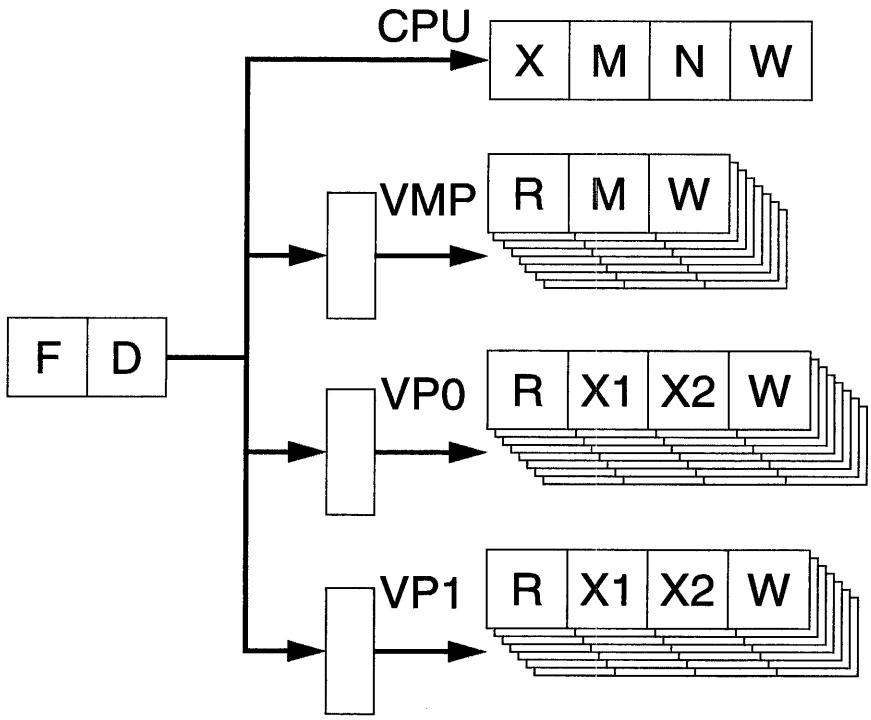
Cycle Counter



T0 Block Diagram



T0 Pipeline Structure

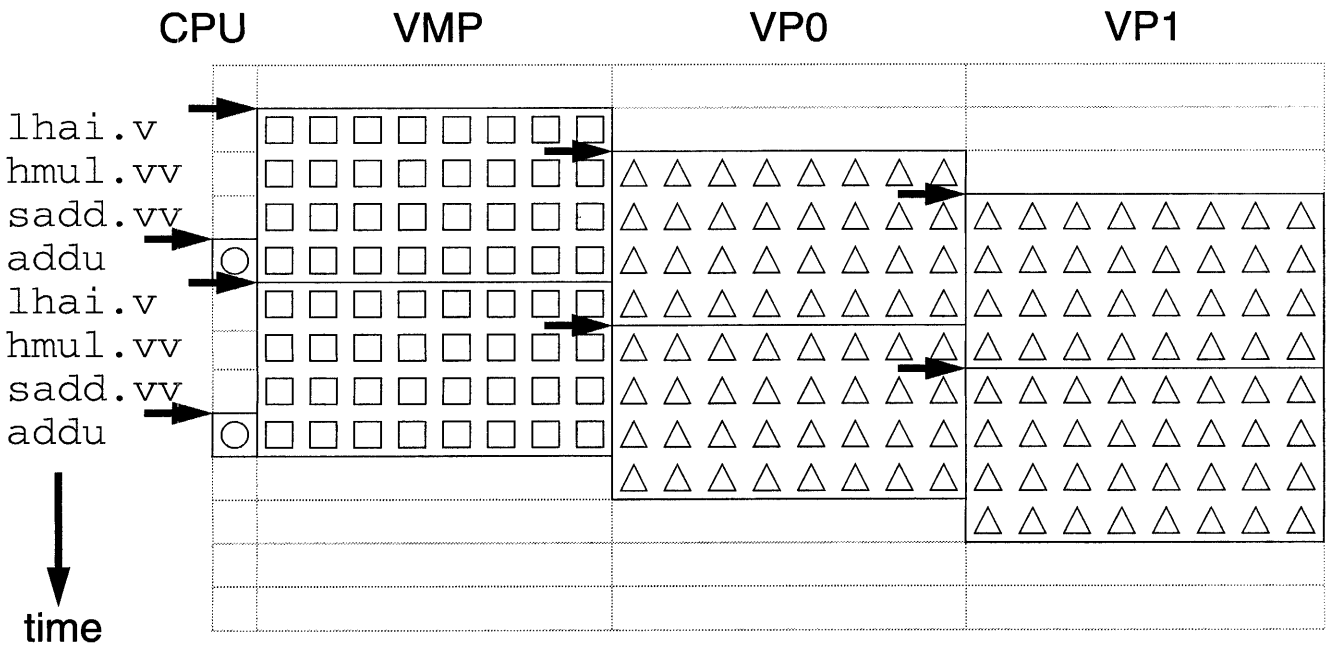


Code Example

```
lhai.v vv1, t0, t1      # Vector load.
hmul.vv vv4, vv2, vv3   # Vector mul.
sadd.vv vv7, vv5, vv7   # Vector add.
addu t2, -1             # Scalar add.
lhai.v vv2, t0, t1      # Vector load.
hmul.vv vv5, vv1, vv3   # Vector mul.
sadd.vv vv8, vv4, vv8   # Vector add.
addu t7, t4             # Scalar add.
```

(taken from matrix-vector multiply routine)

Execution of Code Example



T0 I-Cache and Scalar Unit

Instruction Cache

1 KB, direct-mapped, 16 byte lines.

Cache line prefetched if memory otherwise idle:

- 2 cycle miss penalty with prefetch,
- 3 cycle miss penalty without prefetch.

Cache misses in parallel with other interlocks.

System Coprocessor 0

Exception handling.

Host communication registers.

32-bit counter/timer.

MIPS-II 32-bit Integer RISC CPU

One instruction per cycle in 6 stage pipeline.

Single architected branch delay slot.

Annulling branch likelies.

Interlocked load delay slots.

3 cycle load latency (no data cache).

18 cycle integer multiply.

33 cycle integer divide.

6.3-11

T0 Vector Memory Operations

Unit-stride with address post-increment

```
lbai.v vv1, t0, t1  # t1 holds post-increment.
```

Eight 8-bit elements per cycle.

Eight 16-bit elements per cycle.

Four 32-bit elements per cycle.

+1 cycle if first element not aligned to 16 byte boundary.

Strided operations

```
lwst.v vv3, t0, t1  # t1 holds byte stride.
```

One 8-bit, 16-bit, or 32-bit element per cycle.

Indexed operations (scatter/gather)

```
shx.v vv1, t0, vv3  # vv3 holds byte offsets.
```

One 8-bit, 16-bit, or 32-bit element per cycle.

+ 3 cycle startup for first index.

Indexed stores need 1 extra cycle every 8 elements.

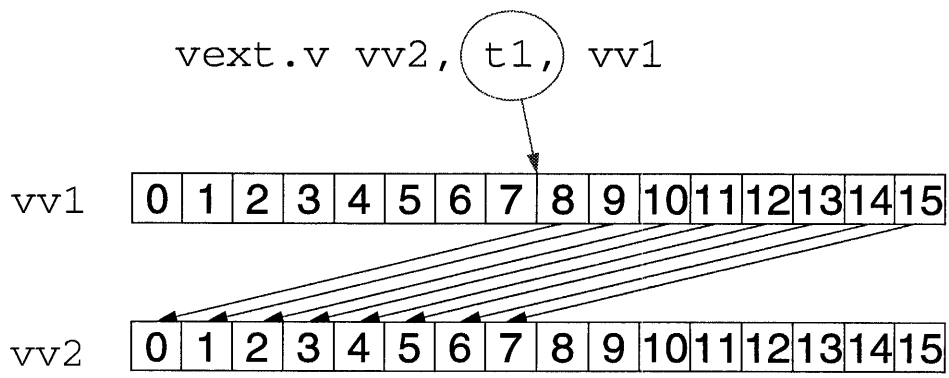
6.3-12

T0 Vector Editing Instructions

Executed in vector memory unit.

Scalar insert/extract to/from vector register element.

Vector extract supports reduction operations:



Separates data movement from arithmetic operations.

Avoids multiple memory accesses.

Software can schedule component instructions within reduction.

T0 Vector Arithmetic Pipelines

Full set of 32-bit integer vector instructions: add, shift, logical.

Vector fixed-point instructions perform a complete scaled, rounded, and clipped fixed-point arithmetic operation in one pass through pipeline.

Multiplier in VP0 provides 16-bit x 16-bit -> 32-bit pipelined multiplies.
Scale results by any shift amount.
Provides 4 rounding modes including round-to-nearest-even.
Clip results to 8-bit, 16-bit, or 32-bit range.

Vector arithmetic operations have 3 cycle latency.

T0 Vector Conditional Operations

Vectorize loops containing conditional statements.

Executed in either arithmetic pipeline.

Vector compare:

```
# vv2[i] =  (vv5[i] < vv6[i])
slt.vv vv2, vv5, vv6
```

Vector conditional move:

```
# if (vv2[i] > 0) then vv1[i] = vv3[i]
cmvgtz.vv vv1, vv2, vv3
```

Vector condition flag register:

```
# vcond[i] =  (vv1[i] < vv2[i])
flt.vv vv1, vv2     # Set flag bits.
cfc2 r1, vcond      # Read into scalar reg.
```

6.3-15

T0 Vector Unit Hazards

All vector unit hazards fully interlocked in hardware.

Vector instruction startup fully pipelined to eliminate stripmining overhead.

Each functional unit has independent ports into vector register file so no chain slot time and no vector register access conflicts.

All RAW, WAR, and WAW hazards on vector registers fully chained to reduce latency and decrease vector register pressure.

Philosophy: Trade small amount of extra control logic for increased utilization of multiple, expensive datapaths.

6.3-16

T0 External Interfaces

External Memory Interface

Supports up to 4 GB of SRAM with 720 MB/s bandwidth.

SRAM access wave-pipelined over 1.5 cycles.

Industry standard 17ns asynchronous SRAM for 45 MHz.

Serial Interface Port

Based on JTAG, but with 8 bit datapaths.

Provides chip testing and processor single-step.

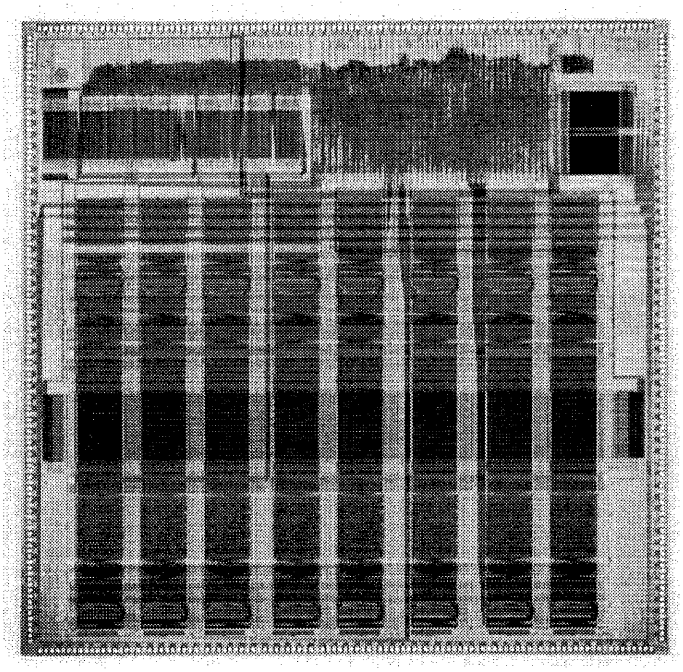
Supports 36 MB/s host-T0 memory bandwidth.

Hardware Performance Monitoring

Eight pins give cycle by cycle CPU and VU status.

Fast External Interrupts

Two prioritized fast interrupt pins with dedicated interrupt vectors.



T0 Die Photo

T0 Die Statistics

Technology:
1.0 μm MOSIS scalable CMOS rules, 2 metal, 1 poly.
Contacted M1 pitch 3.25 μm
Contacted M2 pitch 3.75 μm
Fabbed in HP's CMOS 26G process.

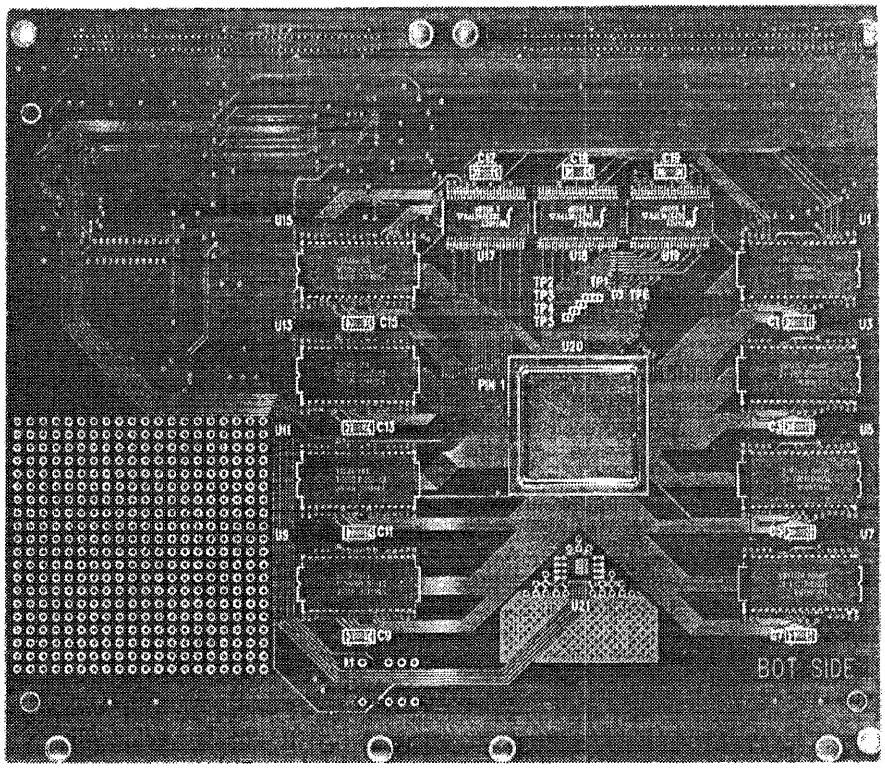
Die Size: 16.75 mm x 16.75 mm

Transistor Count: 730,701

Clock Frequency: 45 MHz

Power Supply: 5 V

Power Dissipation: <10 W



SPERT-II Board Photo

Summary

With a fully programmable, scalable, vector-register instruction set architecture, T0 sustains up to 720 million arithmetic ops/s while accessing up to 360 million operands/s from main memory, and with up to 36 MB/s of concurrent I/O,

...with roughly 3/4 million transistors (1 Gλ^2),

...at 45 MHz in a 1 μm 2-Al CMOS process,

...in less than 10 person years from no ISA to running applications.