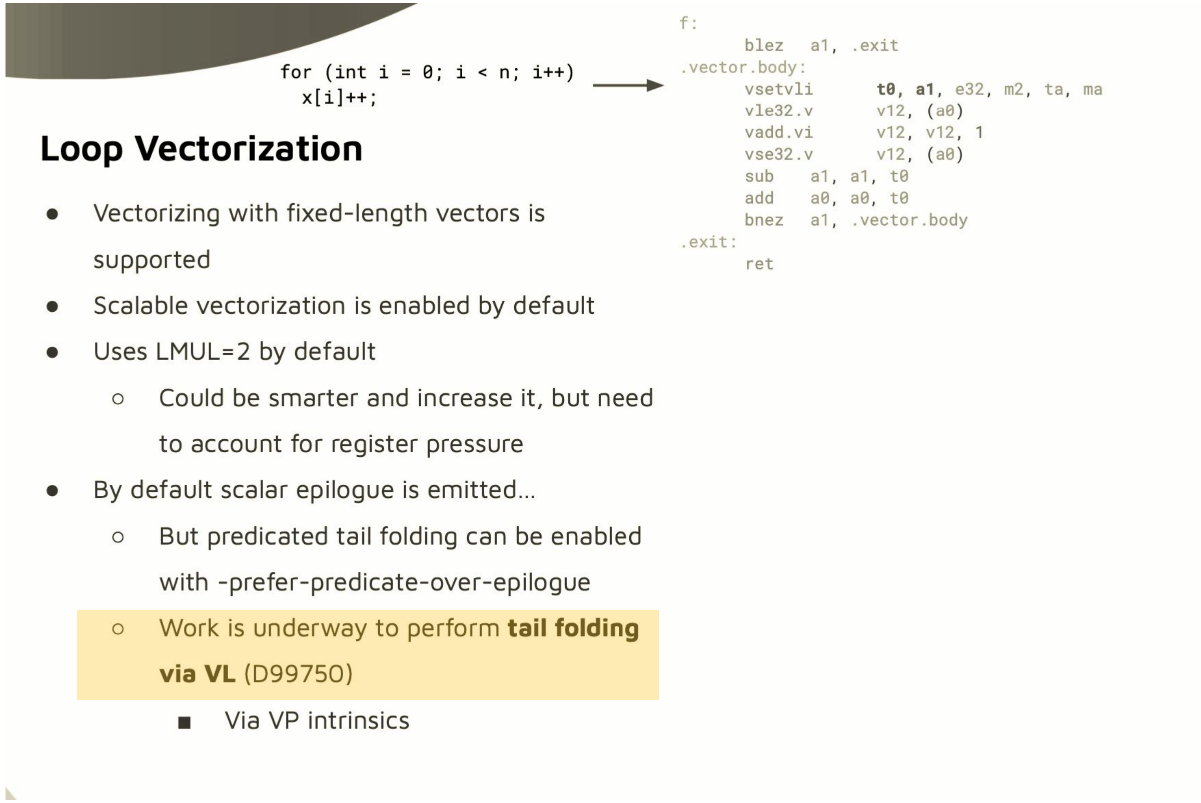
Accidental Dataflow Analysis: Extending the RISC-V VL Optimizer

Luke Lau, Igalia
Michael Maitland, SiFive

EuroLLVM 2025

Outline

- EVL tail folding
- Measuring codegen impact
- RISCVVLOptimizer
- Dataflow analysis



```
for (int i = 0; i < n; i++)
  x[i]++;
```

```
f:
        blez    a1, .exit
.vector.body:
        vsetvli     t0, a1, e32, m2, ta, ma
        vle32.v     v12, (a0)
        vadd.vi     v12, v12, 1
        vse32.v     v12, (a0)
        sub     a1, a1, t0
        add     a0, a0, t0
        bnez    a1, .vector.body
.exit:
        ret
```

Loop Vectorization

- Vectorizing with fixed-length vectors is supported
- Scalable vectorization is enabled by default
- Uses LMUL=2 by default
  - Could be smarter and increase it, but need to account for register pressure
- By default scalar epilogue is emitted…
  - But predicated tail folding can be enabled with -prefer-predicate-over-epilogue
  - Work is underway to perform **tail folding via VL** (D99750)
    - Via VP intrinsics

Body

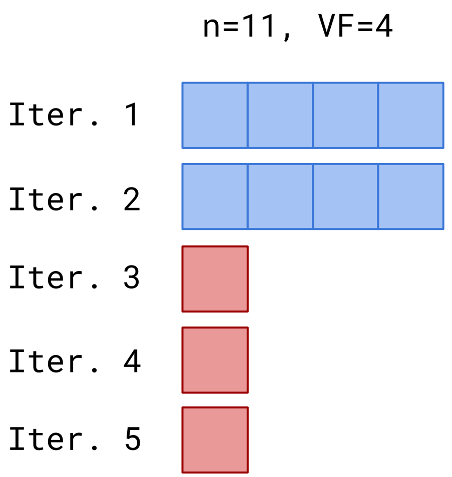
Tail

```
.vector.preheader:
        andi    a2, a1, -8
        vsetivli     zero, 8, e32, m2, ta, ma
        mv      a3, a2
        mv      a4, a0
.vector.body:
        vle32.v      v8, (a4)
        vadd.vi      v8, v8, 1
        vse32.v      v8, (a4)
        addi    a3, a3, -8
        addi    a4, a4, 32
        bnez    a3, .vector.body
        beq     a2, a1, .exit
.scalar.preheader:
        slli    a3, a2, 2
        add     a0, a0, a3
        sub     a1, a1, a2
.scalar.body:
        lw      a2, 0(a0)
        addi    a2, a2, 1
        sw      a2, 0(a0)
        addi    a1, a1, -1
        addi    a0, a0, 4
        bnez    a1, .scalar.body
.exit:
        ret
```

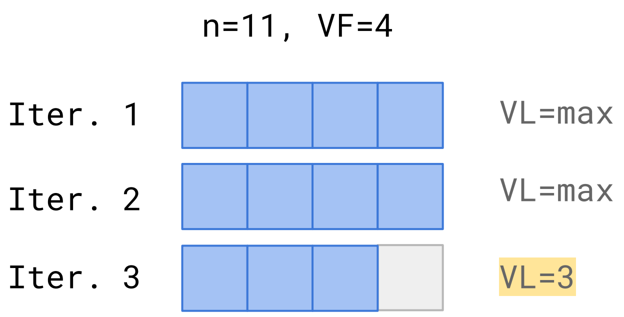
EVL tail folding $\longrightarrow$

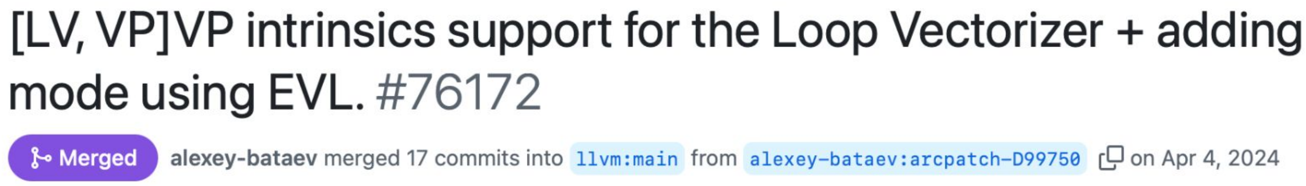
```
.vector.body:
        vsetvli      t0, a1, e32, m2, ta, ma
        vle32.v      v12, (a0)
        vadd.vi      v12, v12, 1
        vse32.v      v12, (a0)
        sub     a1, a1, t0
        add     a0, a0, t0
        bnez    a1, .vector.body
.exit:
        ret
```

```
int i = 0;

// vector body
for (; i < (n/VF)*VF; i+=VF)
  x[i:i+VF]++;

// scalar tail
for (; i < n; i++)
  x[i]++;
```



```
// vl tail folded vector loop
for (int i = 0; i < n;) {
  int vl = getVL(n - i);
  x[i:i+vl]++
  n += vl;
}
```





[LV, VP]VP intrinsics support for the Loop Vectorizer + adding new tail-folding mode using EVL. #76172

Merged alexey-bataev merged 17 commits into `llvm:main` from `alexey-bataev:arcpatch-D99750` on Apr 4, 2024

```
clang -march=rva23u64 -O3 foo.c
-mllvm -force-tail-folding-style=data-with-evl
-mllvm -prefer-predicate-over-epilogue=predicate-else-scalar-epilogue
```

What do we need to do to turn it on by default?

```
$ cd llvm-test-suite

# configure
$ cmake -B build.rva23u64-evl -C cmake/caches/O3.cmake
        -DCMAKE_C_COMPILER=$HOME/llvm-project/build/bin/clang
        -DCMAKE_C_FLAGS="-march=rva23u64 -mllvm -force-tail-folding-style=..."
        -DCMAKE_SPEC2017_ROOT=$HOME/cpu2017

# build
$ ninja -C build.rva23u64-evl

# run
$ llvm-lit build.rva23u64-evl
-- Testing: 2074 tests, 12 workers –
...
```

```
$ cd llvm-test-suite

# configure
$ cmake -B build.rva23u64-evl -C cmake/caches/O3.cmake
        -DCMAKE_C_COMPILER=$HOME/llvm-project/build/bin/clang
        -DCMAKE_C_FLAGS="-march=rva23u64 -mllvm -force-tail-folding-style=..."
        -DCMAKE_SPEC2017_ROOT=$HOME/cpu2017

# build
$ ninja -C build.rva23u64-evl

# run
$ llvm-lit build.rva23u64-evl
-- Testing: 2074 tests, 12 workers –
...
FAIL: test-suite :: External/SPEC/CINT2017rate/502.gcc_r/502.gcc_r.test (1 of 2074)
******************** TEST 'test-suite :: External/SPEC/CINT2017rate/502.gcc_r/502.gcc_r.test' FAILED
********************
benchmark internal error: in ?, at reload1.c:2020
The 502.gcc_r benchmark binary 'cpugcc_r' has encountered an internal error.
It is possible that there is an error in the benchmark 502.gcc_r
source code, but it is more likely that your compiler
has mis-optimized or otherwise generated bad code for
the benchmark.  You might try reducing the optimization
```

Debugging miscompiles

- Write a script that can reproduce the miscompile

```
#!/bin/bash

ninja -C llvm-project/build clang
ninja -C llvm-test-suite/build.rva23u64-evl 502.gcc_r
./llvm-test-suite/build.rva23u64-evl/…/502.gcc_r input.c -O3 …
```

Debugging miscompiles

- Write a script that can reproduce the miscompile
- Problem 1: This benchmark **takes hours**

```
#!/bin/bash

ninja -C llvm-project/build clang
ninja -C llvm-test-suite/build.rva23u64-evl 502.gcc_r
qemu-riscv64 -cpu rv64,v=true,vlen=128,vext_spec=v1.0 ./llvm-test-suite/build.rva23u64-evl/…/502.gcc_r input.c -O3
```

Debugging miscompiles

- Write a script that can reproduce the miscompile
- Problem 1: This benchmark **takes hours**
- So just check it doesn’t crash within the first few seconds

```bash
#!/bin/bash

ninja -C llvm-project/build clang
ninja -C llvm-test-suite/build.rva23u64-evl 502.gcc_r
timeout 15 qemu-riscv64 -cpu rv64,v=true,vlen=128,vext_spec=v1.0 ./llvm-test-suite/build.rva23u64-evl/…/502.gcc_r i
if [ $? -eq 124 ]
then
        exit 0
else
        exit 1
fi
```

Debugging miscompiles

- The miscompile has always been there – **Can't use git bisect**
- Have to manually investigate the diff between the binaries
- But the diff is way too large

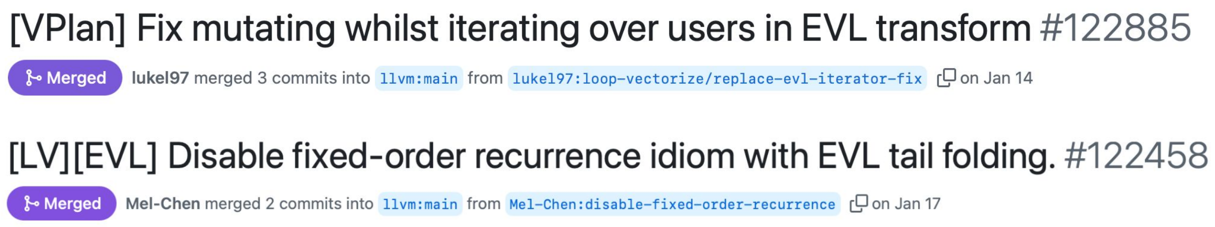
```
$ llvm-objdump -d build.rva23u64/…/502.gcc_r -o before.s
$ llvm-objdump -d build.rva23u64-evl/…/502.gcc_r -o after.s
$ git diff --no-index --stat before.s after.s
 {before.s => after.s} | 1809529 +++++++++++++++++++++++++++++++++----------------------------------
 1 file changed, 904172 insertions(+), 905357 deletions(-)
```

- Use **./llvm/utils/rsp_bisect.py**
- Take a good build (no tail folding) and a bad build (with tail folding)
- Copy the objects linked into the final binary into an rsp file
- Write a script that links & tests for the miscompile

```
$ ninja -C build.rva23u64 && ninja -C build.rva23u64-evl
$ cat 502.gcc.rsp
foo.c.o bar.c.o baz.c.o …
$ cat bisect.sh
#!/bin/bash

./build/bin/clang -march=rva23u64 -O3 -o 502.gcc_r @502.gcc_r.rsp
qemu-riscv64 -cpu rv64,v=on,vext_spec=v1.0 ./502.gcc_r …
…
$ cd build.rva23u64
```

- **rsp_bisect.py** swaps out good and bad object files till it finds a single offender

```
$ cd build.rva23u64
$ ./llvm/utils/rsp_bisect.py --test=../bisect.sh \
                             --rsp=../502.gcc_r.rsp \
                             --other-rel-path=../build.rva23u64-evl
387 files in rsp
Initial testing
First build directory returned 0
Trying 193 (0-387)
Trying 290 (193-387)
Trying 241 (193-290)
Trying 217 (193-241)
Trying 229 (217-241)
Trying 235 (229-241)
Trying 238 (235-241)
Trying 239 (238-241)
Trying 240 (239-241)
First file change: External/SPEC/…/reload1.c.o (241)
Bisection point rsp files written to 502.gcc_r.rsp.0 and 502.gcc_r.rsp.1
```



[VPlan] Fix mutating whilst iterating over users in EVL transform #122885

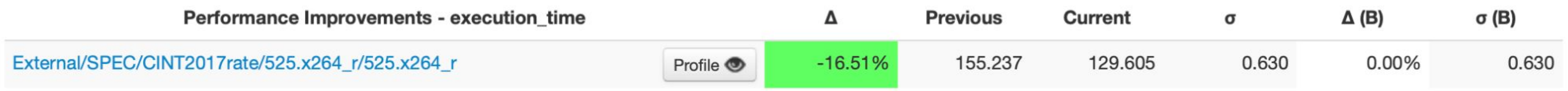
Merged lukel97 merged 3 commits into `llvm:main` from `lukel97:loop-vectorize/replace-evl-iterator-fix` on Jan 14

[LV][EVL] Disable fixed-order recurrence idiom with EVL tail folding. #122458

Merged Mel-Chen merged 2 commits into `llvm:main` from `Mel-Chen:disable-fixed-order-recurrence` on Jan 17

```
$ llvm-lit build.rva23u64-evl
-- Testing: 2074 tests, 12 workers –
…
Total Discovered Tests: 2072
  Passed            :   2072 (100%)
```

Can we enable it by default now?



Performance Improvements - execution_time		Δ	Previous	Current	σ	Δ (B)	σ (B)
External/SPEC/CINT2017rate/525.x264_r/525.x264_r	Profile	-16.51%	155.237	129.605	0.630	0.00%	0.630

- There still might be regressions hidden by other performance improvements
- Should also manually inspect the codegen difference

```
$ cmake -B build.rva23u64 -DCMAKE_C_FLAGS="-save-temps=obj …" …
$ cmake -B build.rva23u64-evl -DCMAKE_C_FLAGS="-save-temps=obj …" …

$ ninja -C build.rva23u64
$ ninja -C build.rva23u64-evl

$ ./utils/tdiff.py -a build.rva23u64 -b build.rva23u64-evl -s all | less
```

```
; before evl tail folding
vwsubu.vv       v16, v14, v15
vsetvli zero, zero, e16, mf2, ta, ma
vwmul.vv        v14, v16, v16
```

```
; after evl tail folding
vzext.vf2       v16, v14
vzext.vf2       v14, v15
vwsubu.vv       v15, v16, v14
vsetvli zero, zero, e32, m1, ta, ma
vmul.vv v14, v15, v15
```

```
$ clang … -emit-llvm | llvm-extract --func=foo | llvm-dis
```

```
%x = mul <vscale x 2 x i32> %y, %z
```

```
%x = call <vscale x 2 x i32> @llvm.vp.mul(
  <vscale x 2 x i32> %x,
  <vscale x 2 x i32> %y,
  <vscale x 2 x i1> %mask,
  i32 %evl
)
```

Why are we missing out on combines?

```
$ clang … -emit-llvm -o before.bc
$ clang … -emit-llvm -o after.bc

$ llvm-extract --func=foo before.bc -o foo.before.bc
$ llvm-extract --func=foo after.bc -o foo.after.bc

$ llvm-dis foo.before.bc
$ llvm-dis foo.after.bc

$ diff -u foo.before.ll foo.after.ll
$ clang … -emit-llvm | llvm-extract --func=foo | llvm-dis
```

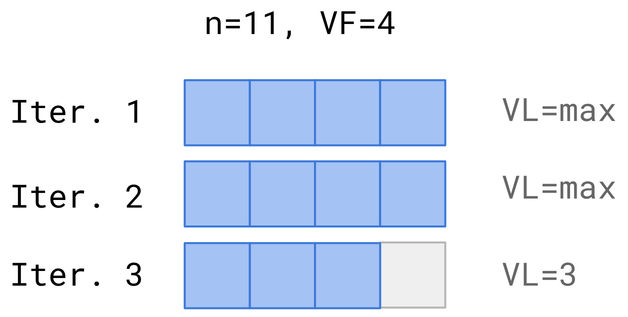
```
; before tail folding
%6 = sdiv <vscale x 4 x i32> %wide.load, splat (i32 3)
```

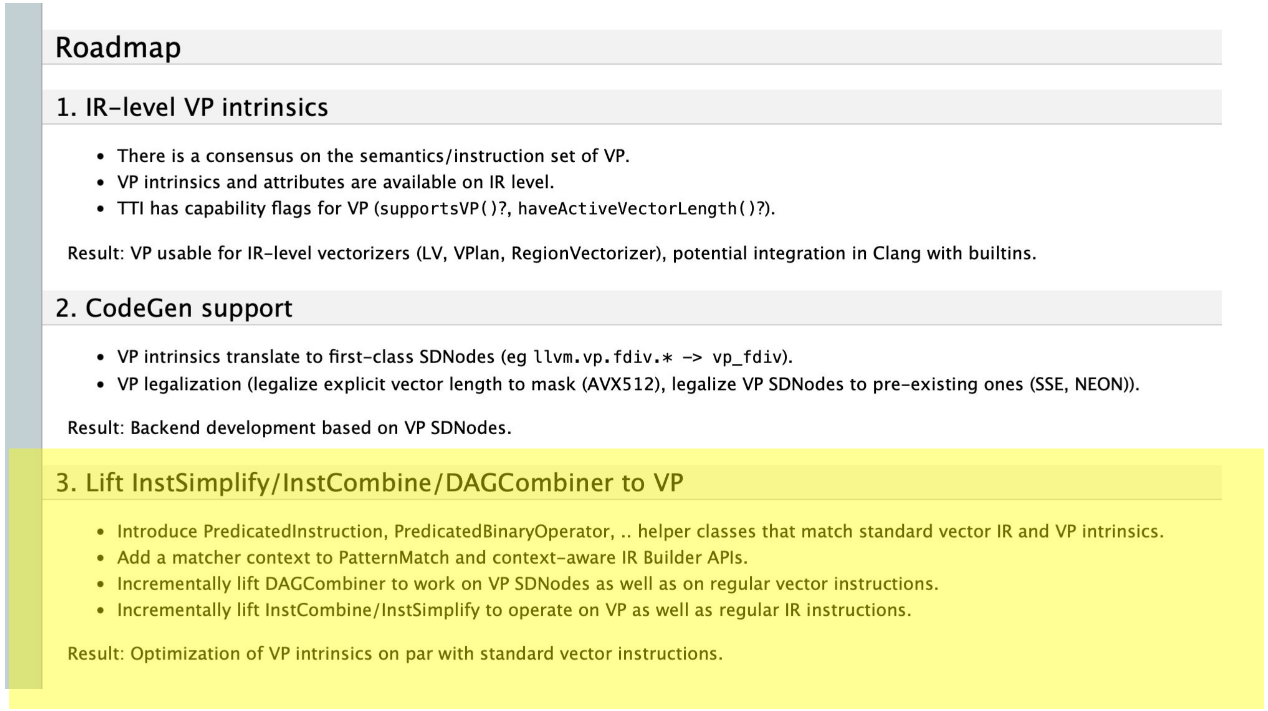
```
; after tail folding
%vp.op.load = call <vscale x 4 x i32> @llvm.vp.load(
  ptr align 4 %6,
  <vscale x 4 x i1> splat (i1 true),
  i32 %evl
)
%vp.op = call <vscale x 4 x i32> @llvm.vp.sdiv(
  <vscale x 4 x i32> %vp.op.load,
  <vscale x 4 x i32> splat (i32 3),
  <vscale x 4 x i1> splat (i1 true),
  i32 %evl
)
call void @llvm.vp.store(
  <vscale x 4 x i32> %vp.op,
  ptr align 4 %6,
  <vscale x 4 x i1> splat (i1 true),
  i32 %evl
)
```

VP intrinsics

```
declare <4 x i32> @llvm.vp.add(<4 x i32> %x, <4 x i32> %y, <4 x i1> %mask, i32 %evl)
```

- Target independent intrinsics for controlling the mask and vector length (EVL)
- **Opaque to a lot of optimisations and patterns**
- Needed for trapping instructions for correctness
- Needed for non-trapping instructions for performance!





Roadmap

1. IR-level VP intrinsics

- There is a consensus on the semantics/instruction set of VP.
- VP intrinsics and attributes are available on IR level.
- TTI has capability flags for VP (`supportsVP()`?, `haveActiveVectorLength()`?).

Result: VP usable for IR-level vectorizers (LV, VPlan, RegionVectorizer), potential integration in Clang with builtins.

2. CodeGen support

- VP intrinsics translate to first-class SDNodes (eg `llvm.vp.fdiv.* -> vp_fdiv`).
- VP legalization (legalize explicit vector length to mask (AVX512), legalize VP SDNodes to pre-existing ones (SSE, NEON)).

Result: Backend development based on VP SDNodes.

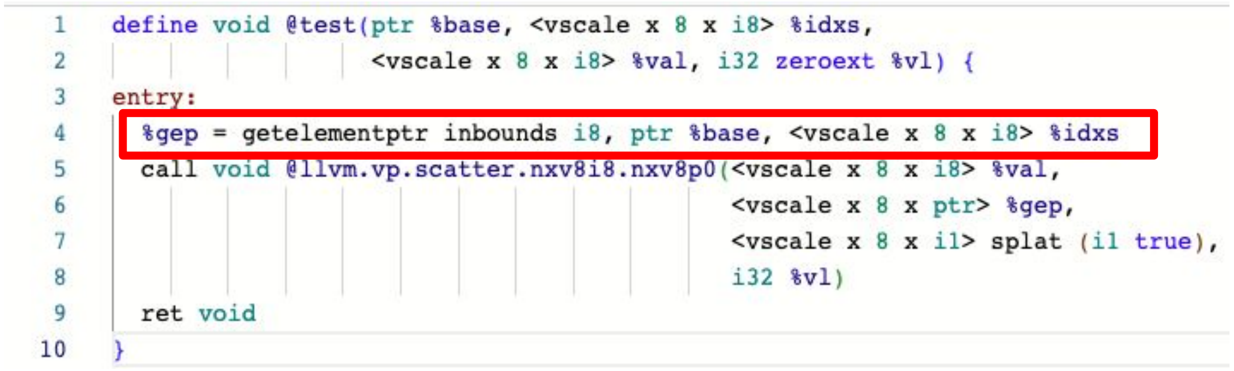
3. Lift InstSimplify/InstCombine/DAGCombiner to VP

- Introduce PredicatedInstruction, PredicatedBinaryOperator, .. helper classes that match standard vector IR and VP intrinsics.
- Add a matcher context to PatternMatch and context-aware IR Builder APIs.
- Incrementally lift DAGCombiner to work on VP SDNodes as well as on regular vector instructions.
- Incrementally lift InstCombine/InstSimplify to operate on VP as well as regular IR instructions.

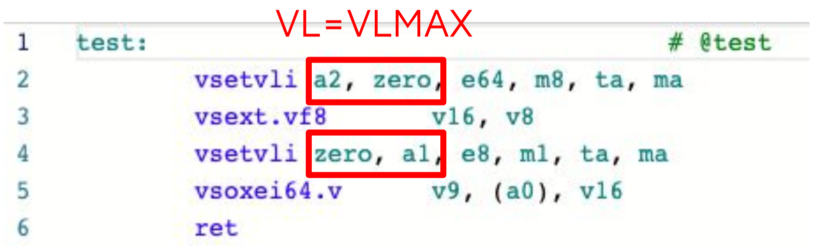
Result: Optimization of VP intrinsics on par with standard vector instructions.

Do we need to lift every combine to work on VP intrinsics?

Meanwhile at SiFive



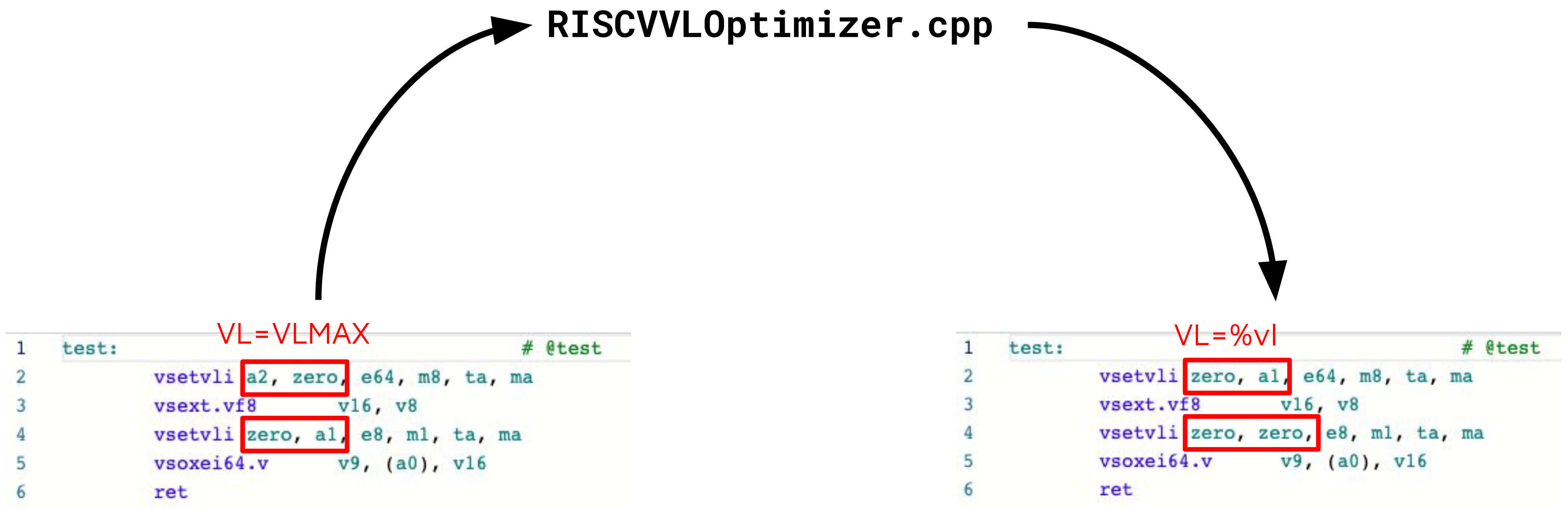
```
define void @test(ptr %base, <vscale x 8 x i8> %idxs,
                  <vscale x 8 x i8> %val, i32 zeroext %vl) {
entry:
  %gep = getelementptr inbounds i8, ptr %base, <vscale x 8 x i8> %idxs
  call void @llvm.vp.scatter.nxv8i8.nxv8p0(<vscale x 8 x i8> %val,
                                           <vscale x 8 x ptr> %gep,
                                           <vscale x 8 x i1> splat (i1 true),
                                           i32 %vl)
  ret void
}
```



```
test:                                   # @test
        vsetvli a2, zero, e64, m8, ta, ma
        vsext.vf8       v16, v8
        vsetvli zero, a1, e8, m1, ta, ma
        vsoxei64.v      v9, (a0), v16
        ret
```

[RISCV] Introduce VLOptimizer pass #108640

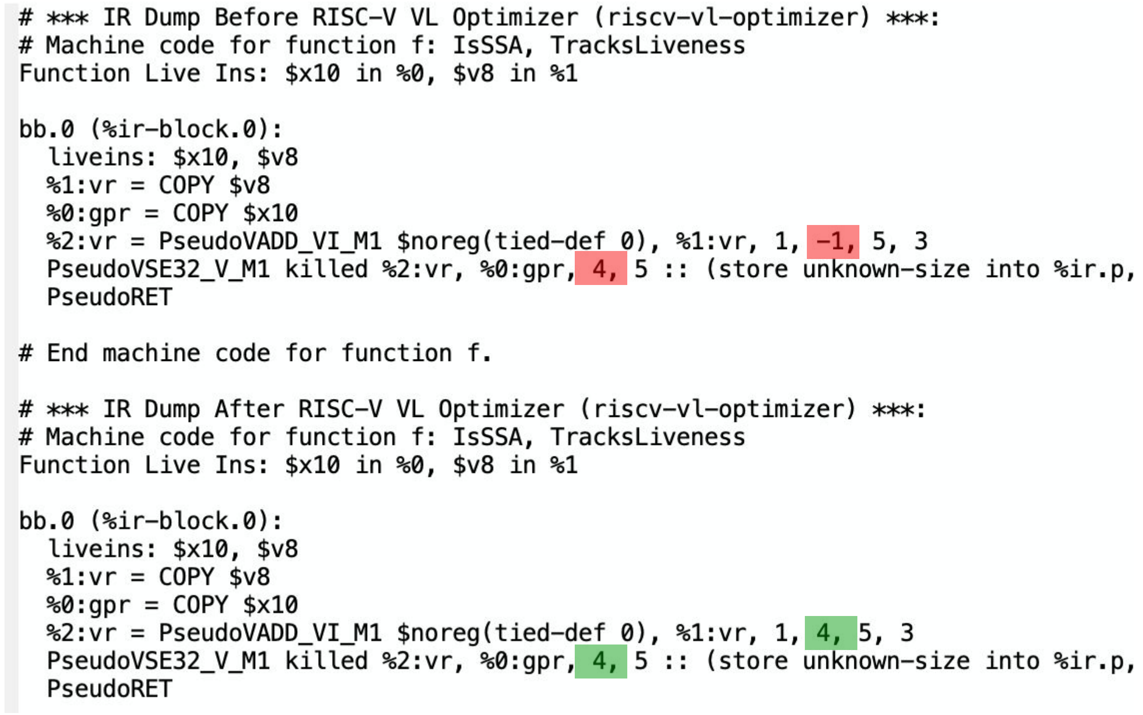
Merged **michaelmaitland** merged 17 commits into `llvm:main` from `michaelmaitland:vl-optimize` on Oct 11, 2024



```
test:                                   # @test
        vsetvli a2, zero, e64, m8, ta, ma
        vsext.vf8       v16, v8
        vsetvli zero, a1, e8, m1, ta, ma
        vsoxei64.v      v9, (a0), v16
        ret
```

```
test:                                   # @test
        vsetvli zero, a1, e64, m8, ta, ma
        vsext.vf8       v16, v8
        vsetvli zero, zero, e8, m1, ta, ma
        vsoxei64.v      v9, (a0), v16
        ret
```

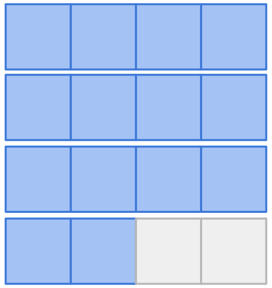
Reduces the VL of **RISC-V instructions** at the **Machine IR** level



```
# *** IR Dump Before RISC-V VL Optimizer (riscv-vl-optimizer) ***:
# Machine code for function f: IsSSA, TracksLiveness
Function Live Ins: $x10 in %0, $v8 in %1

bb.0 (%ir-block.0):
  liveins: $x10, $v8
  %1:vr = COPY $v8
  %0:gpr = COPY $x10
  %2:vr = PseudoVADD_VI_M1 $noreg(tied-def 0), %1:vr, 1, -1, 5, 3
  PseudoVSE32_V_M1 killed %2:vr, %0:gpr, 4, 5 :: (store unknown-size into %ir.p,
  PseudoRET

# End machine code for function f.

# *** IR Dump After RISC-V VL Optimizer (riscv-vl-optimizer) ***:
# Machine code for function f: IsSSA, TracksLiveness
Function Live Ins: $x10 in %0, $v8 in %1

bb.0 (%ir-block.0):
  liveins: $x10, $v8
  %1:vr = COPY $v8
  %0:gpr = COPY $x10
  %2:vr = PseudoVADD_VI_M1 $noreg(tied-def 0), %1:vr, 1, 4, 5, 3
  PseudoVSE32_V_M1 killed %2:vr, %0:gpr, 4, 5 :: (store unknown-size into %ir.p,
  PseudoRET
```

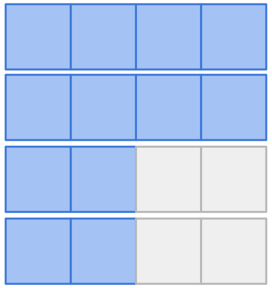
```
for instr in reverse(block):
  demanded_vl = max(users(instr).vl)
  if (demanded_vl < instr.vl):
    instr.vl = demanded_vl
```

```
%x = load %p1, vl=4
%y = load %p2, vl=4
%z = add %x, %y, vl=4
store %z, vl=2
```



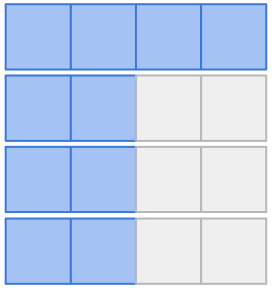
```
for instr in reverse(block):
  demanded_vl = max(users(instr).vl)
  if (demanded_vl < instr.vl):
    instr.vl = demanded_vl
```

```
%x = load %p1, vl=4
%y = load %p2, vl=4
%z = add %x, %y, vl=2
store %z, %p3, vl=2
```



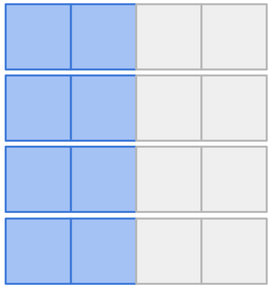
```
for instr in reverse(block):
  demanded_vl = max(users(instr).vl)
  if (demanded_vl < instr.vl):
    instr.vl = demanded_vl
```

```
%x = load %p1, vl=4
%y = load %p2, vl=2
%z = add %x, %y, vl=2
store %z, %p3, vl=2
```



```
for instr in reverse(block):
  demanded_vl = max(users(instr).vl)
  if (demanded_vl < instr.vl):
    instr.vl = demanded_vl
```

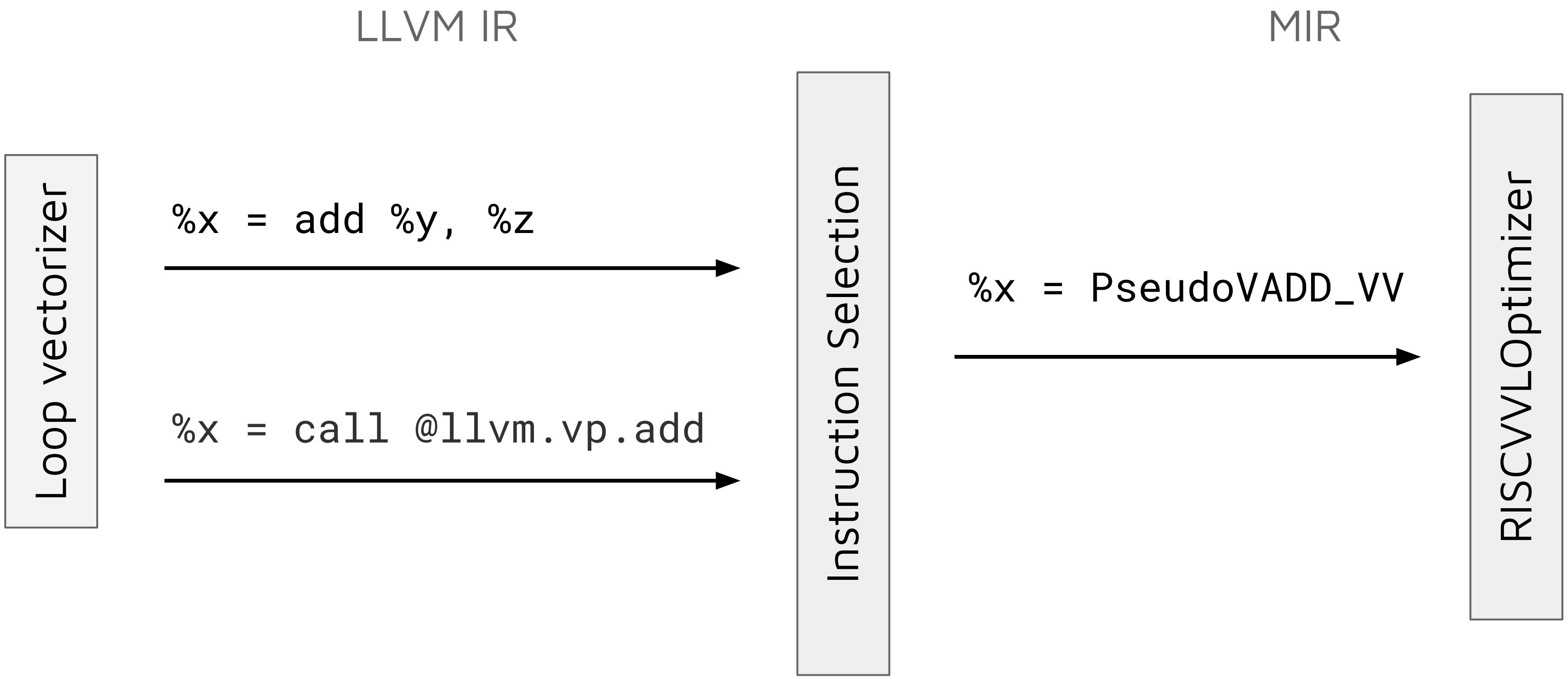
```
%x = load %p1, vl=2
%y = load %p2, vl=2
%z = add %x, %y, vl=2
store %z, %p3, vl=2
```



(For the RISC-V audience: we also need to make the VTYPEs are compatible!)

```
for instr in reverse(block):
  demanded_vl = max(users(instr).vl)
  if not vtype_compat(instr, users(instr)):
    demanded_vl = vlmax
  if (demanded_vl < instr.vl):
    instr.vl = demanded_vl
```

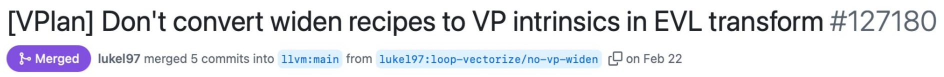
```
%x = load %p1, vl=2, VTYPE=...
%y = load %p2, vl=2, VTYPE=...
%z = add %x, %y, vl=2, VTYPE=...
store %z, %p3, vl=2, VTYPE=...
```


LLVM IR
MIR
Loop vectorizer
%x = add %y, %z
%x = call @llvm.vp.add
Instruction Selection
%x = PseudoVADD_VV
RISCVVLOptimizer

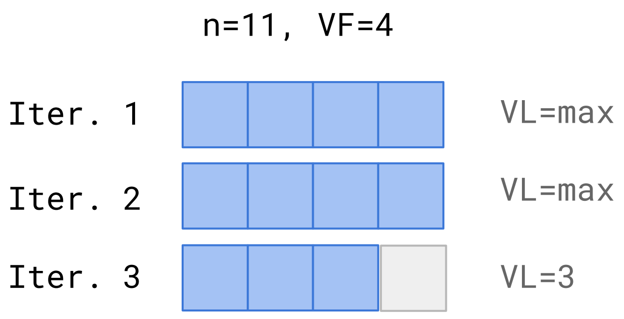
VP intrinsics

- Target independent intrinsics for controlling the mask and vector length (VL)
- Opaque to a lot of optimisations and patterns
- Needed for trapping instructions for correctness
- ~~Needed for non-trapping instructions for performance!~~
- Not needed for non-trapping instructions:

  RISCVVLOptimizer can take care of it!
  - Best of both worlds: Generic combines + optimized VL


[VPlan] Don't convert widen recipes to VP intrinsics in EVL transform #127180

Merged lukel97 merged 5 commits into `llvm:main` from `lukel97:loop-vectorize/no-vp-widen` on Feb 22



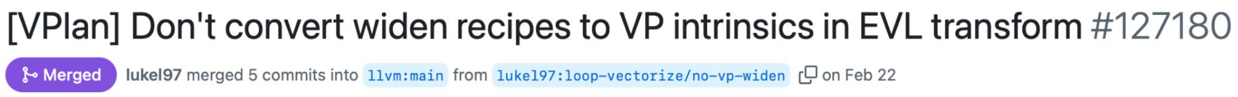
```
  %vp.op.load = call <vscale x 4 x i32> @llvm.vp.load(ptr align 4 %6, <vscale x 4 x i1> splat (i1 true), i32 %evl)
  %vp.op = call <vscale x 4 x i32> @llvm.vp.sdiv(<vscale x 4 x i32> %vp.op.load, <vscale x 4 x i32> splat (i32 3), <vscale x 4 x i1> splat (i1 true), i32 %evl)
  tail call void @llvm.vp.store(<vscale x 4 x i32> %vp.op, ptr align 4 %6, <vscale x 4 x i1> splat (i1 true), i32 %evl)
```

```
%vp.op.load = call <vscale x 4 x i32> @llvm.vp.load(ptr align 4 %6, <vscale x 4 x i1> splat (i1 true), i32 %evl)
%7 = sdiv <vscale x 4 x i32> %wide.load, splat (i32 3)
tail call void @llvm.vp.store(<vscale x 4 x i32> %7, ptr align 4 %6, <vscale x 4 x i1> splat (i1 true), i32 %evl)
```

```
body:
     sub  a5, a1, a2
     sh2add    a3, a2, a0
     vsetvli   a5, a5, e32, m2, ta, ma
     vle32.v   v8, (a3)
     vdiv.vx   v8, v8, a6
     sub  a4, a4, a7
     vse32.v   v8, (a3)
     add  a2, a2, a5
     bnez a4, .LBB0_2
```

```
body:
     sub  a5, a1, a2
     sh2add    a3, a2, a0
     vsetvli   a5, a5, e32, m2, ta, ma
     vle32.v   v8, (a3)
     vmulh.vx  v8, v8, a6
     vsrl.vi   v10, v8, 31
     vadd.vv   v8, v8, v10
     sub  a4, a4, a7
     vse32.v   v8, (a3)
     add  a2, a2, a5
     bnez a4, .LBB0_2
```

Problems can be easier to solve in different places!


[VPlan] Don't convert widen recipes to VP intrinsics in EVL transform #127180

Merged lukel97 merged 5 commits into `llvm:main` from `lukel97:loop-vectorize/no-vp-widen` on Feb 22

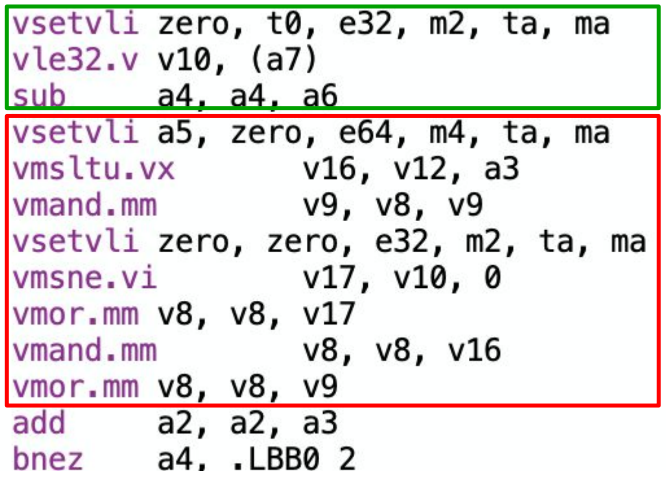
```
; before tail folding
body:
    vl2re32.v v8, (a3)
    vmulh.vx  v8, v8, a5
    vsrl.vi   v10, v8, 31
    vadd.vv   v8, v8, v10
    vs2r.v    v8, (a3)
    add  a3, a3, a7
    bne  a3, a4, .LBB0_4
```

```
; after tail folding
body:
    sub  a5, a1, a2
    sh2add    a3, a2, a0
    vsetvli   a5, a5, e32, m2, ta, ma
    vle32.v   v8, (a3)
    vmulh.vx  v8, v8, a6
    vsrl.vi   v10, v8, 31
    vadd.vv   v8, v8, v10
    sub  a4, a4, a7
    vse32.v   v8, (a3)
    add  a2, a2, a5
    bnez a4, .LBB0_2
```

```
$ ./utils/tdiff.py -a build.evl -b build.noevl -s all | less
```

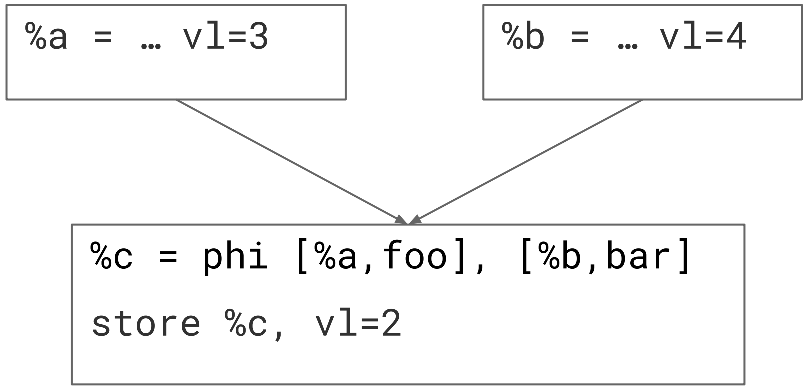
vl=t0

vl=max

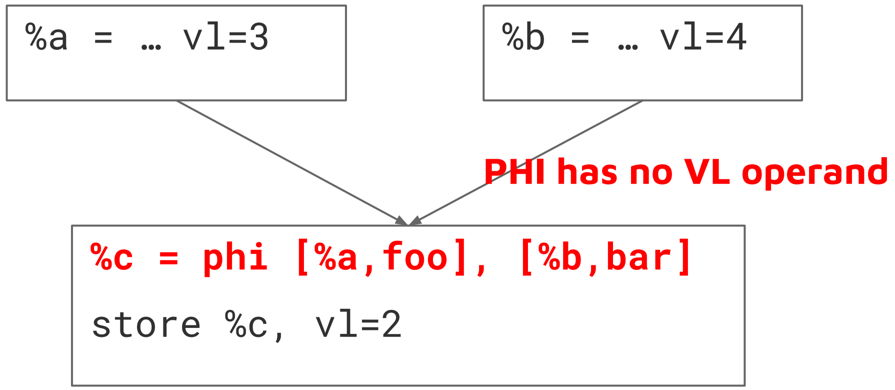


```
vsetvli zero, t0, e32, m2, ta, ma
vle32.v v10, (a7)
sub     a4, a4, a6
vsetvli a5, zero, e64, m4, ta, ma
vmsltu.vx       v16, v12, a3
vmand.mm        v9, v8, v9
vsetvli zero, zero, e32, m2, ta, ma
vmsne.vi        v17, v10, 0
vmor.mm v8, v8, v17
vmand.mm        v8, v8, v16
vmor.mm v8, v8, v9
add     a2, a2, a3
bnez    a4, .LBB0 2
```

```
for block in RPOT(func):
  for instr in reverse(block):
    demanded = max(users(instr).vl)
    if (demanded < instr.vl):
      instr.vl = demanded
```



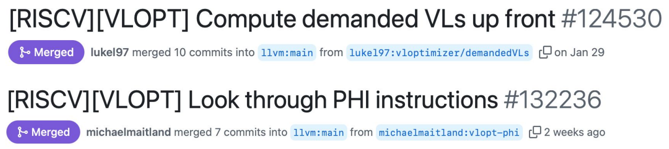
```
for block in RPOT(func):
  for instr in reverse(block):
    demanded = max(users(instr).vl)
    if (demanded < instr.vl):
      instr.vl = demanded
```



```
demanded = {}
for instr in func:
  demanded[instr] = instr.vl || vlmax
```

```
for block in RPOT(func):
  for instr in reverse(block):
    demanded[instr] = max(demanded[users(instr)])
```

```
for instr in block:
  if (demanded[instr] < x.vl):
    x.vl = demanded[instr]
```

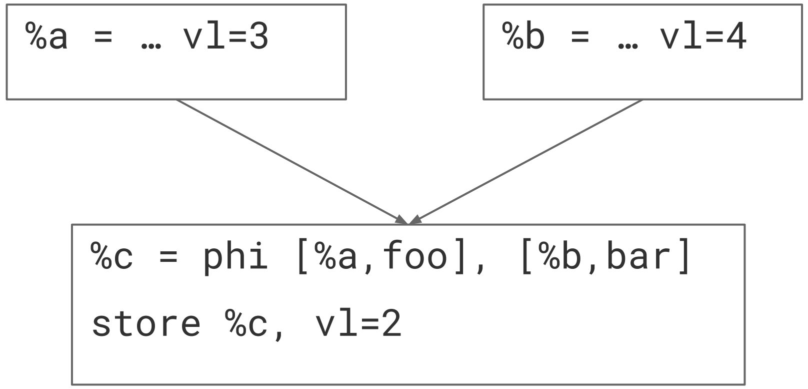


[RISCV][VLOPT] Compute demanded VLs up front #124530

Merged lukel97 merged 10 commits into `llvm:main` from `lukel97:vloptimizer/demandedVLs` on Jan 29

[RISCV][VLOPT] Look through PHI instructions #132236

Merged michaelmaitland merged 7 commits into `llvm:main` from `michaelmaitland:vlopt-phi` 2 weeks ago



```
%a = … vl=3
```

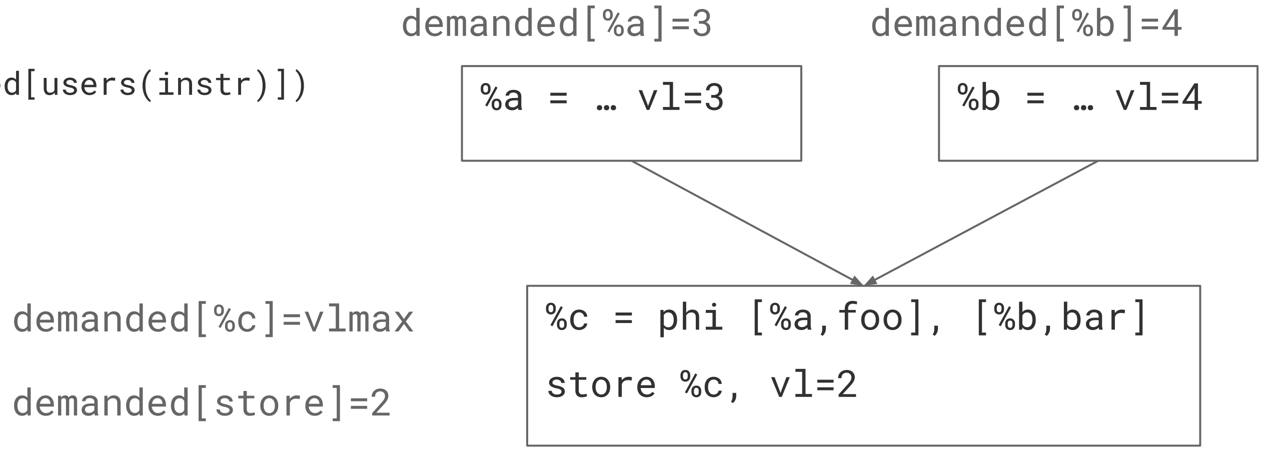
```
%b = … vl=4
```

```
%c = phi [%a,foo], [%b,bar]
store %c, vl=2
```

```
demanded = {}
for instr in func:
  demanded[instr] = instr.vl || vlmax
```

```
for block in RPOT(func):
  for instr in reverse(block):
    demanded[instr] = max(demanded[users(instr)])
```

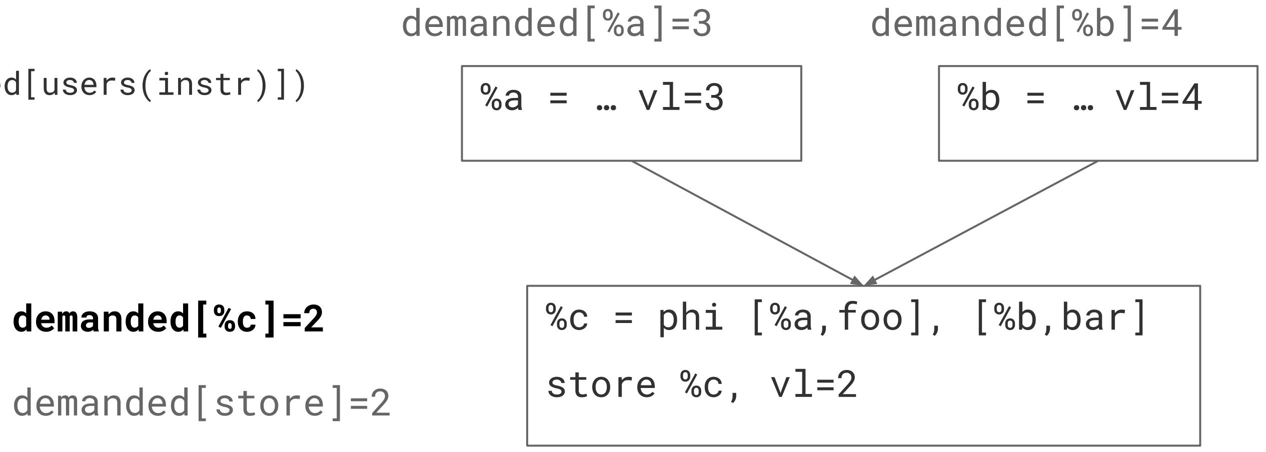
```
for instr in block:
  if (demanded[instr] < x.vl):
    x.vl = demanded[instr]
```



```
demanded = {}
for instr in func:
  demanded[instr] = instr.vl || vlmax
```

```
for block in RPOT(func):
  for instr in reverse(block):
    demanded[instr] = max(demanded[users(instr)])
```

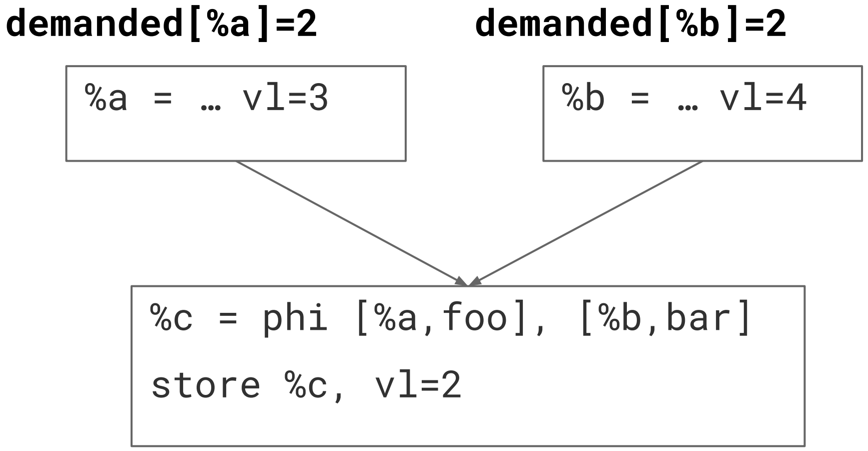
```
for instr in block:
  if (demanded[instr] < x.vl):
    x.vl = demanded[instr]
```



```
demanded = {}
for instr in func:
  demanded[instr] = instr.vl || vlmax
```

```
for block in RPOT(func):
  for instr in reverse(block):
    demanded[instr] = max(demanded[users(instr)])
```

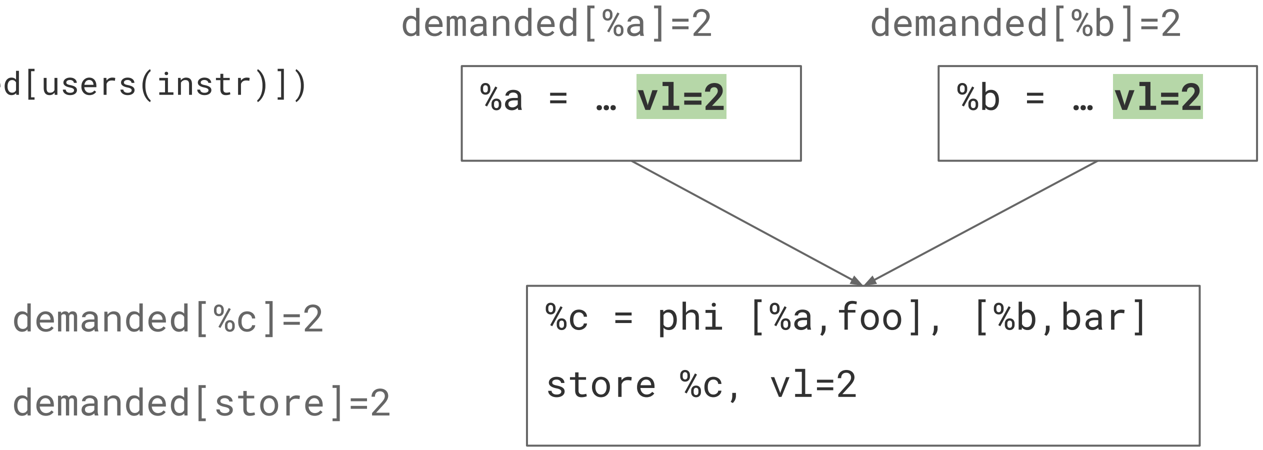
```
for instr in block:
  if (demanded[instr] < x.vl):
    x.vl = demanded[instr]
```



```
demanded = {}
for instr in func:
  demanded[instr] = instr.vl || vlmax
```

```
for block in RPOT(func):
  for instr in reverse(block):
    demanded[instr] = max(demanded[users(instr)])
```

```
for instr in block:
  if (demanded[instr] < x.vl):
    x.vl = demanded[instr]
```

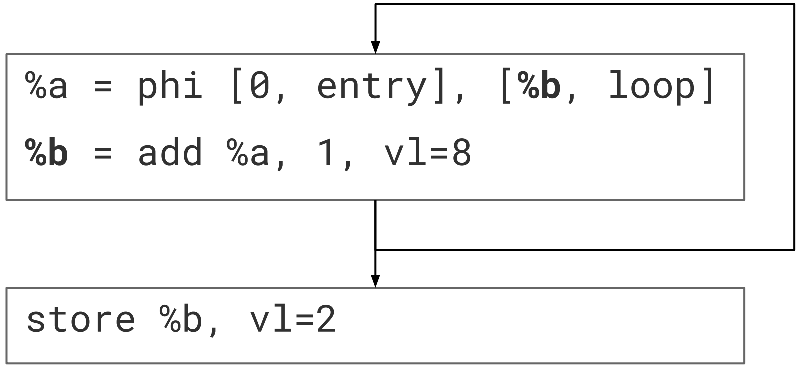


```
%a = … vl=2
```

```
%b = … vl=2
```

```
%c = phi [%a,foo], [%b,bar]
store %c, vl=2
```

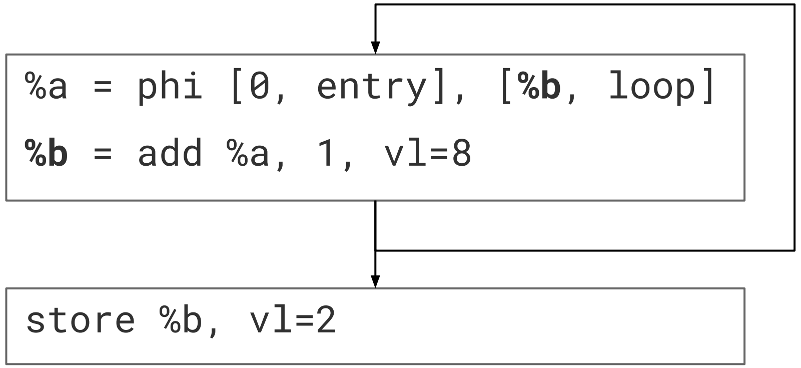
```
demanded = {}
for instr in func:
  demanded[instr] = instr.vl || vlmax



for block in RPOT(func):
  for instr in reverse(block):
    demanded[instr] = max(demanded[users(instr)])



for instr in block:
  if (demanded[instr] < x.vl):
    x.vl = demanded[instr]
```



```
%a = phi [0, entry], [%b, loop]
%b = add %a, 1, vl=8
```

```
store %b, vl=2
```

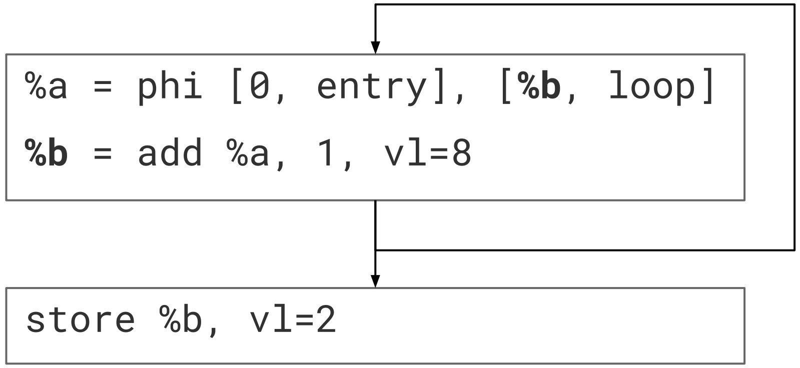
```
demanded = {}, worklist = {}
for instr in func:
  demanded[instr] = instr.vl || vlmax
for block in RPOT(func):
  for instr in reverse(block):
    worklist.insert(instr)

while instr = worklist.pop_front():
  demanded[instr] = max(demanded[users(instr)])
  worklist += instr.ops()




for instr in block:
  if (demanded[instr] < x.vl):
    x.vl = demanded[instr]
```



```
%a = phi [0, entry], [%b, loop]
%b = add %a, 1, vl=8
```

```
store %b, vl=2
```

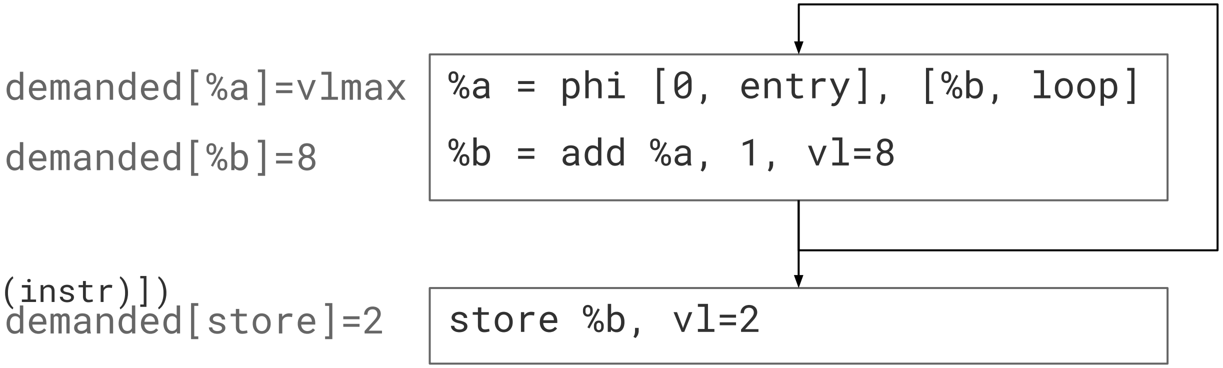
```
demanded = {}, worklist = {}
for instr in func:
  demanded[instr] = instr.vl || vlmax
for block in RPOT(func):
  for instr in reverse(block):
    worklist.insert(instr)

while instr = worklist.pop_front():
  prev = demanded[instr]
  demanded[instr] = max(demanded[users(instr)])
  if demanded[instr] != prev
    worklist += instr.ops()


for instr in block:
  if (demanded[instr] < x.vl):
    x.vl = demanded[instr]
```



```
%a = phi [0, entry], [%b, loop]
%b = add %a, 1, vl=8
```

```
store %b, vl=2
```

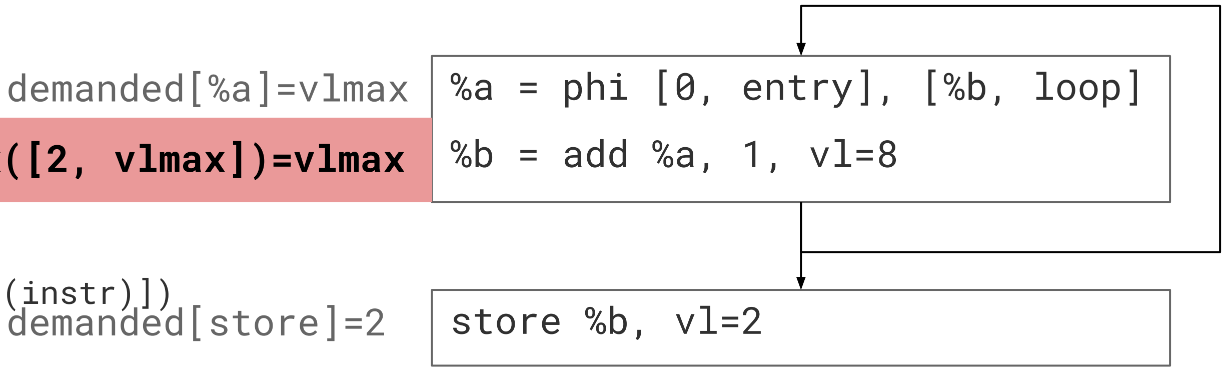
```
demanded = {}, worklist = {}
for instr in func:
  demanded[instr] = instr.vl || vlmax
for block in RPOT(func):
  for instr in reverse(block):
    worklist.insert(instr)

while instr = worklist.pop_front():
  prev = demanded[instr]
  demanded[instr] = max(demanded[users(instr)])
  if demanded[instr] != prev
    worklist += instr.ops()


for instr in block:
  if (demanded[instr] < x.vl):
    x.vl = demanded[instr]
```



```
%a = phi [0, entry], [%b, loop]
%b = add %a, 1, vl=8
```

```
store %b, vl=2
```

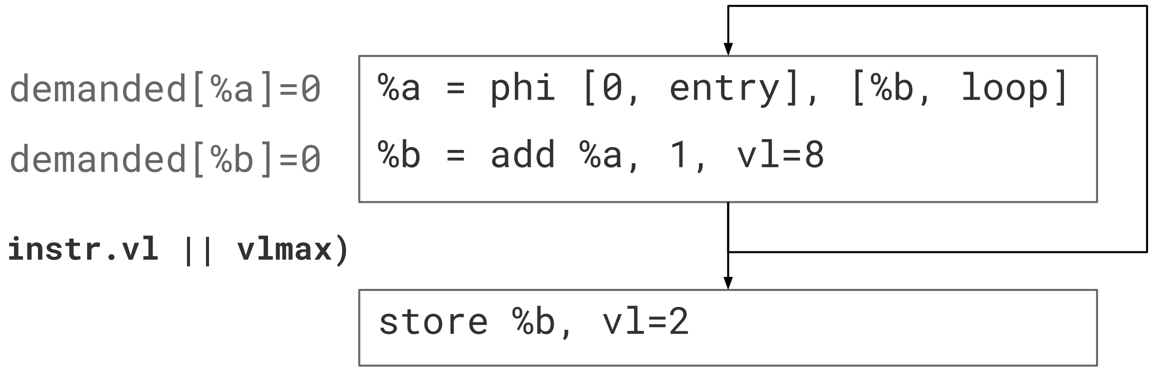
```
demanded = {}, worklist = {}
for instr in func:
  demanded[instr] = instr.vl || vlmax
for block in RPOT(func):
  for instr in reverse(block):
    worklist.insert(instr)

while instr = worklist.pop_front():
  prev = demanded[instr]
  demanded[instr] = max(demanded[users(instr)])
  if demanded[instr] != prev
    worklist += instr.ops()


for instr in block:
  if (demanded[instr] < x.vl):
    x.vl = demanded[instr]
```



```
%a = phi [0, entry], [%b, loop]
%b = add %a, 1, vl=8
```

```
store %b, vl=2
```

```
demanded = {}, worklist = {}
for def in defs(func):
  demanded[def] = 0
for block in RPOT(func):
  for instr in reverse(block):
    worklist.insert(instr)

while instr = worklist.pop_front():
  demands = min(demanded[instr] || vlmax, instr.vl || vlmax)
  for op in instr.ops():
    prev = demanded[op]
    demanded[op] = max(prev, demands)
    if demanded[op] != prev
      worklist += [op]


for instr in block:
  if (demanded[instr] < x.vl):
    x.vl = demanded[instr]
```

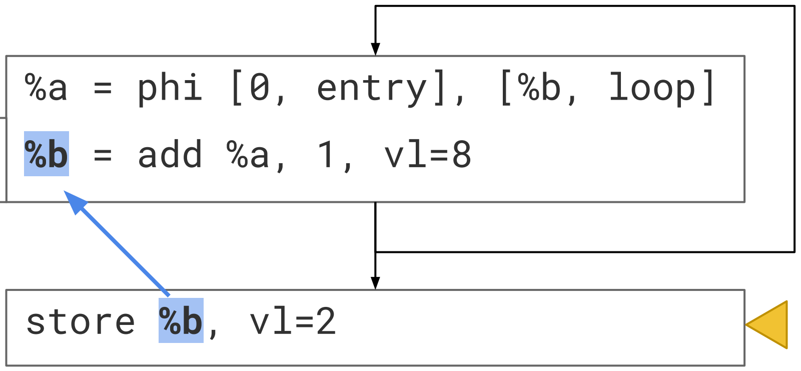


```
demanded[%a]=0    %a = phi [0, entry], [%b, loop]
demanded[%b]=0    %b = add %a, 1, vl=8

                  store %b, vl=2
```

```
demanded = {}, worklist = {}
for def in defs(func):
  demanded[def] = 0
for block in RPOT(func):
  for instr in reverse(block):
    worklist.insert(instr)

while instr = worklist.pop_front():
  demands = min(demanded[instr] || vlmax, instr.vl || vlmax)
  for op in instr.ops():
    prev = demanded[op]
    demanded[op] = max(prev, demands)
    if demanded[op] != prev
      worklist += [op]


for instr in block:
  if (demanded[instr] < x.vl):
    x.vl = demanded[instr]
```

demanded[%a]=0

demanded[%b]=max(0, min(vlmax, 2))=2



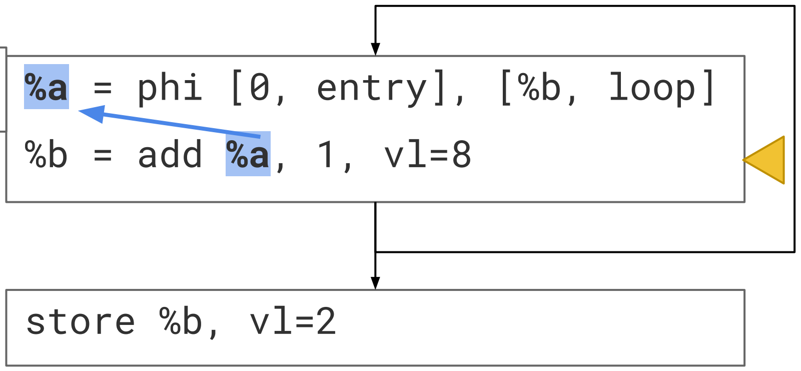
```
%a = phi [0, entry], [%b, loop]
%b = add %a, 1, vl=8
```

```
store %b, vl=2
```

```
demanded = {}, worklist = {}
for def in defs(func):
  demanded[def] = 0
for block in RPOT(func):
  for instr in rever
    worklist.insert(

while instr = worklist.pop_front():
  demands = min(demanded[instr] || vlmax, instr.vl || vlmax)
  for op in instr.ops():
    prev = demanded[op]
    demanded[op] = max(prev, demands)
    if demanded[op] != prev
      worklist += [op]


for instr in block:
  if (demanded[instr] < x.vl):
    x.vl = demanded[instr]
```

demanded[%a]=max(0, min(2, 8))=2

demanded[%b]=2

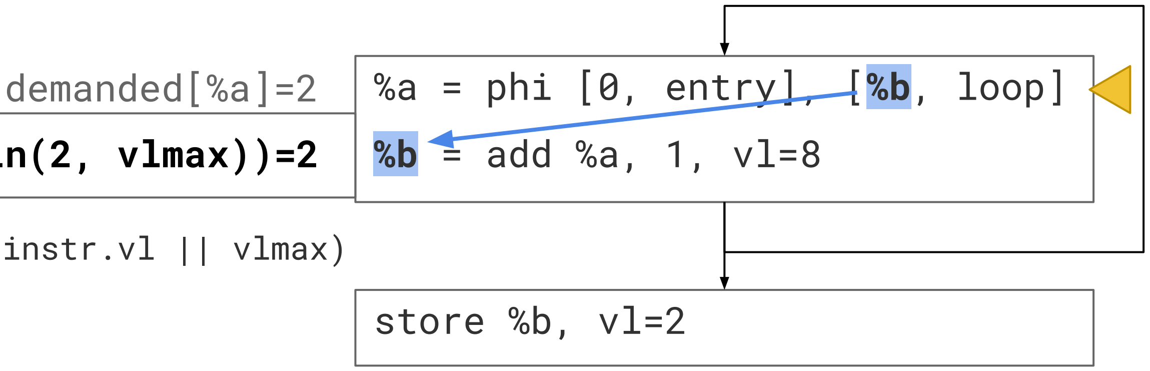


```
%a = phi [0, entry], [%b, loop]
%b = add %a, 1, vl=8
```

```
store %b, vl=2
```

```
demanded = {}, worklist = {}
for def in defs(func):
  demanded[def] = 0
for block in RPOT(func):
  for instr in reverse(block):
    worklist.ins

while instr = worklist.pop_front():
  demands = min(demanded[instr] || vlmax, instr.vl || vlmax)
  for op in instr.ops():
    prev = demanded[op]
    demanded[op] = max(prev, demands)
    if demanded[op] != prev
      worklist += [op]


for instr in block:
  if (demanded[instr] < x.vl):
    x.vl = demanded[instr]
```

demanded[%b]=max(2, min(2, vlmax))=2

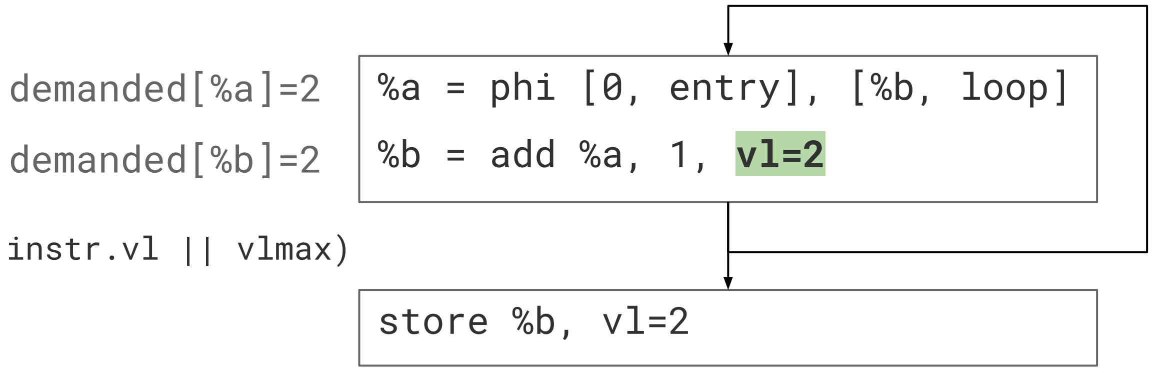


demanded[%a]=2

```
%a = phi [0, entry], [%b, loop]
%b = add %a, 1, vl=8
```

```
store %b, vl=2
```

```
demanded = {}, worklist = {}
for def in defs(func):
  demanded[def] = 0
for block in RPOT(func):
  for instr in reverse(block):
    worklist.insert(instr)

while instr = worklist.pop_front():
  demands = min(demanded[instr] || vlmax, instr.vl || vlmax)
  for op in instr.ops():
    prev = demanded[op]
    demanded[op] = max(prev, demands)
    if demanded[op] != prev
      worklist += [op]


for instr in block:
  if (demanded[instr] < x.vl):
    x.vl = demanded[instr]
```



```
demanded[%a]=2    %a = phi [0, entry], [%b, loop]
demanded[%b]=2    %b = add %a, 1, vl=2

                  store %b, vl=2
```

```
demanded = {}, worklist = {}
for def in defs(func):
  demanded[def] = 0
for block in RPOT(func):
  for instr in reverse(block):
    worklist.insert(instr)

while instr = worklist.pop_front():
  demands = min(demanded[instr] || vlmax, instr.vl || vlmax)
  for op in instr.ops():
    prev = demanded[op]
    demanded[op] = max(prev, demands)
    if demanded[op] != prev
      worklist += [op]


for instr in block:
  if (demanded[instr] < x.vl):
    x.vl = demanded[instr]
```

```
demanded = {}, worklist = {}
for def in defs(func):
  demanded[def] = 0
for block in RPOT(func):
  for instr in reverse(block):
    worklist.insert(instr)

while instr = worklist.pop_front():
  demands = min(demanded[instr] || vlmax, instr.vl || vlmax)
  for op in instr.ops():
    prev = demanded[op]
    demanded[op] = max(prev, demands)
    if demanded[op] != prev
      worklist += [op]


for instr in block:
  if (demanded[instr] < x.vl):
    x.vl = demanded[instr]
```

Optimistic

$$S = \{a \Rightarrow 3, b \Rightarrow \text{vlmax}, \ldots, z \Rightarrow 4\}$$

$$\bot = \forall x \in \text{defs}, \{x \Rightarrow 0\}$$

$$f(i, S) = \forall x \in \text{defs}, \{x \Rightarrow \begin{cases} \max(S[x], \min(S[i], \text{vl}(i))) & \text{if } x \in \text{ops}(i) \\ S[x] & \text{otherwise} \end{cases} \}$$

Dataflow Analysis!

Sparse

Set of possible demanded VLs forms a semilattice

$$S_1 \leq S_2 = \forall x \in \text{defs}, S_1[x] \leq S_2[x]$$

$$S_1 \vee S_2 = \forall x \in \text{defs}, x \Rightarrow \max(S_1[x], S_2[x])$$

$$\begin{tikzcd}
 & & \top = \{a \Rightarrow \text{vlmax}, b \Rightarrow \text{vlmax}\} \arrow[dll, dash] \arrow[drr, dash] & & \\
\{a \Rightarrow \text{vlmax}, b \Rightarrow 4\} \arrow[d, dash] \arrow[drr, dash] & & & & \{a \Rightarrow 4, b \Rightarrow \text{vlmax}\} \arrow[dll, dash] \arrow[d, dash] \\
\{a \Rightarrow \text{vlmax}, b \Rightarrow 0\} \arrow[d, dash] & & \{a \Rightarrow 4, b \Rightarrow 4\} \arrow[dll, dash] \arrow[drr, dash] & & \{a \Rightarrow 0, b \Rightarrow \text{vlmax}\} \arrow[d, dash] \\
\{a \Rightarrow 4, b \Rightarrow 0\} \arrow[drr, dash] & & & & \{a \Rightarrow 0, b \Rightarrow 4\} \arrow[dll, dash] \\
 & & \bot = \{a \Rightarrow 0, b \Rightarrow 0\} & &
\end{tikzcd}$$

$$\begin{tikzcd}
 & \top = \{a \Rightarrow \text{vlmax}, b \Rightarrow \text{vlmax}\} \arrow[dl, no head] \arrow[dr, no head] & \\
\{a \Rightarrow \text{vlmax}, b \Rightarrow 4\} \arrow[d, no head] \arrow[dr, no head] & & \{a \Rightarrow 4, b \Rightarrow \text{vlmax}\} \arrow[dl, no head] \arrow[d, no head] \\
\{a \Rightarrow \text{vlmax}, b \Rightarrow 0\} \arrow[d, no head] & \{a \Rightarrow 4, b \Rightarrow 4\} \arrow[dl, no head] \arrow[dr, no head] & \{a \Rightarrow 0, b \Rightarrow \text{vlmax}\} \arrow[d, no head] \\
\{a \Rightarrow 4, b \Rightarrow 0\} \arrow[dr, no head] & & \{a \Rightarrow 0, b \Rightarrow 4\} \arrow[dl, no head] \\
 & \bot = \{a \Rightarrow 0, b \Rightarrow 0\} &
\end{tikzcd}$$

Does our loop terminate?

```
while instr = worklist.pop_front():
  demands = min(demanded[instr] || vlmax, instr.vl || vlmax)
  for op in instr.ops():
    prev = demanded[op]
    demanded[op] = max(prev, demands)
    if demanded[op] != prev
      worklist += [op]
```

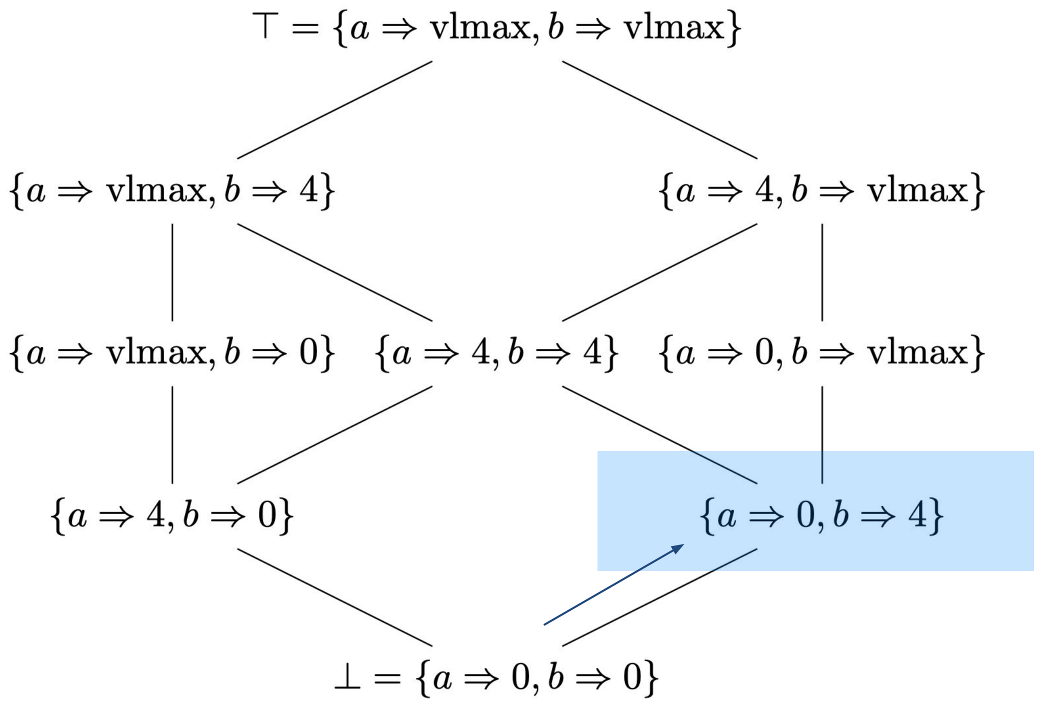
```
%a = add 1,  1, vl=vlmax
%b = add %a, 1, vl=vlmax
store %b, vl=4
```

```
demanded[%a]=0
demanded[%b]=0
```

$$\begin{tikzcd}
 & \top = \{a \Rightarrow \text{vlmax}, b \Rightarrow \text{vlmax}\} \arrow[dl, dash] \arrow[dr, dash] & \\
\{a \Rightarrow \text{vlmax}, b \Rightarrow 4\} \arrow[d, dash] \arrow[dr, dash] & & \{a \Rightarrow 4, b \Rightarrow \text{vlmax}\} \arrow[dl, dash] \arrow[d, dash] \\
\{a \Rightarrow \text{vlmax}, b \Rightarrow 0\} \arrow[d, dash] & \{a \Rightarrow 4, b \Rightarrow 4\} \arrow[dl, dash] \arrow[dr, dash] & \{a \Rightarrow 0, b \Rightarrow \text{vlmax}\} \arrow[d, dash] \\
\{a \Rightarrow 4, b \Rightarrow 0\} \arrow[dr, dash] & & \{a \Rightarrow 0, b \Rightarrow 4\} \arrow[dl, dash] \\
 & \bot = \{a \Rightarrow 0, b \Rightarrow 0\} &
\end{tikzcd}$$

```
%a = add 1,  1, vl=vlmax
%b = add %a, 1, vl=vlmax
store %b, vl=4
```

```
demanded[%a]=0
demanded[%b]=4
```



$\top = \{a \Rightarrow \text{vlmax}, b \Rightarrow \text{vlmax}\}$
$\{a \Rightarrow \text{vlmax}, b \Rightarrow 4\}$
$\{a \Rightarrow 4, b \Rightarrow \text{vlmax}\}$
$\{a \Rightarrow \text{vlmax}, b \Rightarrow 0\}$
$\{a \Rightarrow 4, b \Rightarrow 4\}$
$\{a \Rightarrow 0, b \Rightarrow \text{vlmax}\}$
$\{a \Rightarrow 4, b \Rightarrow 0\}$
$\{a \Rightarrow 0, b \Rightarrow 4\}$
$\bot = \{a \Rightarrow 0, b \Rightarrow 0\}$

```
%a = add 1,  1, vl=vlmax
%b = add %a, 1, vl=vlmax
store %b, vl=4
```

demanded[%a]=4

demanded[%b]=4

$$\begin{tikzcd}
 & \top = \{a \Rightarrow \text{vlmax}, b \Rightarrow \text{vlmax}\} \arrow[dl, dash] \arrow[dr, dash] & \\
\{a \Rightarrow \text{vlmax}, b \Rightarrow 4\} \arrow[d, dash] \arrow[dr, dash] & & \{a \Rightarrow 4, b \Rightarrow \text{vlmax}\} \arrow[dl, dash] \arrow[d, dash] \\
\{a \Rightarrow \text{vlmax}, b \Rightarrow 0\} \arrow[d, dash] & \boxed{\{a \Rightarrow 4, b \Rightarrow 4\}} \arrow[dl, dash] \arrow[dr, dash] & \{a \Rightarrow 0, b \Rightarrow \text{vlmax}\} \arrow[d, dash] \\
\{a \Rightarrow 4, b \Rightarrow 0\} \arrow[dr, dash] & & \{a \Rightarrow 0, b \Rightarrow 4\} \arrow[dl, dash] \arrow[ul, bend right] \\
 & \bot = \{a \Rightarrow 0, b \Rightarrow 0\} \arrow[ur, bend right] &
\end{tikzcd}$$

We know the analysis will terminate if:

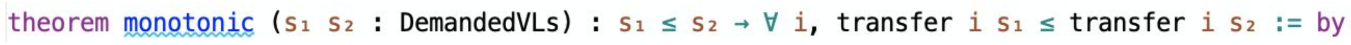
- The height of the semilattice is **finite:** height = num defs * num unique vl values
- The transfer function is **monotonic**: it never computes a smaller demanded vl

$$S_1 \leq S_2 \rightarrow \forall i, f(i, S_1) \leq f(i, S_2)$$

$$S_1 \leq S_2 = \forall x \in \text{defs}, S_1[x] \leq S_2[x]$$

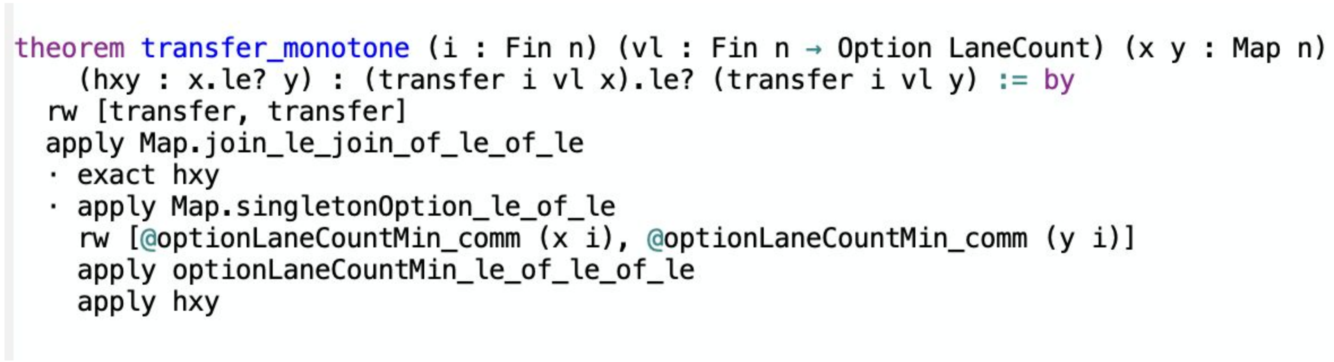
$$f(i, S) = \forall x \in \text{defs}, \{x \Rightarrow \begin{cases} \max(S[x], \min(S[i], \text{vl}(i))) & \text{if } x \in \text{ops}(i) \\ S[x] & \text{otherwise} \end{cases} \}$$

- Patches for RISCVVLOptimizer hopefully posted soon
- Hard to evaluate usefulness of full dataflow analysis at the moment: EVL tail folding codegen is still under development
- Future work: Formally verify properties in lean4 etc?



```
theorem monotonic (s₁ s₂ : DemandedVLs) : s₁ ≤ s₂ → ∀ i, transfer i s₁ ≤ transfer i s₂ := by
```

- Patches for RISCVVLOptimizer hopefully posted soon
- Hard to evaluate usefulness of full dataflow analysis at the moment: EVL tail folding codegen is still under development
- ~~Future work~~: Formally verified semilattice + monotonicity in lean4!



```
theorem transfer_monotone (i : Fin n) (vl : Fin n → Option LaneCount) (x y : Map n)
    (hxy : x.le? y) : (transfer i vl x).le? (transfer i vl y) := by
  rw [transfer, transfer]
  apply Map.join_le_join_of_le_of_le
  · exact hxy
  · apply Map.singletonOption_le_of_le
    rw [@optionLaneCountMin_comm (x i), @optionLaneCountMin_comm (y i)]
    apply optionLaneCountMin_le_of_le_of_le
    apply hxy
```

Thank you Siddharth Bhat! 🙏

igalia