NIR: A new IR for Mesa

Connor Abbott
cwabbott0@gmail.com

Background

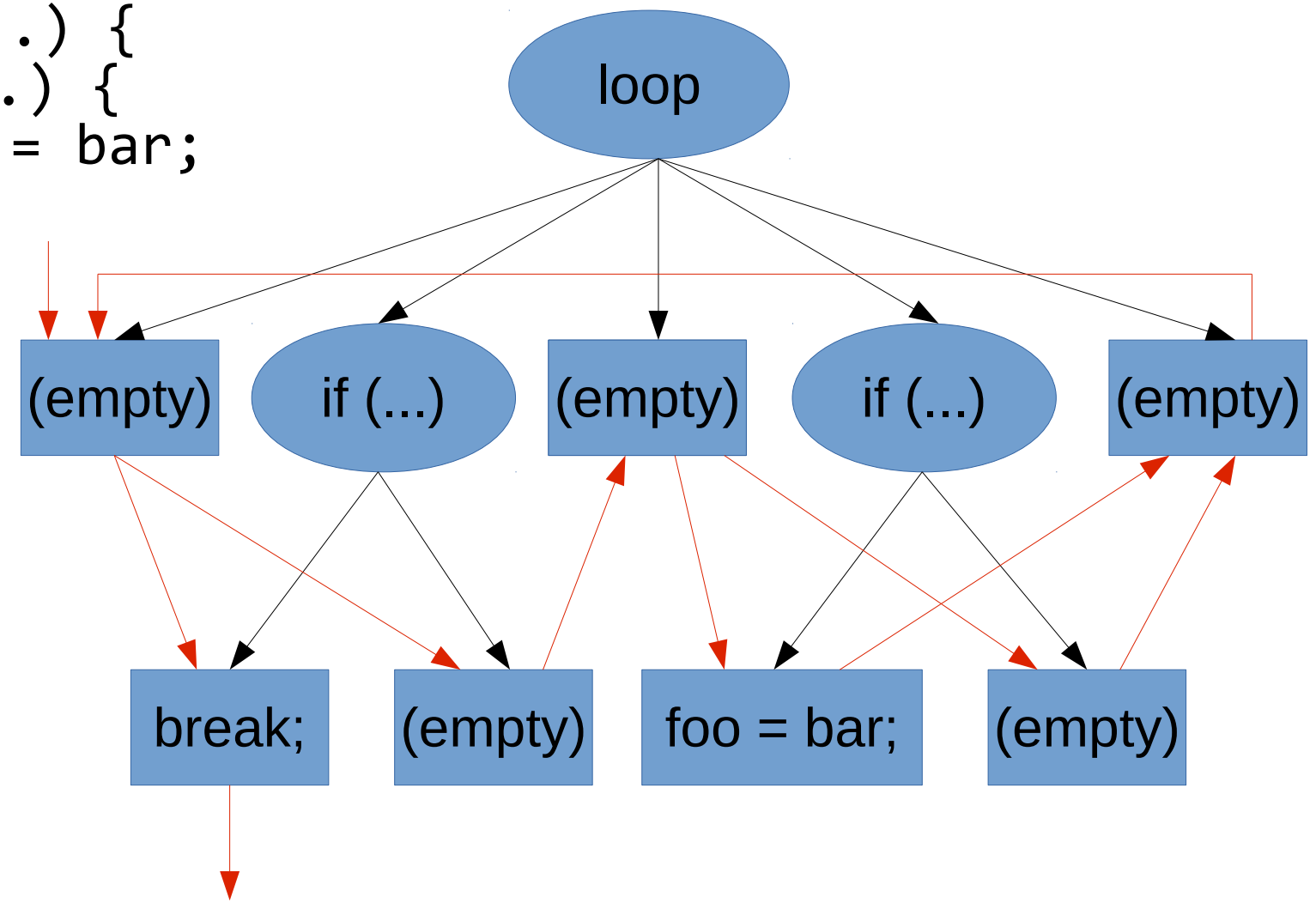
- GLSL IR SSA work
  - Yuck!
- Modify the IR?
  - Basic blocks, flat, SSA, ...
- Make a new one?

Goals

- Flat, SSA-based
- Basic block based, but...
- Preserve original control structure (loops, ifs)
- Support high-level GLSL things, but...
  - Linking, array splitting, structure splitting, etc.
- Also support low-level things
  - DRY – don't make drivers copy optimizations unless they have to!
- Try and make it extensible, but...
- Don't try and support everyone's wacky HW!

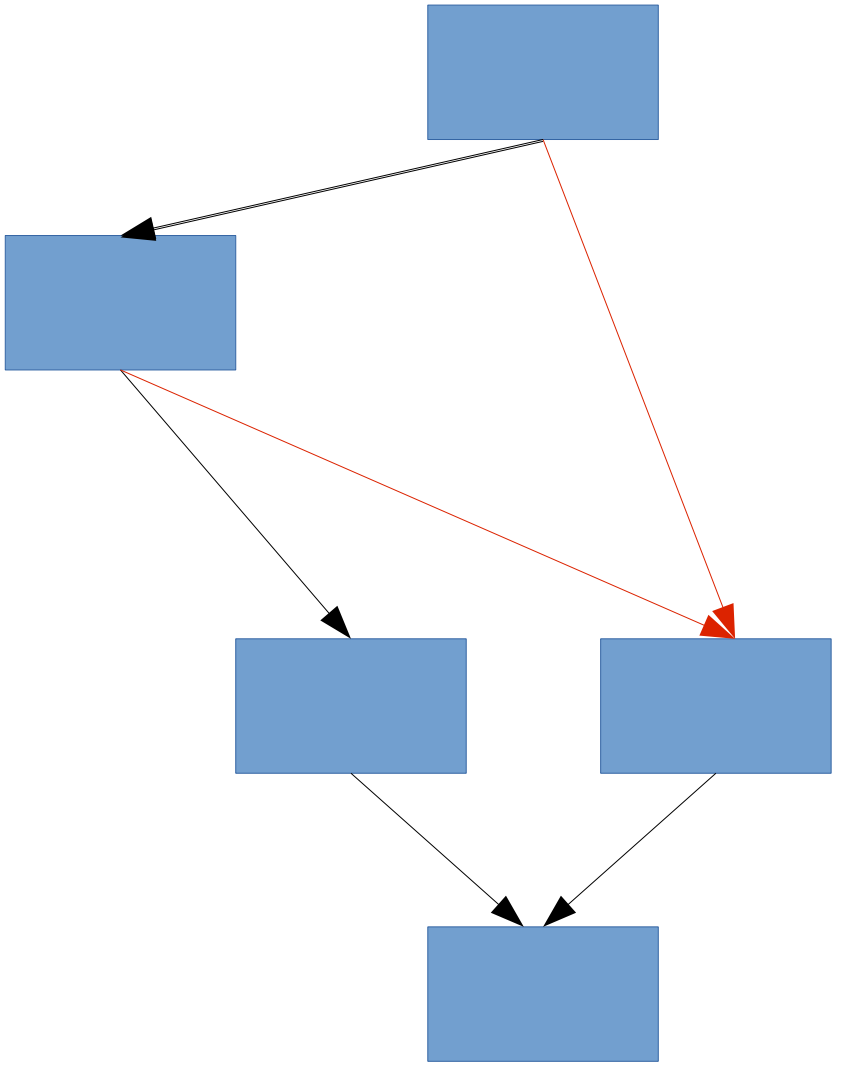
Control Flow

```
while (...) {
   if (...) {
      foo = bar;
   }
}
```



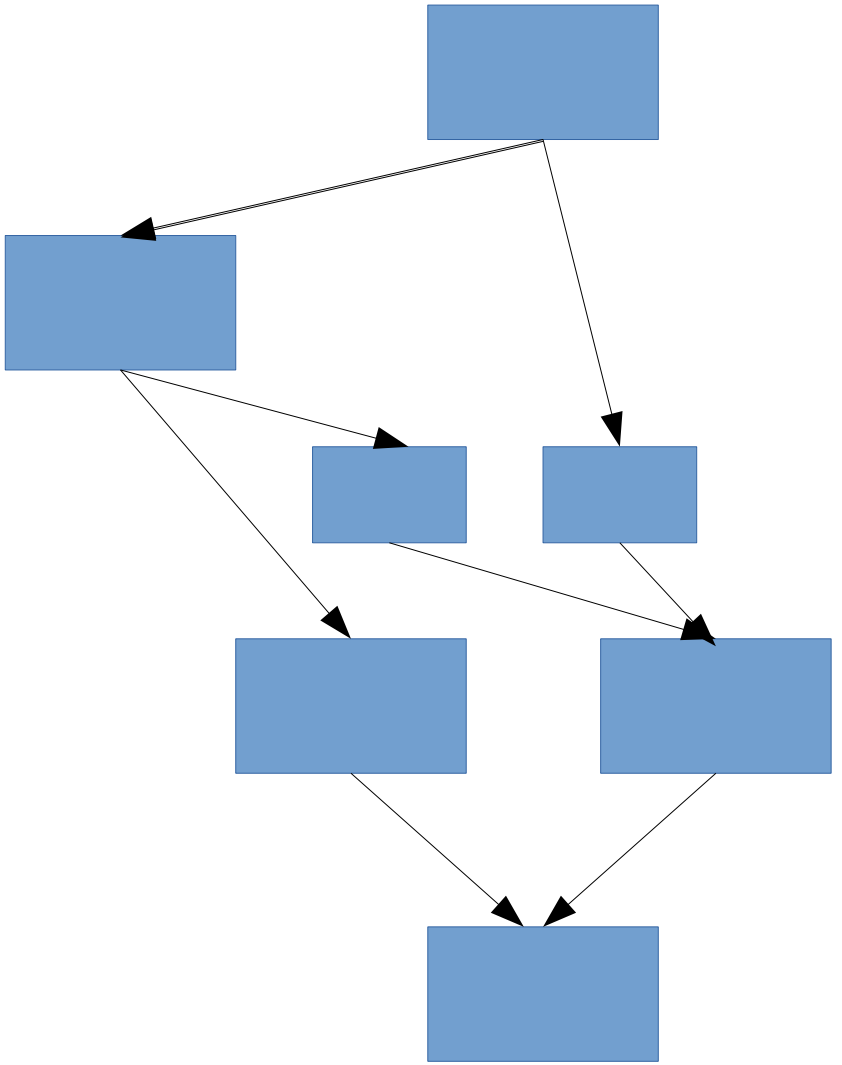
Critical Edges

```
if (...) {
   ...
}
if (...) {
   ...
} else {
   ...
}
```



Critical Edges

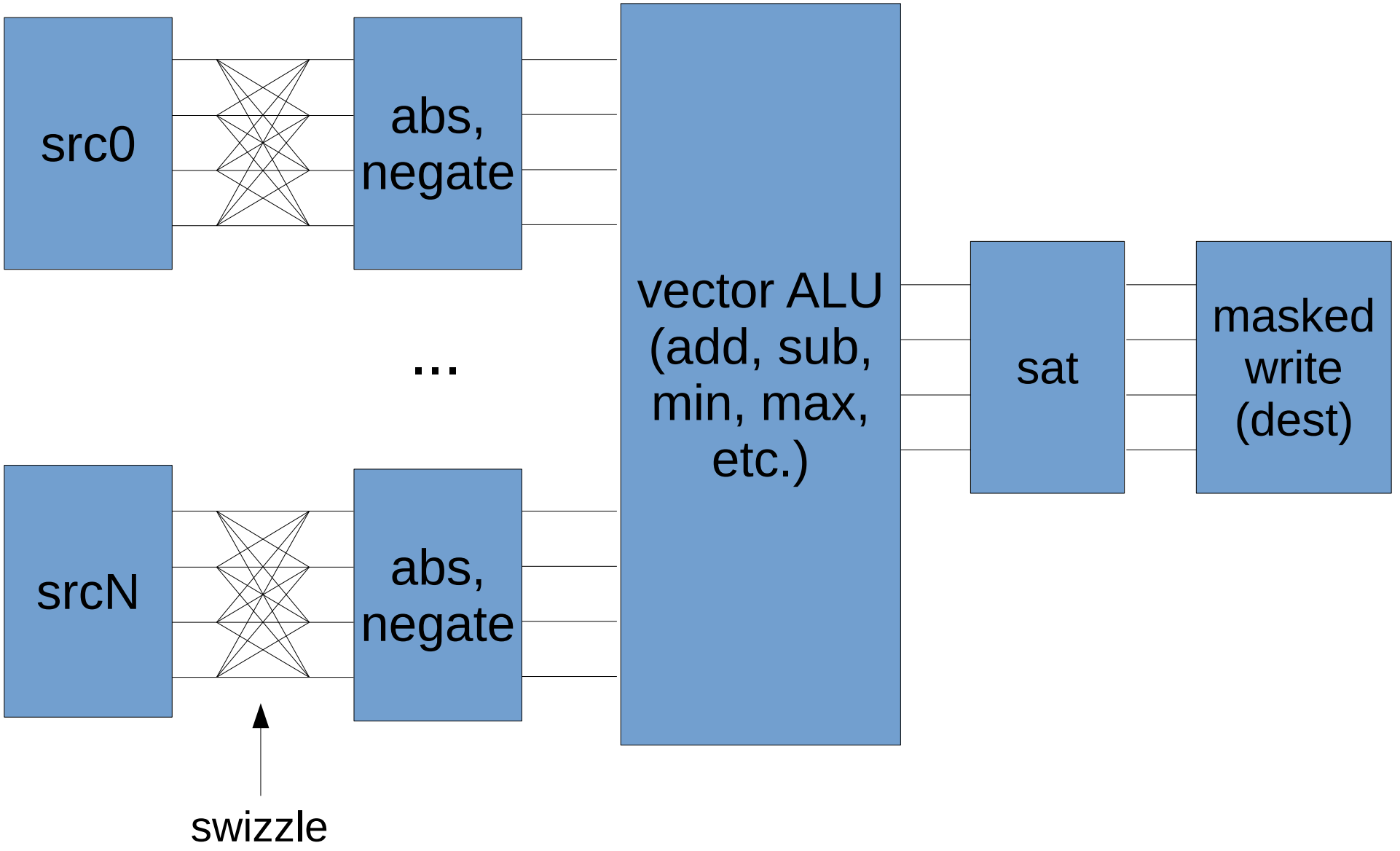
```
if (...) {
   ...
}
if (...) {
   ...
} else {
   ...
}
```



Traditional Vec4 Model

- Gallium (TGSI), i965 FS/GS, Mali 400/600
- Dying out (yay!) except for mobile
- Everything: varying interpolation, variable indexing etc. in terms of 4-vectors

Traditional Vec4 Pipeline



Vec4 + SSA = ??

- Masked writes not possible in SSA
  - No partial updates allowed
- Need a way to combine two SSA values
  - `foo.xy = bar;`
  - Becomes...
  - `newFoo = vec4(bar, oldFoo.zw);`
- Out of SSA becomes tricky

The Vec4 Pipeline/Modifiers in NIR

- Abs, negate, etc. easy for scalar backend
- But not for vec4 backends
- Don't want old backends to have to do more than necessary
- Have to fold abs, neg, etc. *before* out-of-SSA
  - Or keep around tons of metadata
- It's in there, but most of the time we can ignore it

Variables

- Mostly copied from variables in GLSL IR
- Ways to use them:
  - Intrinsics (load/store/copy, etc)
  - Function arguments & return values
  - Samplers for texture instructions (before lowering)
- Differences from LLVM
  - Entirely logical (no notion of alignment)
  - Can't take the address – no notion of pointers!
  - Supports GLSL concepts without extra metadata

Registers

- Main goals:
  - Make it easier for older backends that don't do SSA
  - Sharing code for lowering arrays of structures or structures with arrays
  - Avoiding backend optimizations to do with array indexing
- Can create an array of given vector width and size
- Or a scalar array that can be accessed anywhere
- Indexing (stride, alignment, etc.) up to backends

Intrinsics

- Almost everything that can't be constant folded
- 4 arguments:
  - Register/SSA value input
  - Register/SSA value output
  - Variables
  - Constant integers (indices, sizes, etc.)
- Additional semantic bits
  - Can we delete this intrinsic? Can we reorder it?
- Defined statically by macros (nir_intrinsics.c)

Todo

- Make i965 FS backend use SSA
- Add i965 Vec4 backend
- fp64?
- More precise semantics for intrinsics
- Optimizations!

Questions?