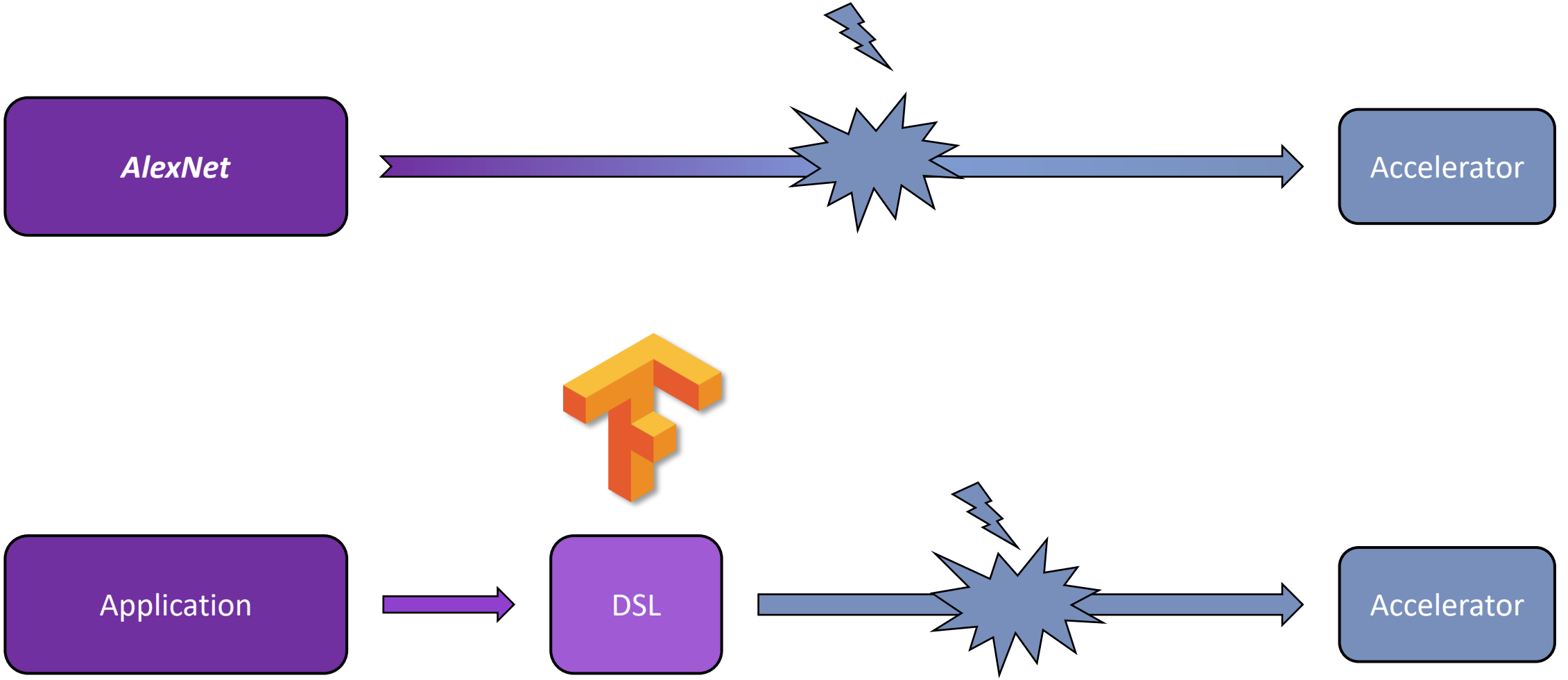
A Toolkit for Designing Hardware DSLs

Griffin Berlstein, Rachit Nigam, Chris Gyurgyik, Adrian Sampson

Computer **A**rchitecture and
Programming **A**bstractions Group

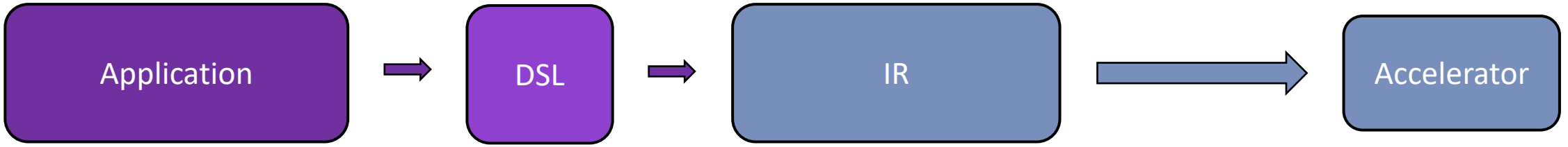
calyxir.org

Designing hardware accelerators is ***challenging***, particularly for ***domain experts*** from other fields



Lack of a ***robust compiler infrastructure*** hinders DSL development

Limited ***debugging tools*** hinder DSL adoption



Calyx

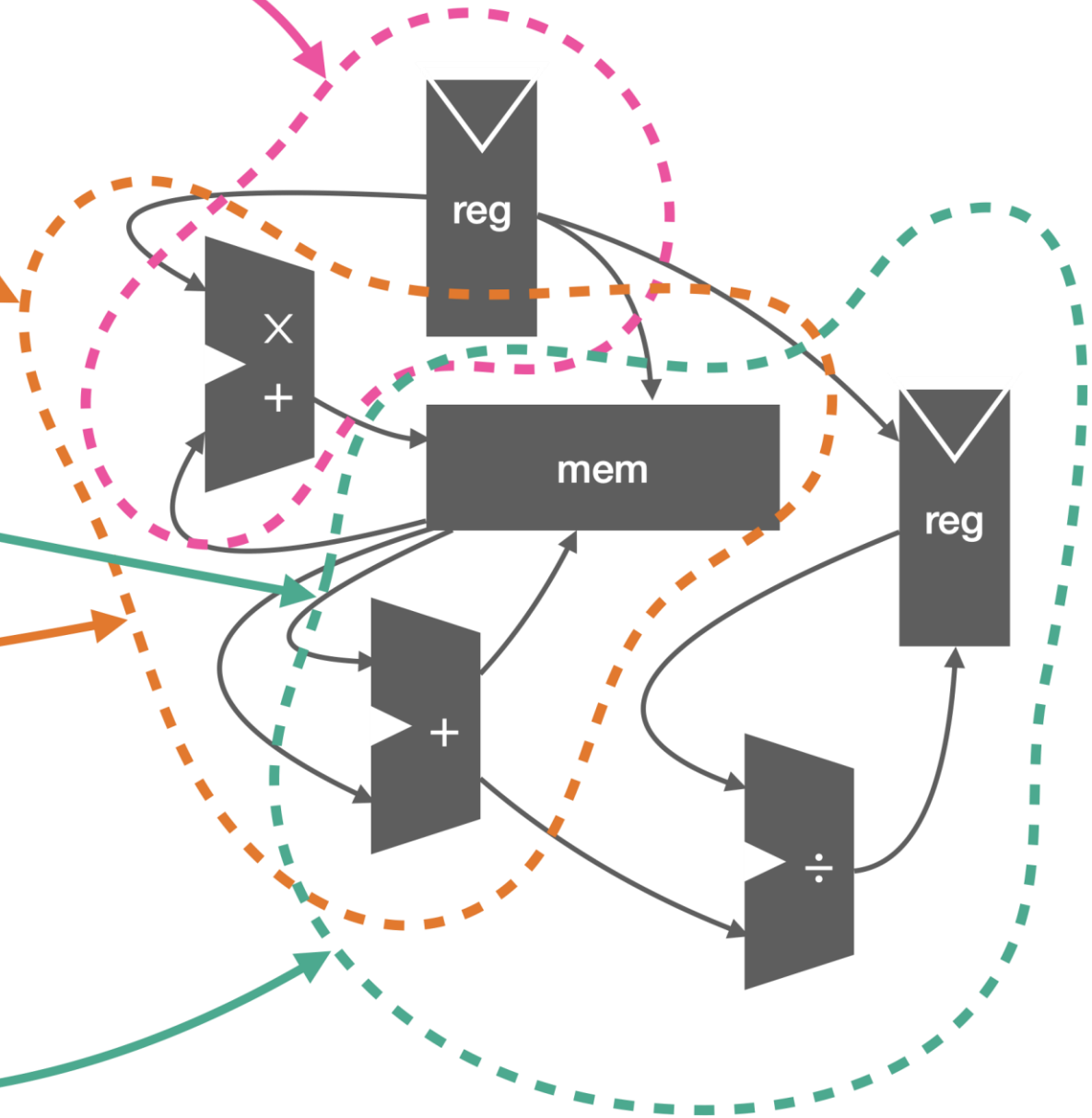
High-level control flow

Low-level structure

Image by Eucalyp

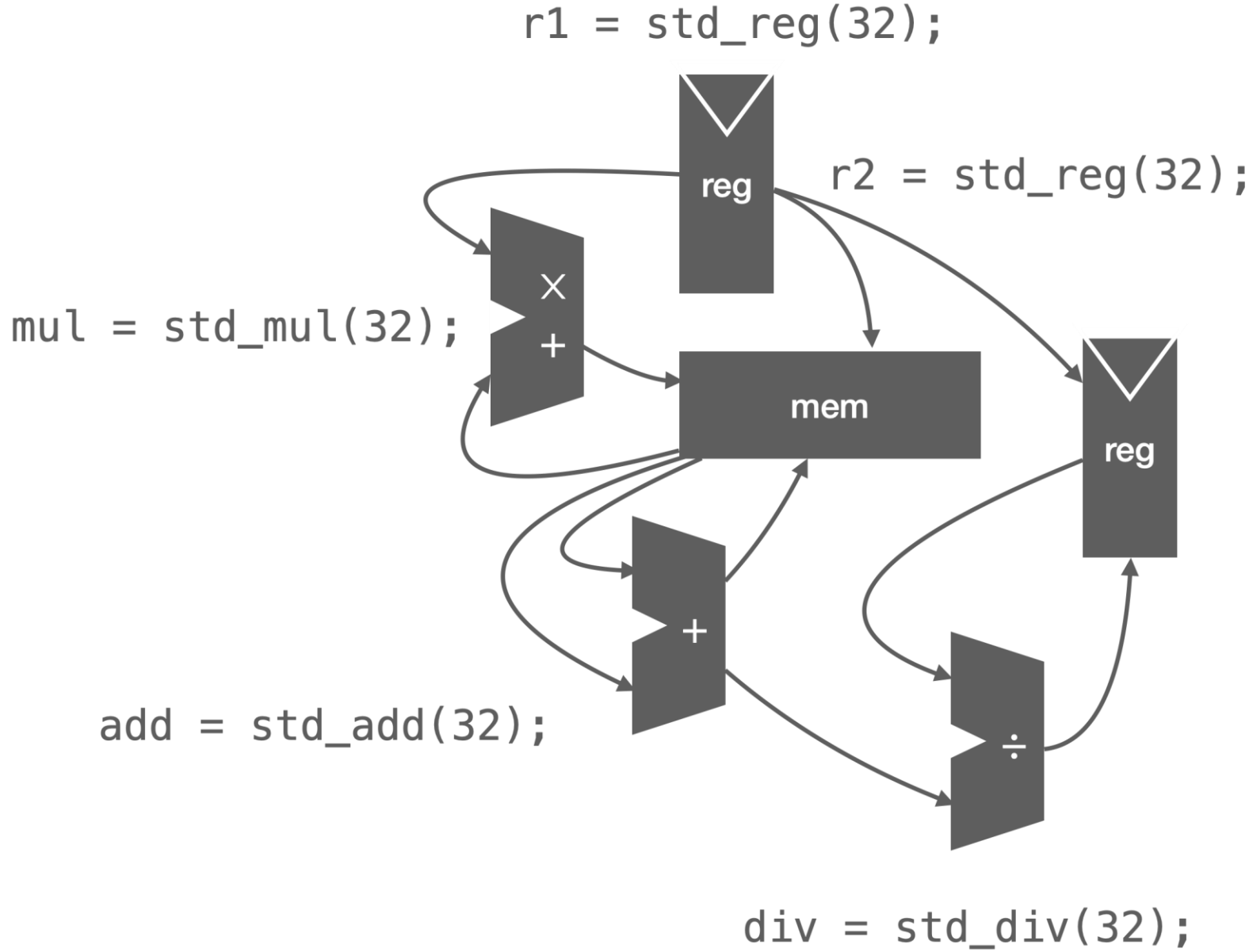
Calyx

```
if    {
  while   {
    }
} else {
  par {
    ;
    }
}
```



Calyx

```
if ●  {
   while ● {
      ●
   }
} else {
   par {
      ● ;
      ●
   }
}
```



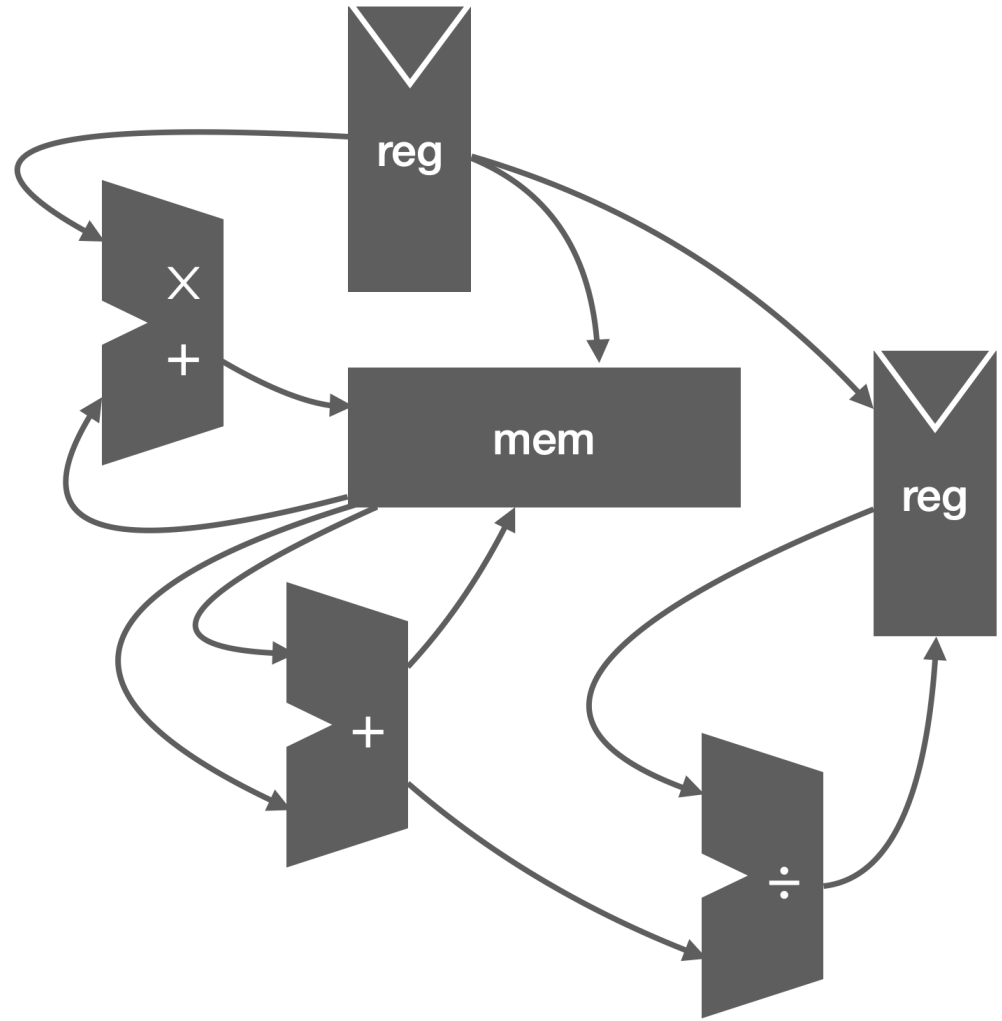
Calyx

```
if ● {
   while ● {
      ●
   }
} else {
   par {
      ● ;
      ●
   }
}
```

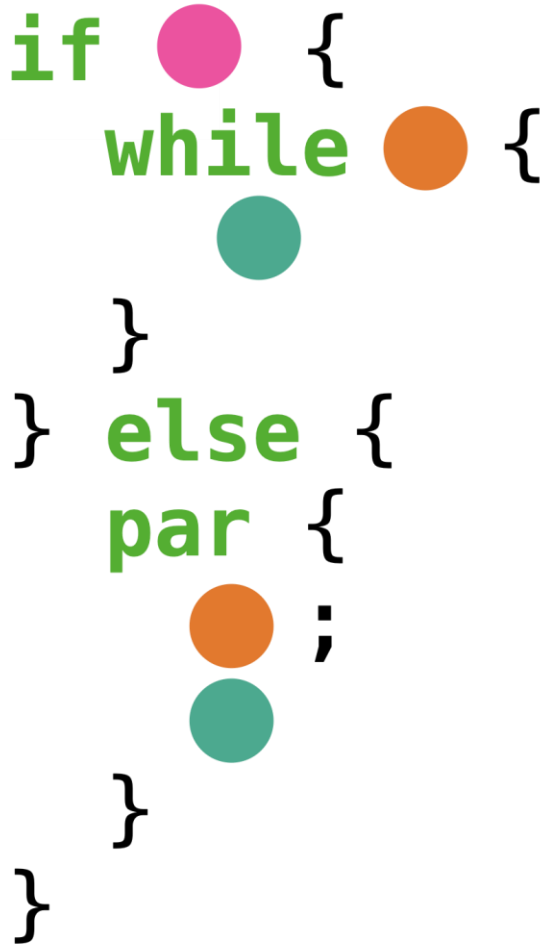
control

```
r1 = std_reg(32);
mul = std_mul(32);
r2 = std_reg(32);
add = std_add(32);
div = std_div(32);
```

cells



Calyx



control

```
r1 = std_reg(32);
mul = std_mul(32);
r2 = std_reg(32);
add = std_add(32);
div = std_div(32);
```

cells

```
group pink {
  mul.left = reg.out;
  mul.right = mem.out;
  mem.in = mul.out;
}

group orange {
  mem.in = add.out;
  add.left = mem.out;
  add.right = mem.out;
  ...

group green {
  ...
```

wires

Calyx

```
if pink {
  while orange {
    green;
  }
} else {
  par {
    orange;
    green;
  }
}
```

control

```
r1 = std_reg(32);
mul = std_mul(32);
r2 = std_reg(32);
add = std_add(32);
div = std_div(32);
```

cells

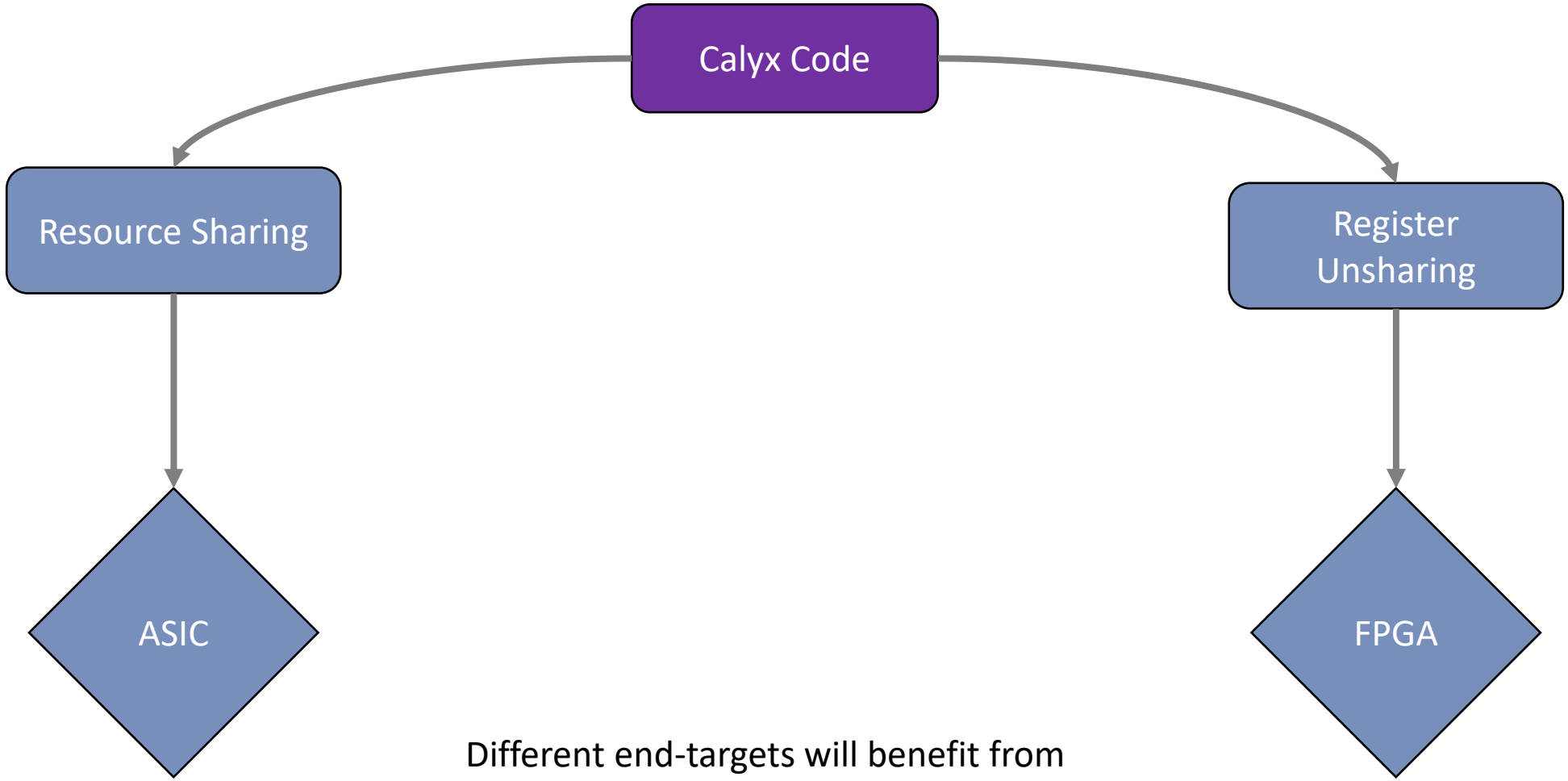
```
group pink {
  mul.left = reg.out;
  mul.right = mem.out;
  mem.in = mul.out;
}

group orange {
  mem.in = add.out;
  add.left = mem.out;
  add.right = mem.out;
  ...

group green {
  ...
```

wires

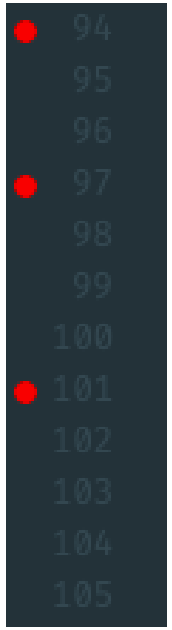
Optimizations in Calyx



Different end-targets will benefit from different sets of optimizations

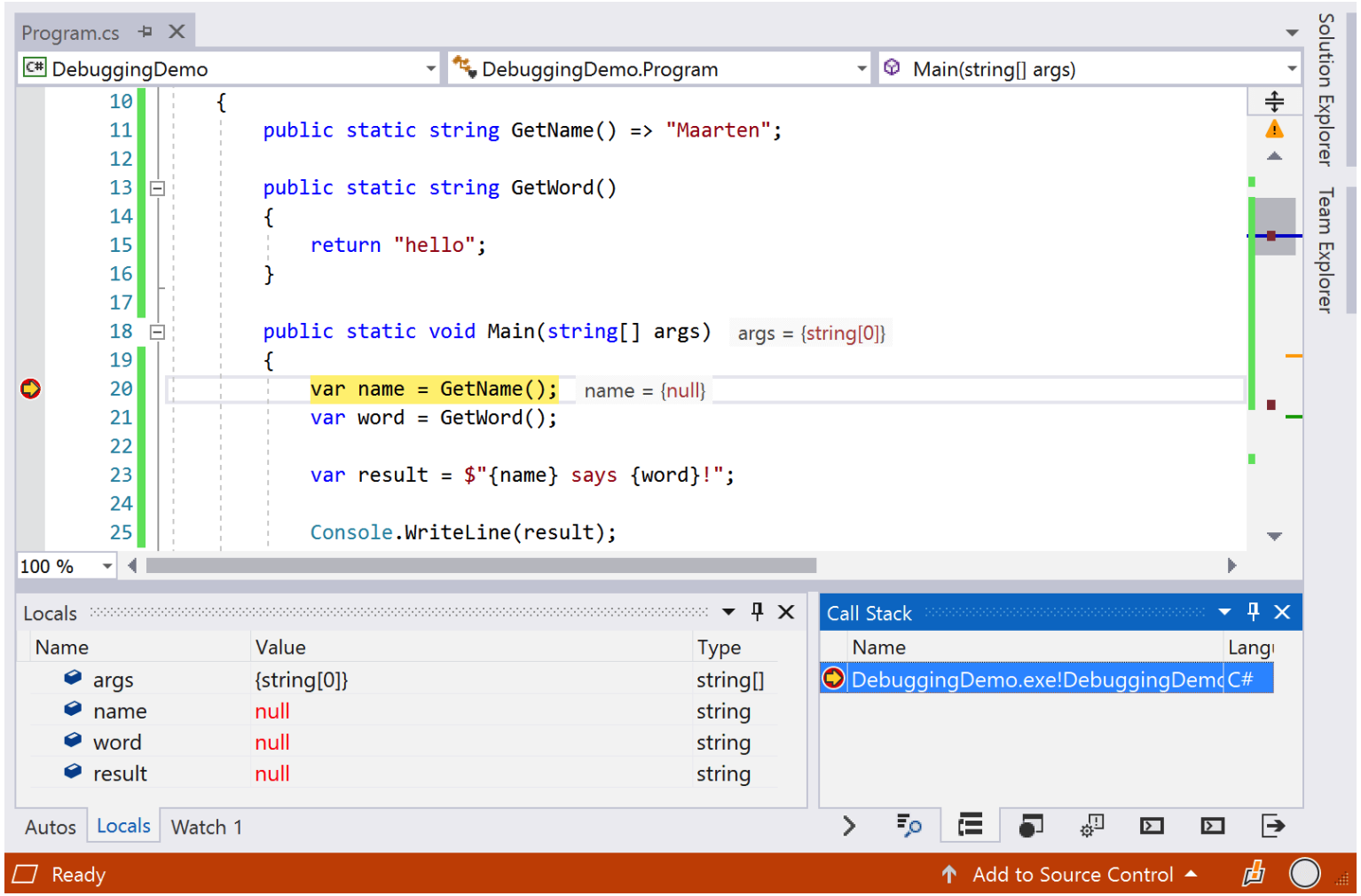
Debugging with Calyx

Debugging in Software



Breakpoints

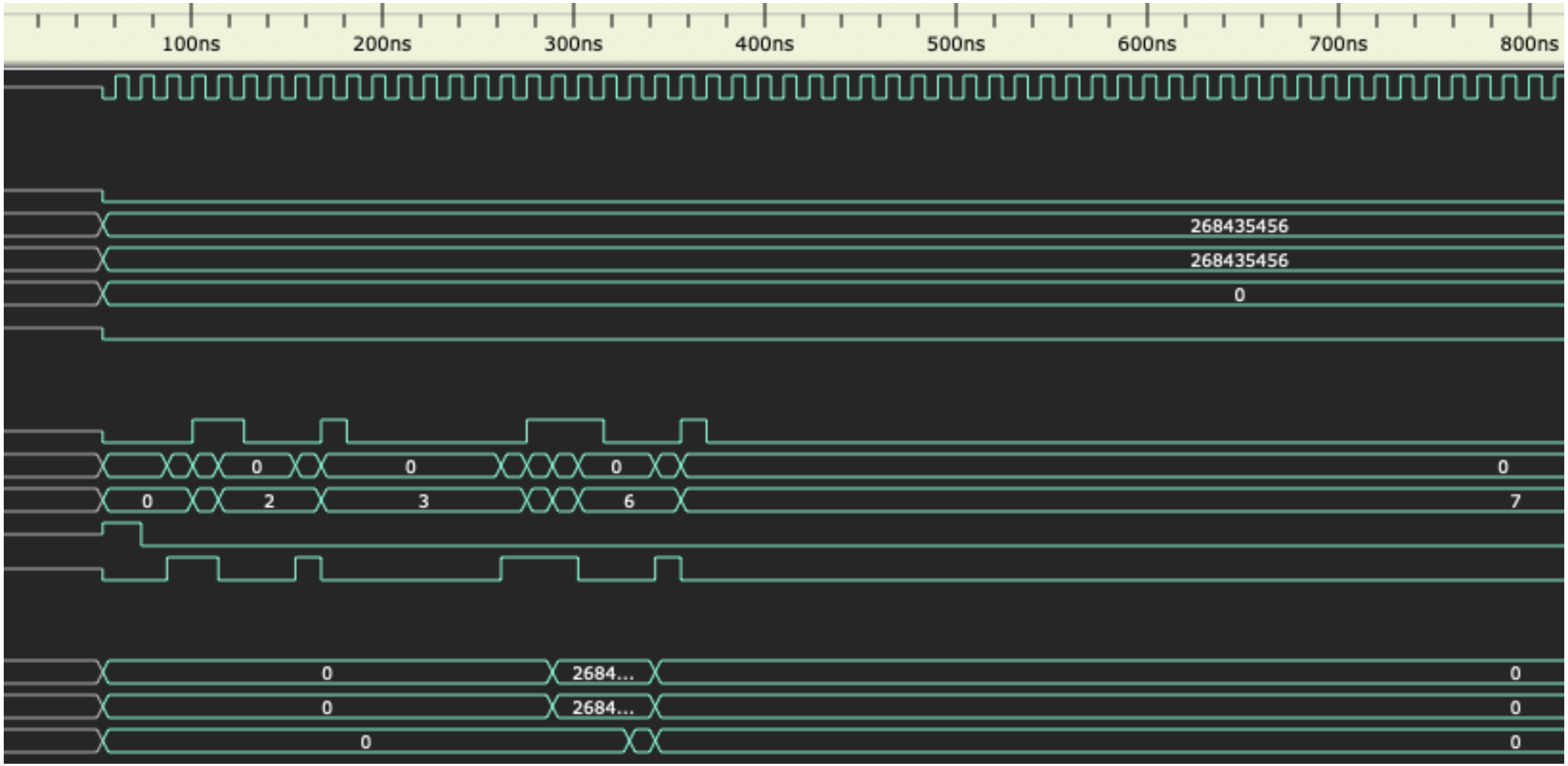
Step execution



Complex state inspection

Image from jetbrains blog

Debugging in Hardware



Debugging in Hardware

RTL doesn’t know about the ***control*** of the accelerator

- Any assignment could change at each clock cycle
- So, debugging tools must capture and display every change

Even simple accelerators can have many thousands of cycles!

Even worse: ***generated*** RTL does not resemble the surface application

We cannot expect domain experts to ***code in a DSL and debug in RTL***

RTL is too low-level for
DSL ***accelerator*** debugging

Instead debug at the level of the
Calyx IR

High-level control flow

Low-level structure

Calyx knows about control flow explicitly. This bypasses many of the challenges of RTL simulation

Image by Eucalyp

High-level control flow

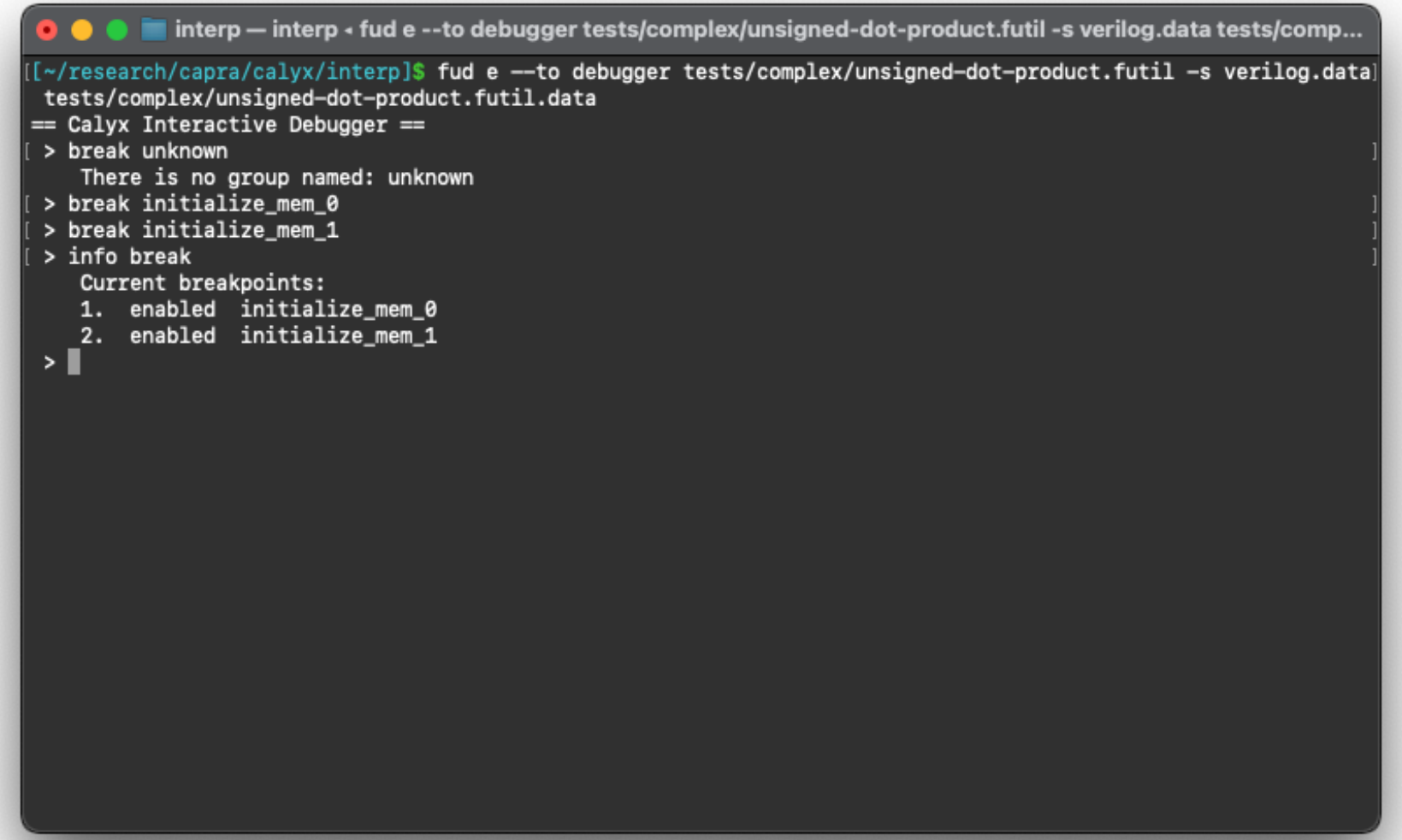
Low-level structure

Calyx represents many of the realities of hardware which are not visible when debugging at the DSL level (or C for HLS)

Image by Eucalyp

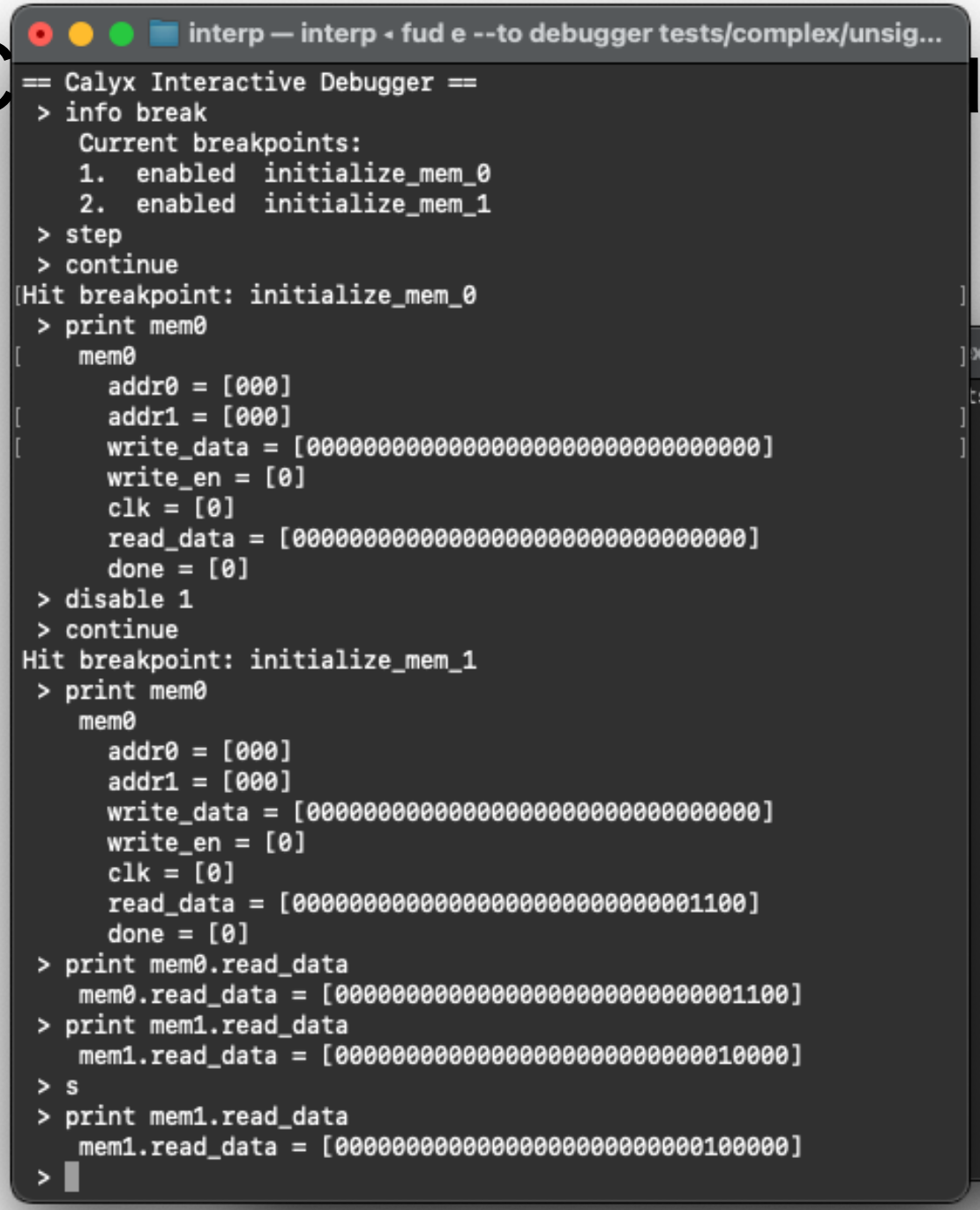
Calyx Interactive Debugger (*CIDR*)

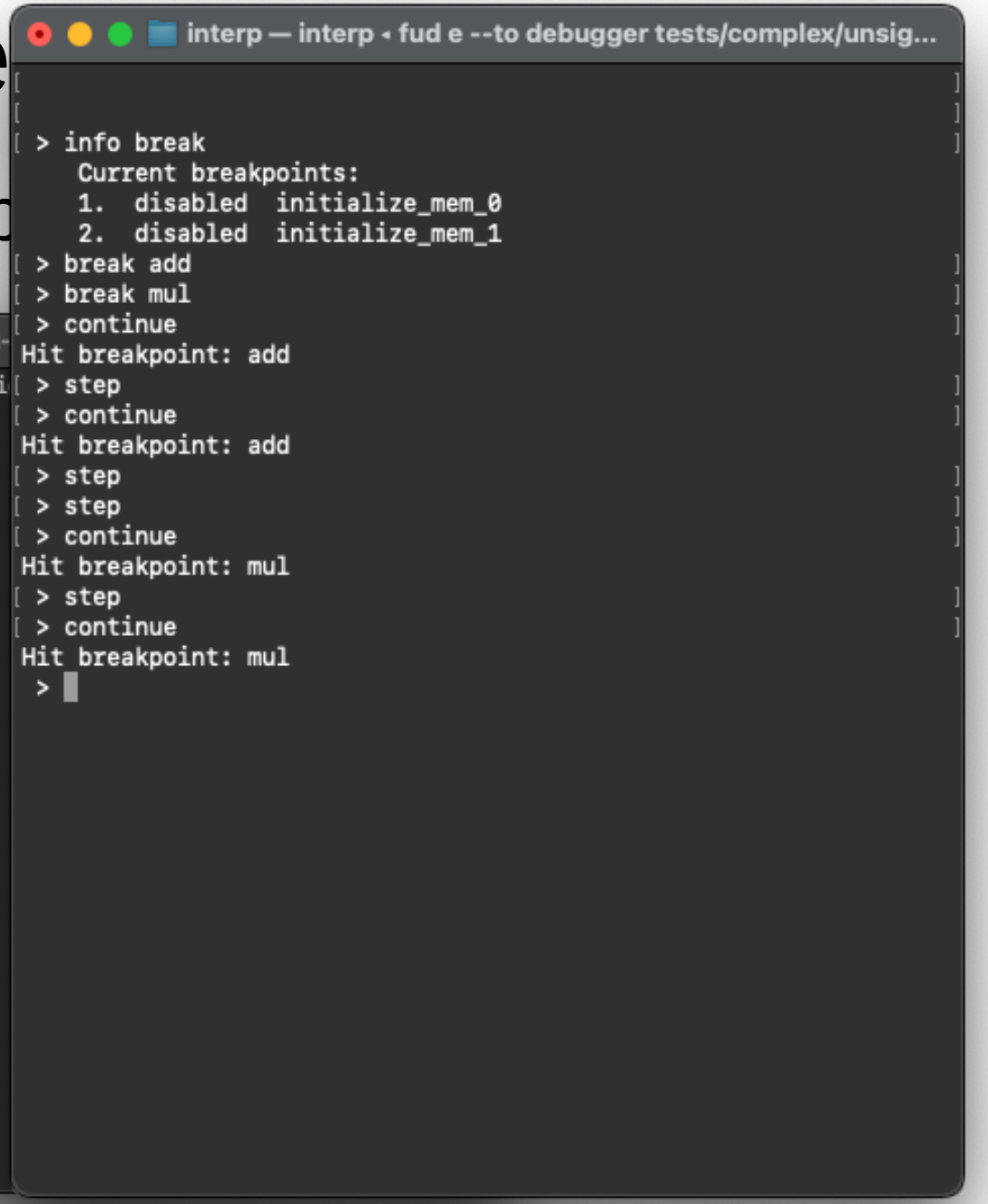
Built on top of the Calyx Interpreter



C ugge

Interp





Future extensions to ***CIDR***

- Data structure rematerialization for frontends
- Source Location support and gui interaction
- IR level debugging constructs/annotations
  - Watch values & asserts
- FPGA integration (capture error state from FPGA to debug in software)

- And much more in the broader Calyx ecosystem

Calyx

High-level control flow

Low-level structure

Image by Eucalyp

Thanks to my co-authors and the rest of the Calyx team!

calyxir.org