Novel Hardware Design Variation through Feature-Oriented Programming

Justin Deters

Washington University in St. Louis
j.deters@wustl.edu

Supported under NSF CISE award CNS-1763503 Performant Architecturally Diverse Systems via Aspect-oriented Programming

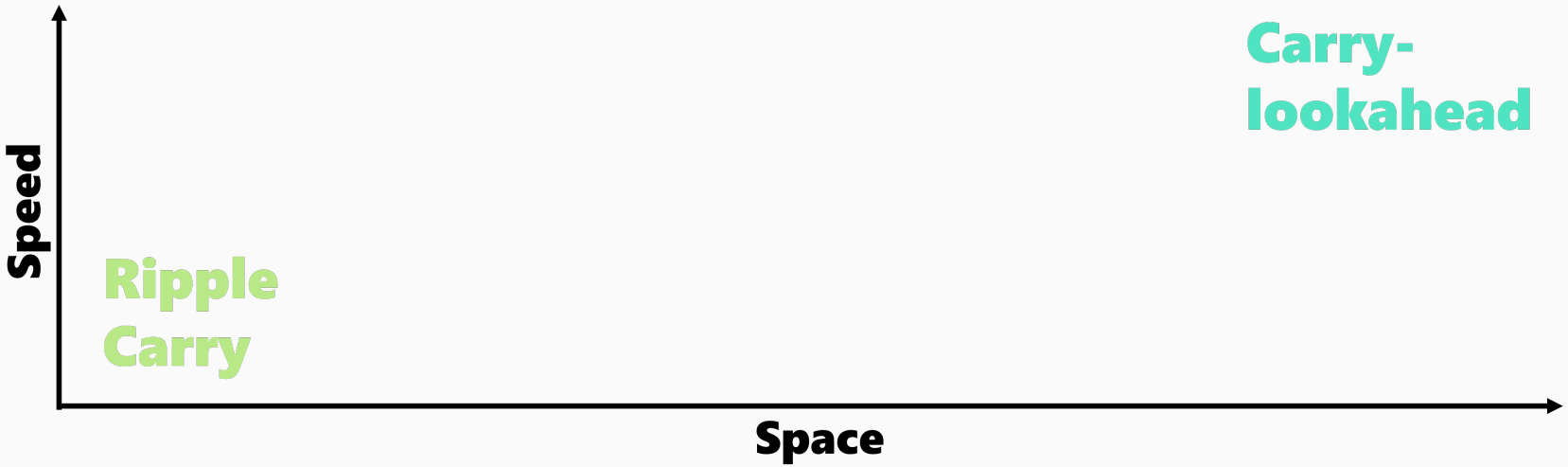
Traditional HDLs

- Hardware designers have to contend with low level structures in Hardware Description Languages (HDLs).
- Hardware designs tend to be monolithic with little flexibility.

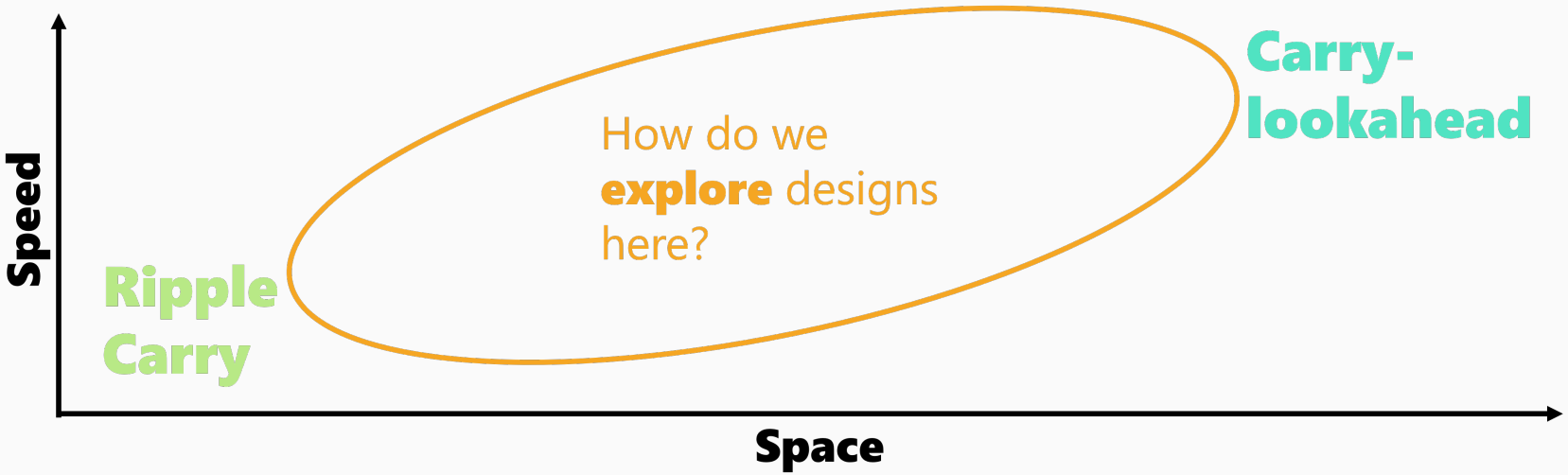
```
1  module OneBitAdder(
2    input    io_a,
3    input    io_b,
4    input    io_carryIn,
5    output   io_sum,
6    output   io_carryOut
7   );
8    wire   p = io_a ^ io_b;
9    wire   g = io_a & io_b;
10   wire   p_c = io_carryIn & p;
11   assign io_sum = p ^ io_carryIn;
12   assign io_carryOut = g | p_c;
13 endmodule
```

- Tedious and error prone to explore design spaces and optimize.
- We demonstrate this with a hardware adder.

Traditional Adder Designs

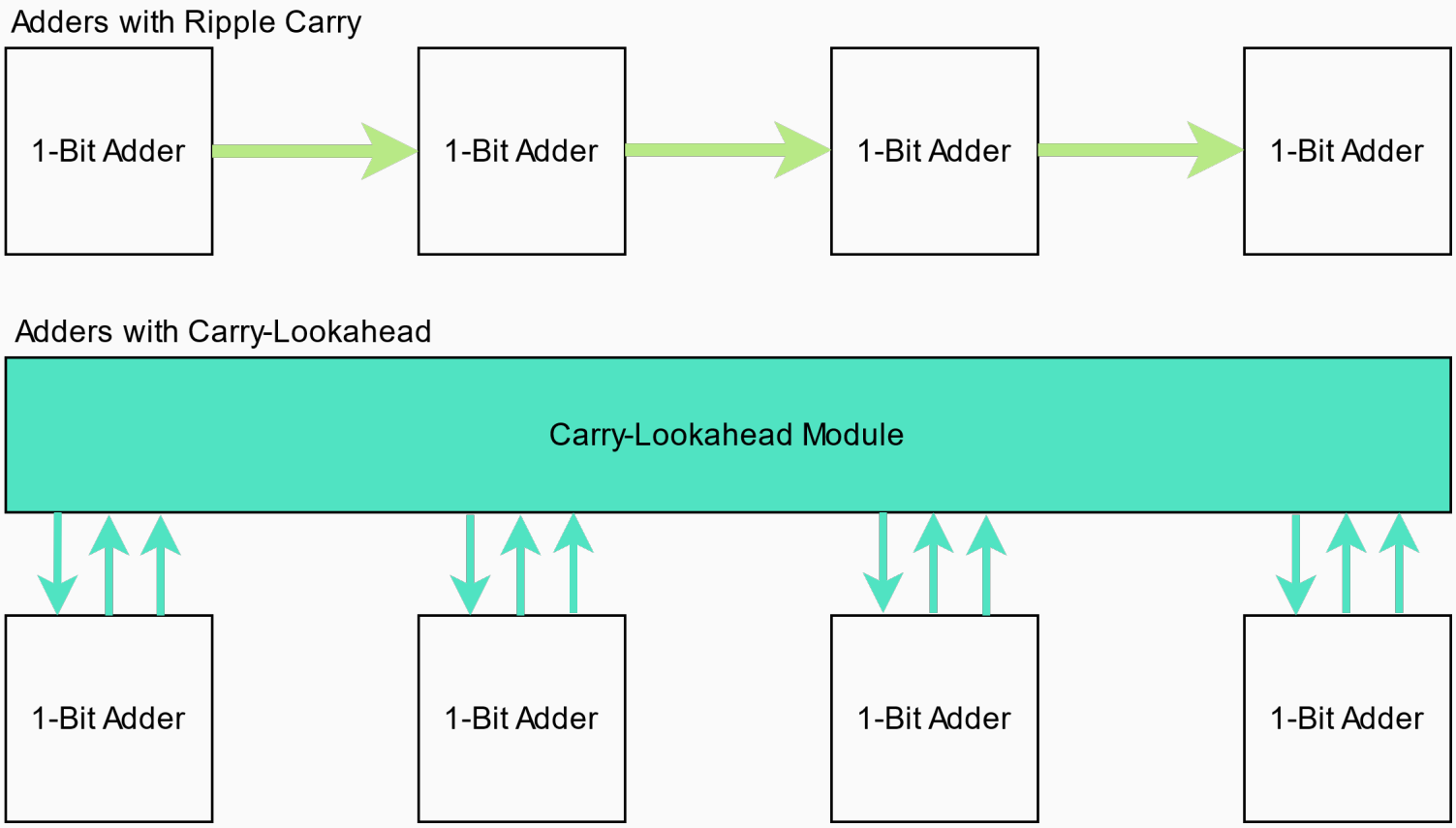


Extending Design Space



Our Approach

- Why not isolate the carry implementation as a feature?
- Excellent candidate for aspect oriented programming.
- Aspects capture implementation information and where it should exist.

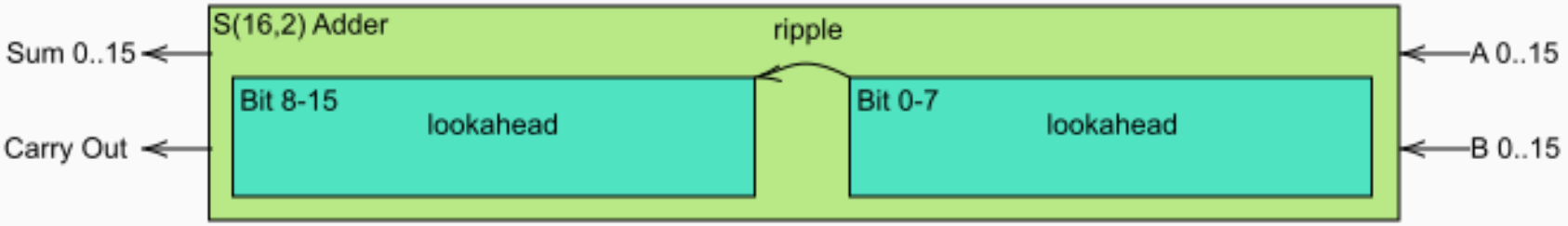


Contribution I

- We use AOP to implement primary functionality of the carry design.
- Implemented in Chisel using their AOP library.
- Nothing is hardcoded, thus refactoring is easy.

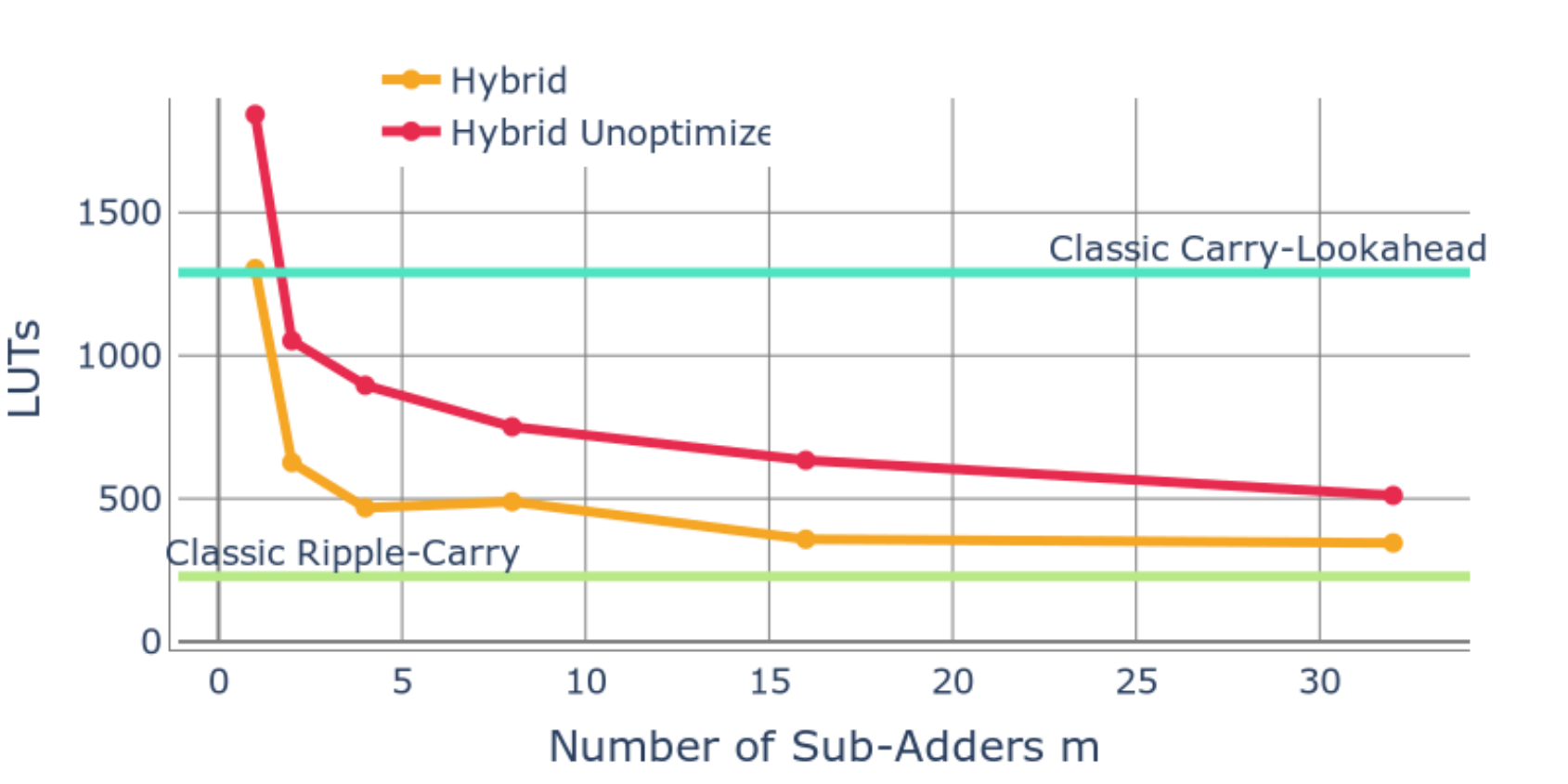
```
InjectingAspect(
  {top: Adder => top.adders},
  {adder: OneBitAdder with CLIO =>
    val g = adder.a & adder.b
    adder.pOut := adder.p
    adder.gOut := g
  }
),
InjectingAspect(
  {top: Adder => Seq(top)},
  {adder: Adder =>
    val cLModule = Module(new cLGenerator(bitWidth))

    for(i <- 0 until bitWidth){
      cLModule.pIn(i) := adder.adders(i).pOut
      cLdModule.gIn(i) := adder.adders(i).gOut
    }

    for(i <- 1 until bitWidth){
        adder.adders(i).carryIn := cLModule.cOut(i-1)
    }

    adder.sums.last := cLdModule.cOut.last
  }
)
```

Contribution II

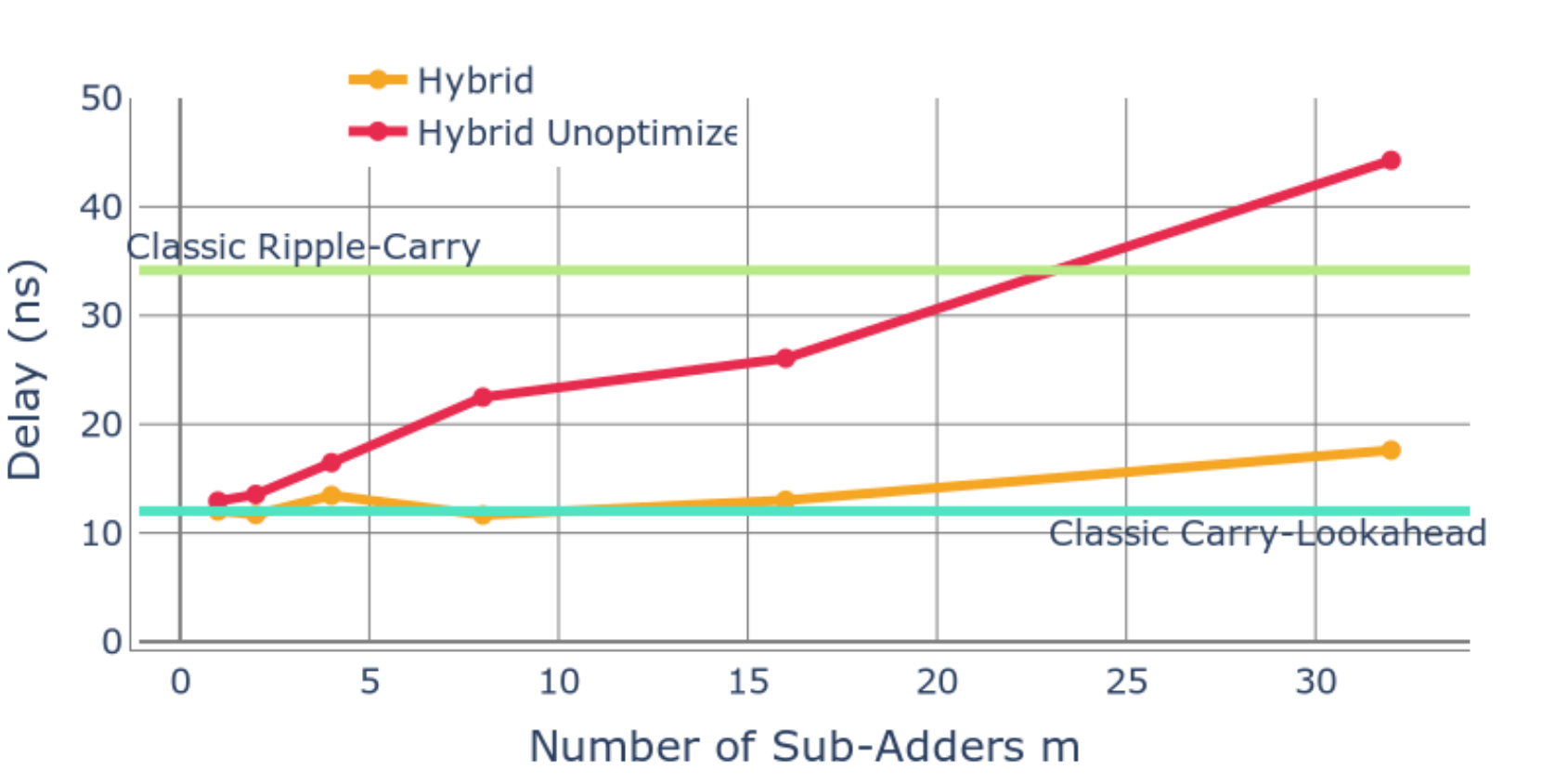


- Built hybrid designs out of our building blocks.
- Two sets of one-bit adders with carry-lookahead applied.
- Each subset can be treated like building blocks.
- Apply ripple-carry between both of the subadders.
- Our approach gets us to the intermediate design space!

Hybrid Design Area



Hybrid Design Delay



Continuing Work

J. Deters, R. Cytron, "Performance Counter Design Variation in Rocket Chip via Feature-Oriented Programming", Fifth Workshop on Computer Architecture Research with RISC-V (CARRV), 2021.

- Chose what events are provides and when events are counted.
- Directly manipulate ASTs of Scala to apply features.
- Provide a aspect-oriented DSL to capture features.