EE290C CORDIC Lab 2

Vighnesh Iyer

October 3, 2018

1 Type Generic CORDIC

I genericified my CORDIC and found that at least **27 stages** were required for the `DspReal` version to achieve 2 LSB precision.

I think my usage of typeclasses was excessive, but I just kept adding what was needed to get the code to compile. Is there a way to alias a conjunction of typeclasses like

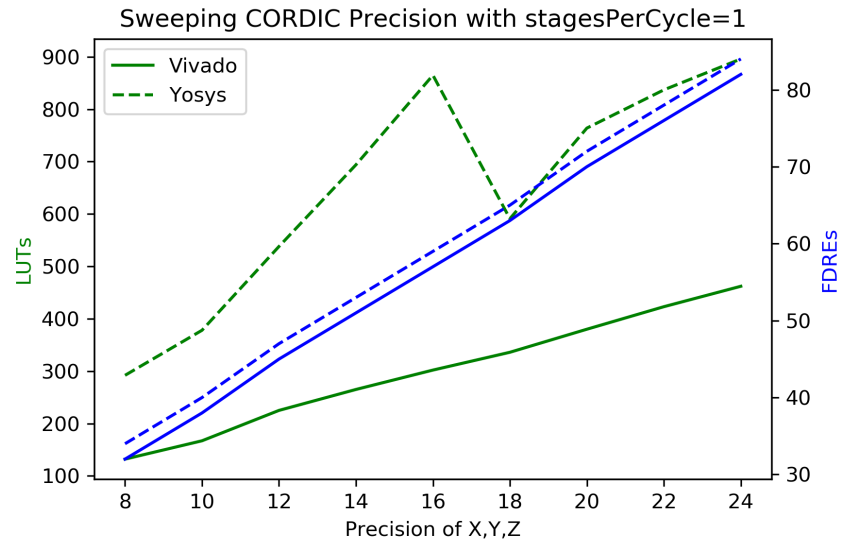
```
typeclass CORDICNumber = Data : Ring : BinaryRepresentation : ConvertableTo : Order
```

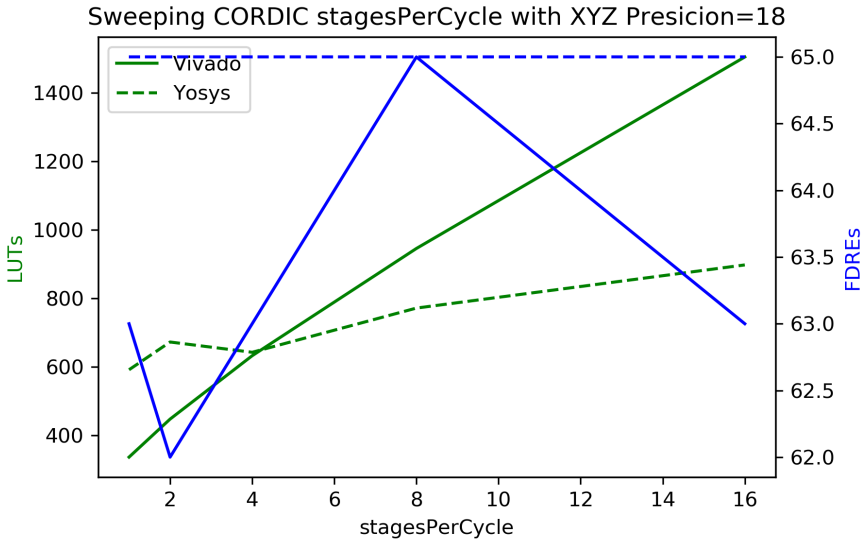
2 Synthesis

I modified the `runhammer.sh` script to sweep precision from 8 to 24 in steps of 2 with fixed `stagesPerCycle` of 1 and to sweep `stagesPerCycle` within [1, 2, 4, 8, 16] for fixed precision of 18 (which yields 16 stages). `runhammer.sh` was also modified to run a local Vivado Webpack edition (so I'm synthesizing for a Kintex Ultrascale+ part).

The number of DSPs was always zero, so I'm going to compare the LUT and FDRE utilization versus the module parameters.

I decided to try out `yosys` Xilinx synthesis just for fun.





3 Rocket-Chip Integration

I filled in the template provided to integrate the CORDIC block as a diplomatic TL module. I tested the rotation mode across the same 16 angles I tested in Scala, and dumped the output to `fa18-by/verisim/vectoring.out`. A reference output was generated by the `CORDIC.ipynb` notebook and is in `fa18-by/verisim/vectoring.gold`. Running `diff --color vectoring.out vectoring.gold` shows the RTL simulation matches the reference within 0-2 LSBs.