Compact CORDIC accelerator implementation for embedded RISC-V core

Alexey Shchekin, Ettore Giliberti

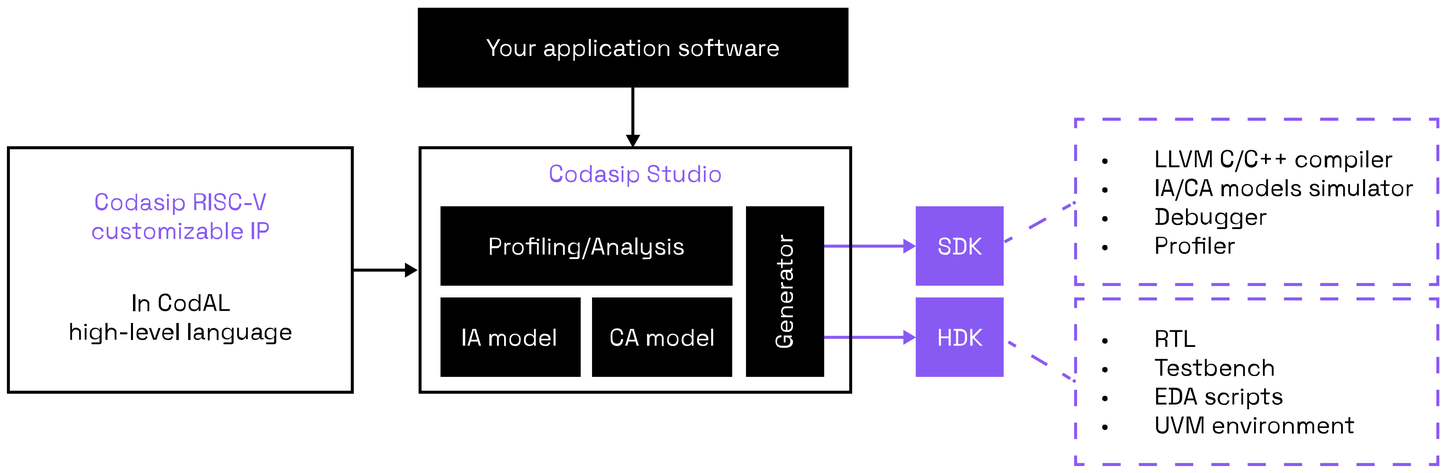
→ Codasip Studio and RISC-V Processors

Begin with a standard core

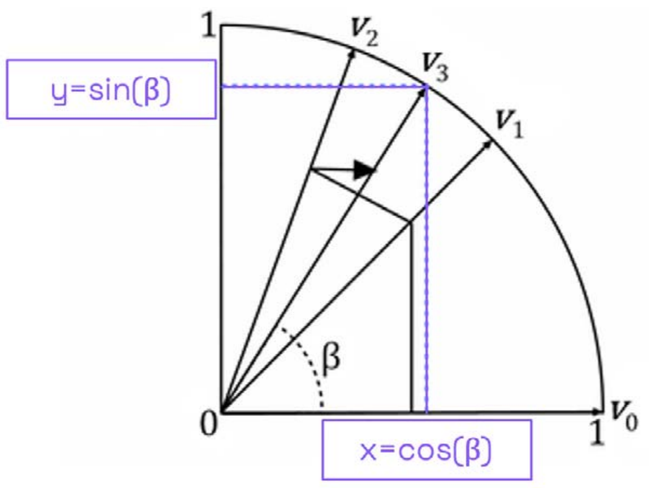
- Embedded and application cores
- High quality, production-ready
- Fully RISC-V compliant

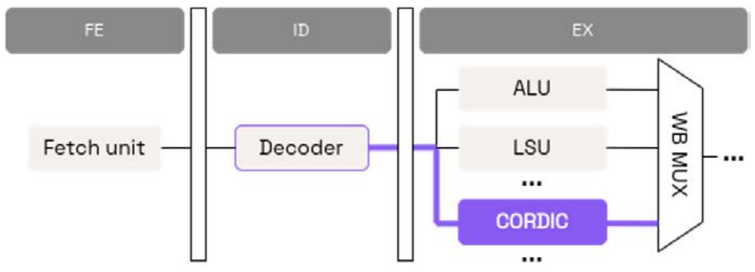
Differentiate with Codasip Studio

- Configure / Customize
- Using CodAL architecture description language



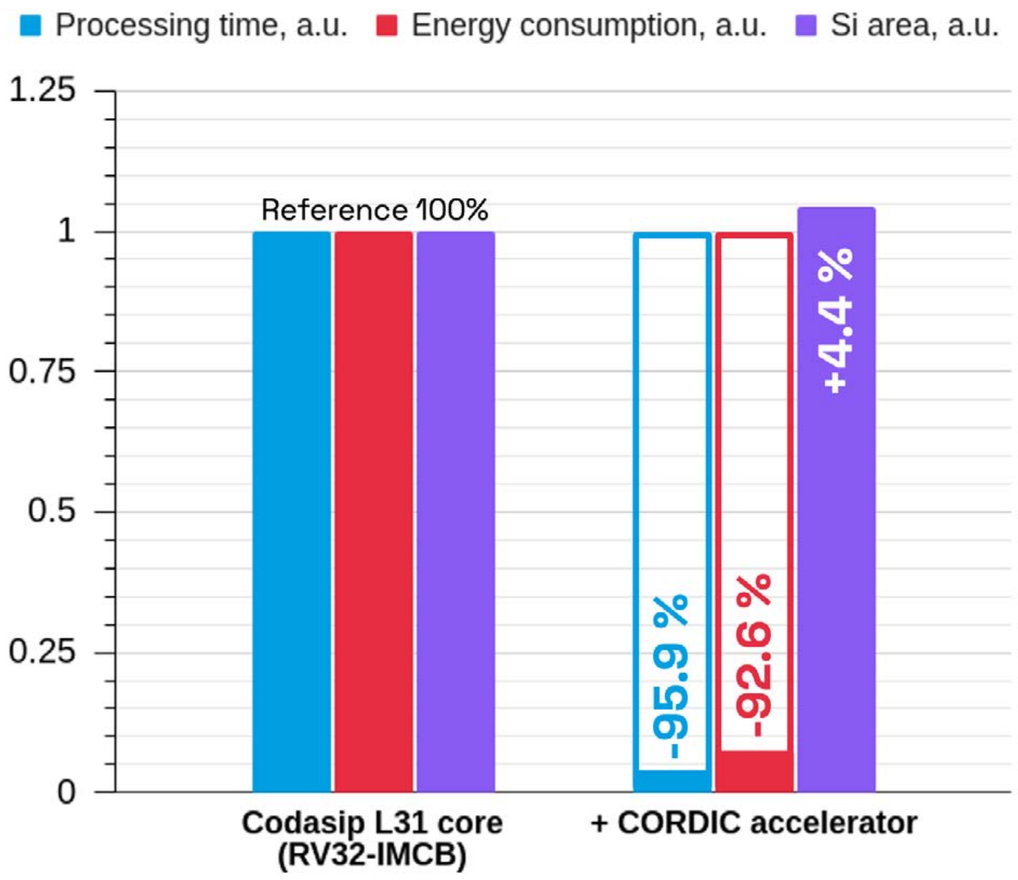
→ CORDIC accelerator implemented with CodAL






```
element i_cordic {
  assembly { "cordic" dst "," src };
  binary { 0:bit[12] src opc dst OPC_CORDIC};
  semantics {
    angle = rf_gpr_read(src);                    //Read the input angle from src
    cos = CORDIC_GAIN;                           //Set the initial cos,sin values
    sin = 0;
    for (shift=0; shift < ITERATIONS; shift++) {  // 16 iterations
      if (angle<0) {                             //Rotate CW if angle is negative
        cos += sin >>> shift;
        sin -= cos >>> shift;                    //Next cos,sin values
        angle += tan[shift];
      } else {                                   //Rotate CCW if angle is positive
        cos -= sin >>> shift;
        sin += cos >>> shift;                    //Next cos,sin values
        angle -= tan[shift];   }
    }
    rf_gpr_write(dst, (cos :: sin));             //Write the results to dst
    codasip_inc_clock_cycle(16);                 //Set the instruction latency
  };
};
```

→ How customization affects the PPA



1 custom instruction call to start "CORDIC" flow

16 cycles to get the result

16-bit fixed-point results representation

(TSMC 28nm)	RISC-V(L31)	+ CORDIC
Area, a.u.	100%	104.4%
Performance gain	1x	24.3x
Energy consumption	100%	7.4%

Design time, lines of code in CodAL		Lines of code in Verilog
3 person-days	210	600 (~3x)