naps

Challenge:

- Build systems with combining FPGAs & CPUs (e.g. Zynq, laptop + ECP5, …)
- Interact & inspect “running” FPGA designs

naps approach:

- Same host language for CPU code and HDL
  - Uses Python based Amaranth HDL
  - Combine HDL and CPU code in the same class hierarchy
- Combine bitstream and code into “fatbitstream”
- Interactive Python shell for live interaction & introspection of the design

naps **SoC code example**

```
class ClockMeter(Elaboratable):
    def __init__(self):
        self.counter = StatusSignal(64)

    def elaborate(self, platform):
        m = Module()
        m.d.sync += self.counter.eq(self.counter + 1)
        return m

    @driver_property
    def mhz(self, t=0.1):
        from time import sleep
        initial_counter = self.counter
        sleep(t)
        return (self.counter - initial_counter) * (1 / t) / 1e6
```

Implemented in FPGA fabric

Running on CPU

```
Python 3.10.10 (main, Apr  5 2023, 14:58:08) [Clang 11.1.0 ] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> dut.clock_meter.mhz
100.0012345
```