Mixed Signal Simulation (by Manuel Moser)

Two ways: True analog simulation (Xyce or ngspice), or the faster mixed-signal simulation where standardcells are converted to a digital delay-model (`xspice`).

Method A: True Mixed-Signal Simulation with xspice using ngspice

Can be done pre-layout and is faster than true-analog post-layout simulation. The standard cells from the `SPICE` file are replaced with timing data (recommendation `-io_time=500p -time=50p -idelay=5p -odelay=50p -cload=250f`).

"Quick" start

Set up an OpenLane project with the RTL `.v` and run `flow.tcl -design <yourdesign> -tag foo -overwrite`. If the flow completes without errors, process `Option A spice-conversion`, else `Option B verilog-conversion` can be used.

- `Option A spice-conversion`: Use `/openlane/designs/<yourdesign>/runs/<run>/results/signoff/<yourdesign>.gds.spice` for the following steps
- Rename the file to `<yourdesign>.spice`
- Run `python3 spi2xspice.py $PDKPATH/libs.ref/sky130_fd_sc_hd/lib/sky130_fd_sc_hd__tt_025C_1v80.lib -io_time=500p -time=50p -idelay=5p -odelay=50p -cload=250f <yourdesign>.spice <yourdesign>.xspice`
- Modify the resulting file `<yourdesign>.xspice`
  - Search for `conb` instances, these instances tie a signal to constant `HIGH` or `LOW`
    - Example:
      `Aadc_core_digital_287 [] [adc_core_digital_287/HI pmatrix_c0_out_n] d_genlut_sky130_fd_sc_hd__conb_1`
      `Aadc_core_digital_288 [] [nmatrix_c0_out_n adc_core_digital_288/LO] d_genlut_sky130_fd_sc_hd__conb_1`
  - replace them with `done` (digital one) and `dzero` (digital zero) instances
    - Example:
      `Aadc_core_digital_287 pmatrix_c0_out_n dzero`
      `Aadc_core_digital_288 nmatrix_c0_out_n done`
  - Check `toana_1v8` and `todig_1v8` instances, sometimes they are not correct

        - Example:
          `AD2A20 [col_out_14_] [a_col_out_14_] toana_1v8`
          `AA2D4 [a_col_out_15_] [col_out_15_] todig_1v8`
          Here you see an output signal has been confused `a_col_out_15_`, it is falsely defined as input `todig_1v8`. Change to
          `AD2D4x [col_out_15_] [a_col_out_15_] toana_1v8`
- Check if the port-order in the spice-subcircuit matches the port order in the simulation. Spice subcircuits are port-order sensitive
- `Option B verilog-conversion`: Use `/openlane/designs/<yourdesign>/runs/<run>/results/final/synthesis/<yourdesign>.v` for the following steps
- Run `vlog2Verilog <yourdesign>.v -o <yourdesign>.vp -l $PDKPATH/libs.ref/sky130_fd_sc_hd/lef/sky130_fd_sc_hd.lef -v "VPWR,VPB" -g "VGND,VNB"`
- Modify the resulting file `<yourdesign>.vp`
    - Search for `conb` instances
    - `conb` has port `.HI()` and `.LO()`, add the missing ports
    - Otherwise there will be an error message `Port missing`
- Run `vlog2Spice <yourdesign>.vp -l $PDKPATH/libs.ref/sky130_fd_sc_hd/spice/sky130_fd_sc_hd.spice -o <yourdesign>.spice`
- Run `python3 spi2xspice.py $PDKPATH/libs.ref/sky130_fd_sc_hd/lib/sky130_fd_sc_hd__tt_025C_1v80.lib -io_time=500p -time=50p -idelay=5p -odelay=50p -cload=250f <yourdesign>.spice <yourdesign>.xspice`
- Modify the resulting file `<yourdesign>.xspice`
    - Search for `conb` instances
        - Example:
          `A_799_ [] [c0n_out_n _noconnect_1_] d_genlut_sky130_fd_sc_hd__conb_1`
          `A_800_ [] [_noconnect_2_ c0p_out_n] d_genlut_sky130_fd_sc_hd__conb_1`
    - replace them with `done` (digital one) and `dzero` (digital zero) instances
        - Example:
          `A_799_ c0n_out_n done`
          `A_800_ c0p_out_n dzero`
    - Check `toana_1v8` and `todig_1v8` instances, sometimes they are not correct
        - Example:
          `AD2A20 [col_out_14_] [a_col_out_14_] toana_1v8`
          `AA2D4 [a_col_out_15_] [col_out_15_] todig_1v8`
          Here you see an output signal has been confused `a_col_out_15_`, it is falsely defined

as input `todig_1v8`. Change to
`AD2D4x [col_out_15_] [a_col_out_15_] toana_1v8`

- Check if the port-order in the spice-subcircuit matches the port order in the simulation. Spice subcircuits are port-order sensitive

Detailed description

There are several different steps to perform depending on which information your files include.

If you have raw verilog-code then you have to first synthesize your file, see `Synthesize Verilog files`.
Raw verilog looks something like:

```
  assign en_pupd = enable & (~(sign^data));
  assign en_vref = enable & (sign^data);
```

If your verilog-file is already synthesized and contains only standard-cells but without powered pins, then you start with `vlog2verilog`. Standardcells in a verilog-file without power pins look like:

```
    sky130_fd_sc_hd__inv_2 _192_ (
    .A(\counter_r[3] ),
    .Y(_044_)
  );
  sky130_fd_sc_hd__or2_2 _193_ (
    .A(_044_),
    .B(_042_),
    .X(_045_)
  );
```

If your verilog-file is already synthesized and contains **powered** standard-cells, then start with `vlog2Spice`. Mind the `.VPWR` `.VGND` `.VNB` `.VPB` ports.

```
sky130_fd_sc_hd__dlymetal6s2s_1 fanout38 (.A(net1),
    .VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR),
    .VPWR(VPWR),
    .X(net38));
sky130_fd_sc_hd__clkbuf_16 clkbuf_0_clk (.A(clk),
    .VGND(VGND),
    .VNB(VGND),
    .VPB(VPWR),
    .VPWR(VPWR),
    .X(clknet_0_clk));
```

If you already have a `.spice` file with sky130-standardcells, then you can skip `vlog2verilog` and `vlog2Spice`. Directly go to `spi2xspice.py`.

```
X_170_ _062_ \current_dac_bit_r[1]\ _128_ _069_ VGND VGND VPWR VPWR _028_
sky130_fd_sc_hd__o211a_1
X_171_ compare_end_w _068_ VGND VGND VPWR VPWR _129_ sky130_fd_sc_hd__nor2_1
```

Synthesize Verilog files with Openlane (or Yosys)

Convert a RTL verilog-file to a gate-level verilog-file with Standard-cells. In OpenLane, run:

- `flow.tcl -design <name> -tag foo -overwrite -interactive`
  - `package require openlane`
  - `run_synthesis`
  - or just run through the regular flow without `-interactive`

Add Power-Pins

The standard cell models must include power pins. Use `vlog2Verilog` the verilog-file does not contain powered standardcells.

- `vlog2Verilog foo.v -o foo.vp -l $PDKPATH/libs.ref/sky130_fd_sc_hd/lef/sky130_fd_sc_hd.lef -v "VPWR,VPB" -g "VGND,VNB"`

Convert to SPICE

The verilog-file needs to be converted to a `.spice` netlist. Use the binary `vlog2Spice` from qflow.

- `vlog2Spice foo.vp -l $PDKPATH/libs.ref/sky130_fd_sc_hd/spice/sky130_fd_sc_hd.spice -o foo.spice`

Convert to Xspice

Use the script `spi2xspice.py` from qflow to convert a spice-netlist to xspice. The script automatically replaces the standard-cells with digital models.

- `python3 spi2xspice.py $PDKPATH/libs.ref/sky130_fd_sc_hd/lib/sky130_fd_sc_hd__tt_025C_1v80.lib -io_time=500p -time=50p -idelay=5p -odelay=50p -cload=250f foo.spice foo.xspice`

> Note: If `conb` standard cells are present, then they need to be replaced with digital pullups `<instance_name> <net_out> done` and pulldowns `<instance_name> <net_out> dzero`. Reason: Missing data for `conb`, simulation will fail.

> Warning: if the verilog code uses bus lines as `net[25:0]`, then the gds-generated xschem file might have pin orders alphabetically like `net0..net10:net19..net1..net20:net25..net2` and they need to be corrected manually. I generally recommend to double-check the pin-order of the generated spice-netlist of your testbench.

> Input signals should not have a slow rise/fall time. If your XSPICE-outputs change to $V_{DD}/2$, then check your input signals, maybe there is a clock transition while an input-signal is not clearly high or low.

Generate xschem-symbol

Create a symbol for the xspice subcircuit with the same input/output/inout ports. The symbol-filename must have the same name as the xspice-subcircuit. With `q` set the type from `subcircuit` to `primitive`. With `Shift`+`a` you can preset a pin-order.

Testbench

`.include` the generated `.xspice` file

True analog simulation

This is the post-layout-simulation

- Generate the digital circuit with OpenLane.
- Extract the SPICE file from the resulting GDS with magic
  - Open the GDS file in `../runs/final/gds/` with `magic <file>.gds`
  - In the tcl console: `extract`, `ext2spice lvs`, `ext2spice`, maybe with `ext2spice cthresh 0`
  - Generate a symbol for xschem with the same pin order as in the `.spice`-file, set pin-numbers with Shift-s in the symbol. With `q`, set the type to `primitive`.
  - In the testbench, include the generated `.spice` file (absolute-path, otherwise ngspice won't find the file)

Bussing

For bussing the syntax D[1..3] is one of the possible syntaxes, and it expands to D1,D2,D3, However if you use D[1:3] it expands to D[1],D[2],D[3]. If you are also using XSPICE primitives, in this case you can add a netlist_options.sym component to instruct xschem to replace [and] with two different characters