Model conversion to ngspice in open_pdks

Tim Edwards

SVP Analog & Platform

efabless

efabless.com

Open Circuit Design

opencircuitdesign.com

Model conversion to ngspice in open_pdks

Origin of the idea:

Thomas Benz, ETHZ

https://codeberg.org/thommythomaso/spectre2spice

Model conversion to ngspice in open_pdks

Main practical implementation:

Generation of sky130 open PDK

Sources in Spectre format only

General open PDK requirement of ngspice compatibility

https://github.com/google/skywater-pdk-libs-sky130_fd_pr

Model conversion to ngspice in open_pdks

Refined implementation:

open_pdks

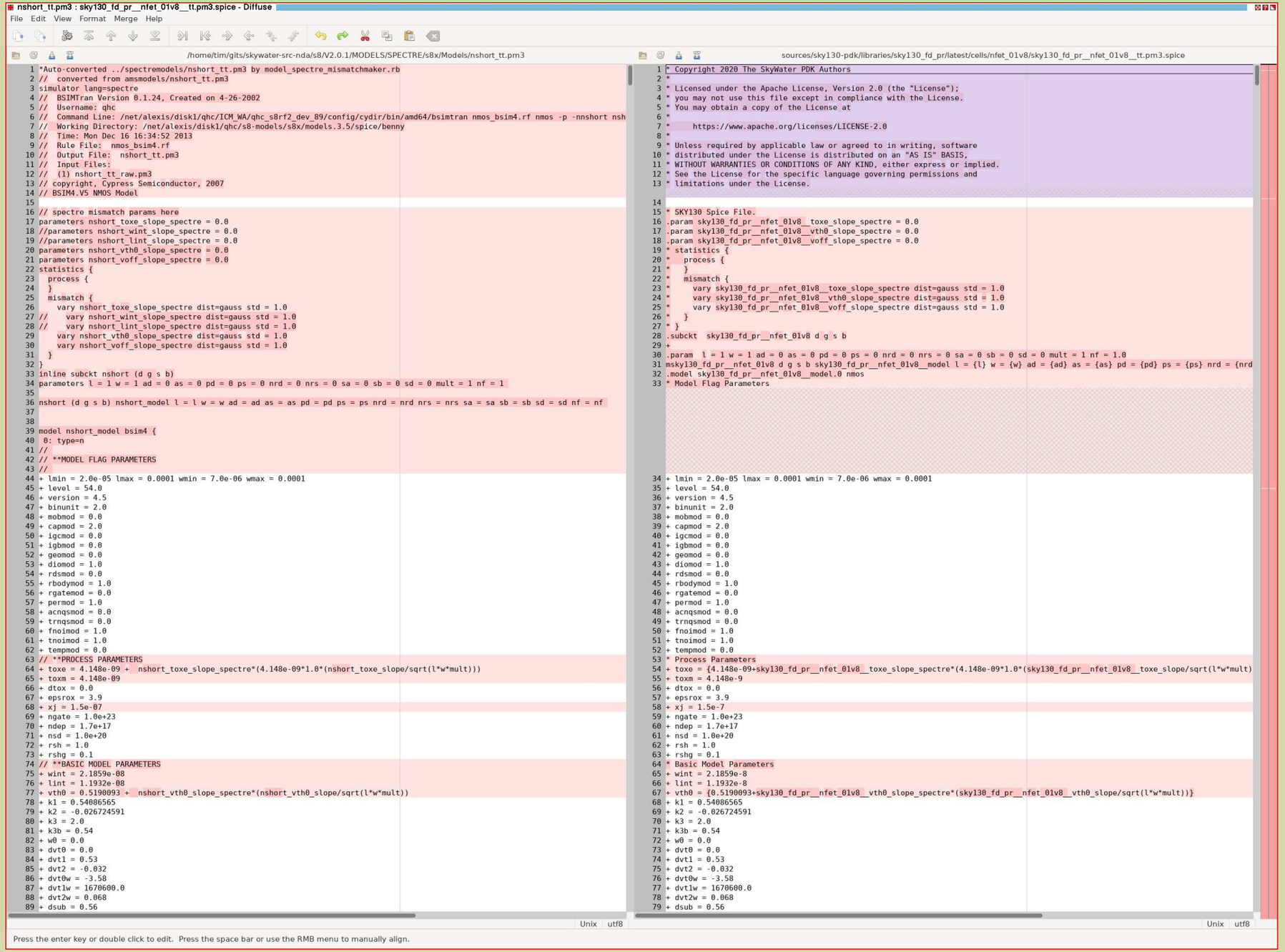
Addresses some issues not managed in time for the open PDK

Mainly monte carlo modeling

`https://github.com/RTimothyEdwards/open_pdks`

Stage 1:

Convert Spectre to ngspice



```
simulator lang=spectre
        .
        .
        .
simulator lang=spice
```

```
// **PROCESS PARAMETERS
```

```
* Process Parameters
```

```
inline subckt nshort (d g s b)
parameters l = 1 w = 1 ad = 0 as = 0 pd = 0 …
nshort (d g s b) nshort_model l = l w = w ad = ad …

model nshort_model bsim4 {
 0: type=n
```

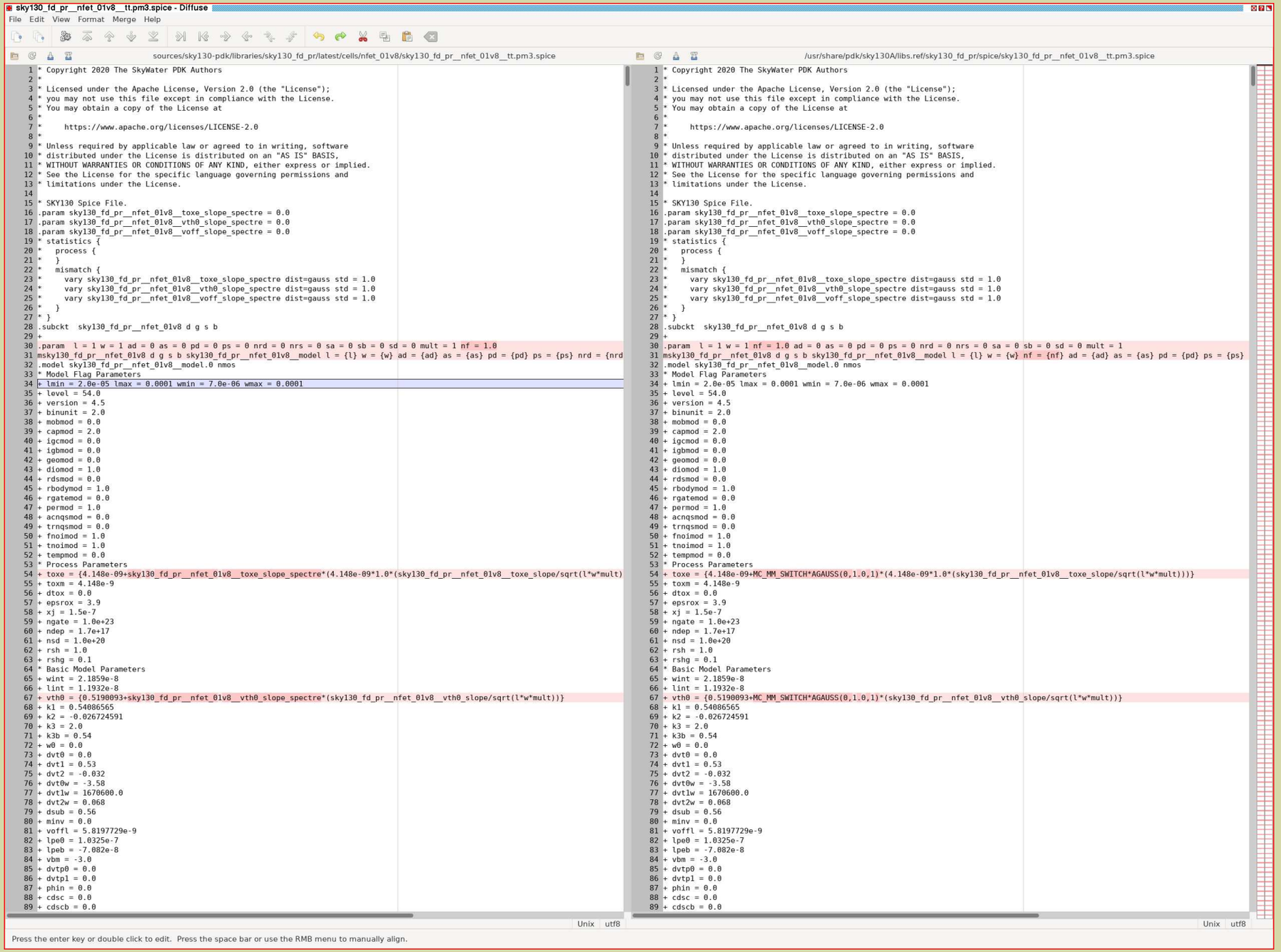
```
.subckt sky130_fd_pr__nfet_01v8 d g s b
.param l = 1 w = 1 ad = 0 pd = 0 ps = 0 …

msky130_fd_pr__nfet_01v8 d g s b sky130_fd_pf__nfet_01v8__model
+ l = {l} w = {w} …
.model sky130_fd_pr__nfet_01v8__model.0 nmos
```

```
subckt xuhrpoly_base (r1 r2 b)
   parameters
      + w l mult
      // parameter values:
      + rsheet  = 2000
      …

   rbody (ra r2) resbody r=rbody*(1 + abs(v(r1,r2))*…
   rend (r1 ra) resistor r=rend*(1 + vc1_raw_end*…
   c1 (r2 b) capacitor c = ((l+2*2.08)*w*crpf_precision*…
   c2 (r1 b) capacitor c = ((l+2*2.08)*w*crpf_precision*…
   model resbody resistor tc1=-1.47e-3  tc2=2.7e-6  tnom=30
ends xurpoly_base
```

```
.subckt sky130_fd_pr__res_xhigh_po__base r1 r2 b
+ w=1 l=1 mult=1
.param  rsheet = 2000 …

rbody ra r2 r = {rbody*(1+vc1_raw_end*…
+ tc1 = -1.47e-3  tc2 = 2.7e-6  tnom = 30
rend r1 ra  r = {rend*(1+vc1_raw-end*…
c1 r2 b  c = {((l+2*2.08)*w*crpf_precision*…
c2 r1 b  c = {((l+2*2.08)*w*crpf_precision*…
.ends sky130_fd_pr__res_xhigh_po__base
```

Stage 2:

Handle Monte-Carlo parameterization



```
* statistics {
*   mismatch {
*      vary sky130_fd_pr__nfet_01v8__toxe_slope_spectre dist=gauss std=1.0
*   }
* }
```

```
toxe = {4.148e-09+sky130_fd_pr__nfet_01v8__toxe_slope_spectre*…
```

```
toxe = {4.148e-09+MC_MM_SWITCH*AGAUSS(0,1.0,1)*…
```

```
.param cnwvc = 0.0
.param diff_cd = 0.0
.param hvn_bodyeffect = 0.0
.param hvn_diode = 0.0
…
```

```
.param cnwvc = 0.0 + MC_PR_SWITCH*AGAUSS(0,1.0,1)
.param diff_cd = 0.0 + MC_PR_SWITCH*AGAUSS(0,1.5,1)
.param hvn_bodyeffect = 0.0 + MC_PR_SWITCH*AGAUSS(0,2.8,1)
.param hvn_diode = 0.0 + MC_PR_SWITCH*AGAUSS(0,1.0,1)
…
```

```
*statistics {
*  process {
*    vary cnwvc dist=gauss std=1.0
*    vary diff_cd dist=gauss std=1.5
*    vary hvn_bodyeffect dist=gauss std=2.8
*    vary hvn_diode dist=gauss std=1.0
    …
```

Stage 3:

Add custom .lib file

```
* Typical corner with mismatch (tt_mm)
.lib tt_mm
.param mc_mm_switch=1
.param mc_pr_switch=0
include "corners/tt.spice"
…
.endl tt_mm
```