28nm FDSOI
Digital Design Tutorial

Circuits Multi-Projets

MPW Services Center for IC / MEMS Prototyping

http://cmp.imag.fr

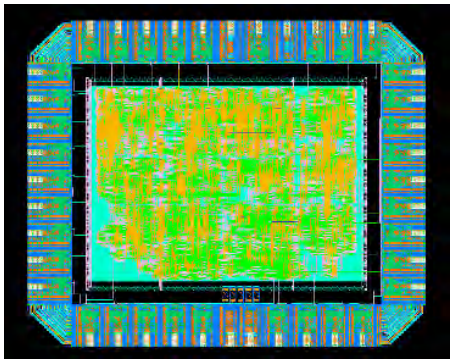
Grenoble - France

Context & Motivation

- ✓ A basic version of this tutorial was developed in 2015 (v1.4).
- ✓ A second version with several updates and additional functionalities was released on summer 2016 (v2.3).
- ✓ Introduce the digital design thanks to a plug and play tutorial:
  - ✓ each step from **RTL to GDSII** is detailed,
  - ✓ based on **standard methodologies and CAD tools**,
  - ✓ all **scripts and testbenches** are provided,
  - ✓ illustrated from a **simple digital circuit** example,
  - ✓ implemented on an **advanced technology**,
  - ✓ integrates **body biasing** functionalities.

Verilog RTL

```
always  @ (negedge clk)
begin
  if (load == 1)
     eq <= 0;
  else
       if (eq1 == 0)
         eq <= 0;
       else
         eq <= 1;
```

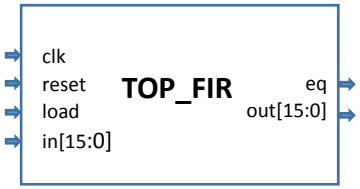


GDSII layout

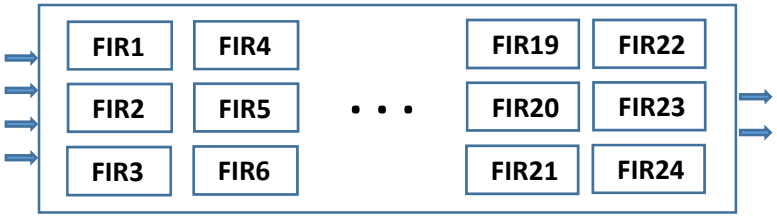
RTL to GDS flow: FIR circuit example

```
module TOP_FIR (
  input clk,
  input reset,
  input load,
  input wire[15:0] in,
  output wire[15:0] out,
  output reg eq);

parameter N = 24;

reg [15:0] in1[N-1:0];
wire [15:0] out1[N-1:0];
wire [N-2:0] eq1;

genvar i;

generate
for (i=0; i<=N-1; i=i+1)
    begin: FIR
        FIR FIR1(
            .clk(clk),
            .reset(reset),
            .load(load),
            .input_sample(in),
            .output_sample(out1[i]) );
    end
endgenerate
```

Basic synchronous and sequential circuit, called “TOP_FIR”

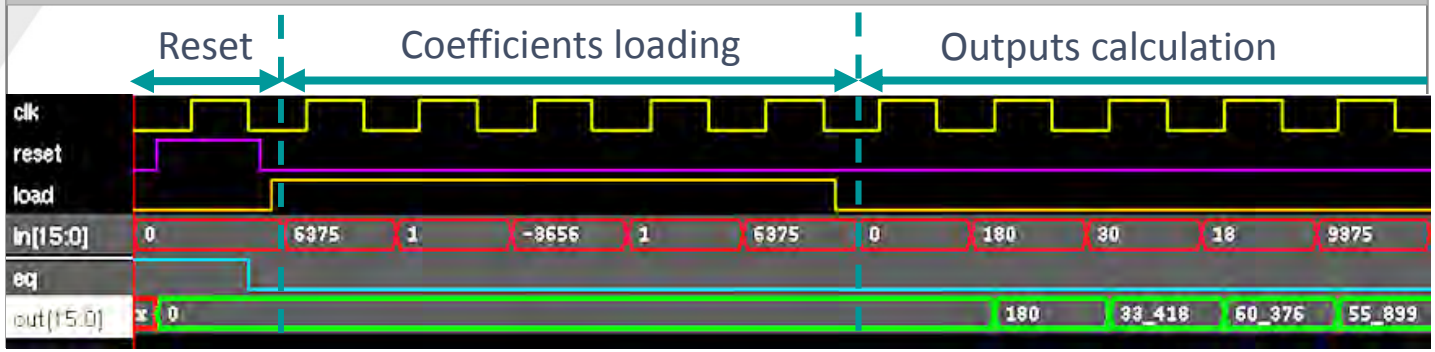


Composed of 24 parallel Finite Impulse Response (FIR) filters



Final circuit:
- 73’000 standard cells
- 0,7mm² area

RTL simulation results:



RTL to GDS flow: logic synthesis

Verilog RTL

```
generate
for (i=0; i<=N-1; i=i+1)
    begin: FIR
        FIR FIR1(
            .clk(clk),
            .reset(reset),
            .load(load),
            .input_sample(in),
            .output_sample(out1[i]) );
    end
endgenerate
```

Synthesis tool

Design Compiler (Synopsys)

Genus (Cadence)

Gate level netlist

```
FIR FIR_0__FIR1(.clk (clk), .reset (reset), .load (load),
     .input_sample (in), .output_sample (out));
FIR_1 FIR_1__FIR1(.clk (clk), .reset (reset), .load (load),
     .input_sample (in), .output_sample ({\out1[1] [15],
     [14], \out1[1] [13], \out1[1] [12], \out1[1] [11],
     [10], \out1[1] [9], \out1[1] [8], \out1[1] [7], \out1
     \out1[1] [5], \out1[1] [4], \out1[1] [3], \out1[1] [2]
     \out1[1] [1], \out1[1] [0]}));
FIR_2 FIR_2__FIR1(.clk (clk), .reset (reset), .load (load),
     .input_sample (in), .output_sample ({\out1[2] [15],
     [14], \out1[2] [13], \out1[2] [12], \out1[2] [11],
     [10], \out1[2] [9], \out1[2] [8], \out1[2] [7], \out1
     \out1[2] [5], \out1[2] [4], \out1[2] [3], \out1[2] [2]
     \out1[2] [1], \out1[2] [0]}));
```

Genus: Next generation of RTL Compiler

Gate level simulation results:

```
////////////////////////////////////
// Loading coefficients...
////////////////////////////////////
time      clk     reset     load      in       out      eq
0         0       0         0         0        x        x
12        0       1         0         0        0        x
24        0       0         1      6375        0        0
36        0       0         1         1        0        0
[...]
////////////////////////////////////////////////
// Coefficients loaded, start processing...
////////////////////////////////////////////////
time      clk     reset     load      in       out      eq
180       0       0         0         0        0        0
192       0       0         0       180      180        0
204       0       0         0        30    33418        0
216       0       0         0        18    60376        0
228       0       0         0      9375    55899        0
[...]
```

Reset, loading, and processing modes

Updates of the tutorial

RTL to GDS flow: place and route

Gate level netlist
+ top cell “PAD_TOP_FIR” (including IOs)

```
module PAD_TOP_FIR ( clk, reset, load, in, out, eq );
  input [15:0] in;
  output [15:0] out;
  input clk, reset, load;
  output eq;
  wire w_clk, w_reset, w_load, w_eq;
  wire [15:0] w_out, w_in;
  wire [16:0] node_float;
  wire netTie1, netTie0;

TOP_FIR  I_TOP_FIR (.clk(w_clk), .reset(w_reset), .load
```

Tech. files issued from ST’s new FoundationTechnoKits

Place & route tool

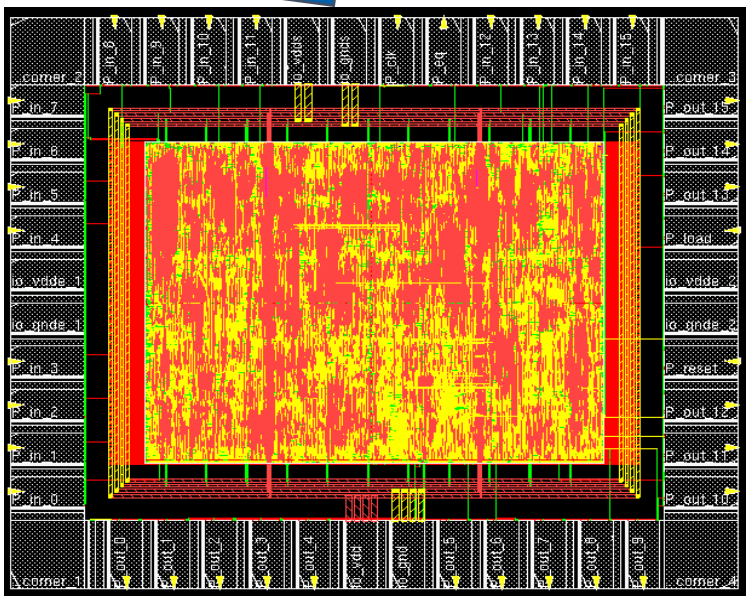
Innovus (Cadence)

Place & route script updated for Innovus 15.20.000

Verilog netlist

```
module PAD_TOP_FIR (
        clk,
        reset,
        load,
        in,
        out,
        eq);
   input clk;
   input reset;
   input load;
   input [15:0] in;
   output [15:0] out;
   output eq;
```

GDSII layout



Back-annotated simulation results:

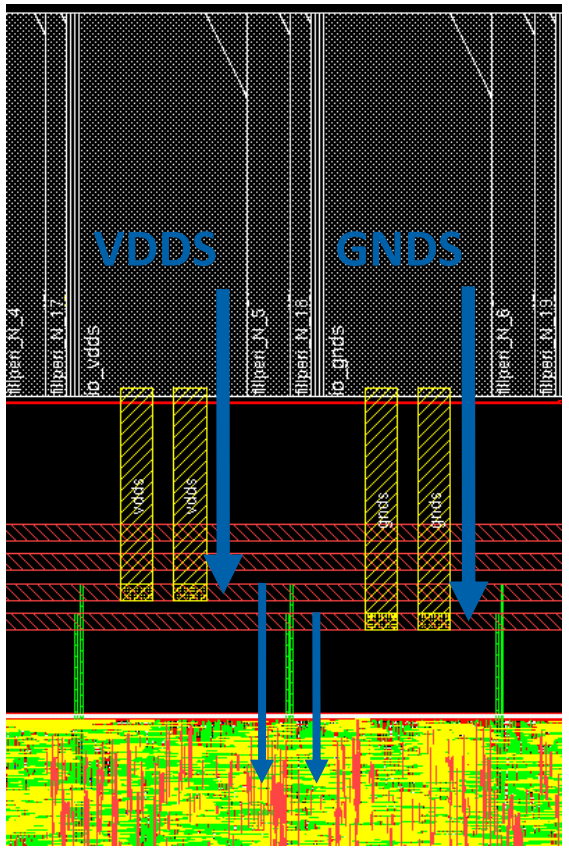
```
Annotating SDF timing data:
        Compiled SDF file:      PAD_TOP_FIR_routed_genus.sdf.X
        Log file:               sdffile_rc.log
        Backannotation scope:   top.U
        Configuration file:
Annotation completed successfully...

////////////////////////////////////
// Loading coefficients...
////////////////////////////////////
time      clk     reset    load     in       out      eq
0         0       0        0          0      x    x
12        0       1        0          0      0    x
24        0       0        1       6375      0    0
36        0       0        1          1      0    0
[...]
////////////////////////////////////////////////
// Coefficients loaded, start processing...
////////////////////////////////////////////////
time      clk     reset    load     in       out      eq
180       0       0        0          0      0    0
192       0       0        0        180    180    0
204       0       0        0         30  33418    0
216       0       0        0         18  60376    0
228       0       0        0       9375  55899    0
[...]
```

Updates of the tutorial

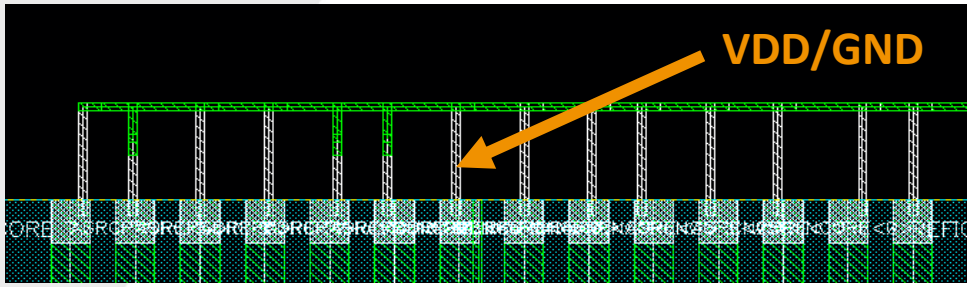
28nm FDSOI features during place and route

IOs placement:

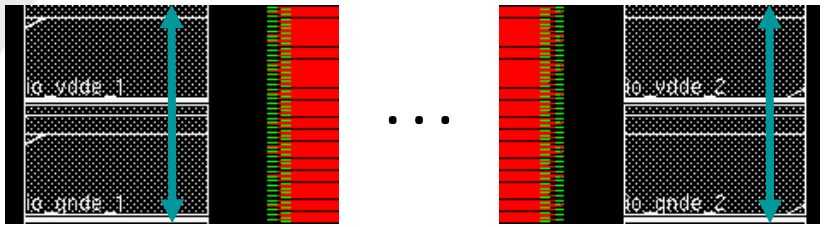
- ✓ Addition of 2 specific pads dedicated to supply VDDS and GNDS body biasing voltages (-1,8V to +1,8V):



- ✓ Addition of an IO filler cell to tie high or low the compensation signals:



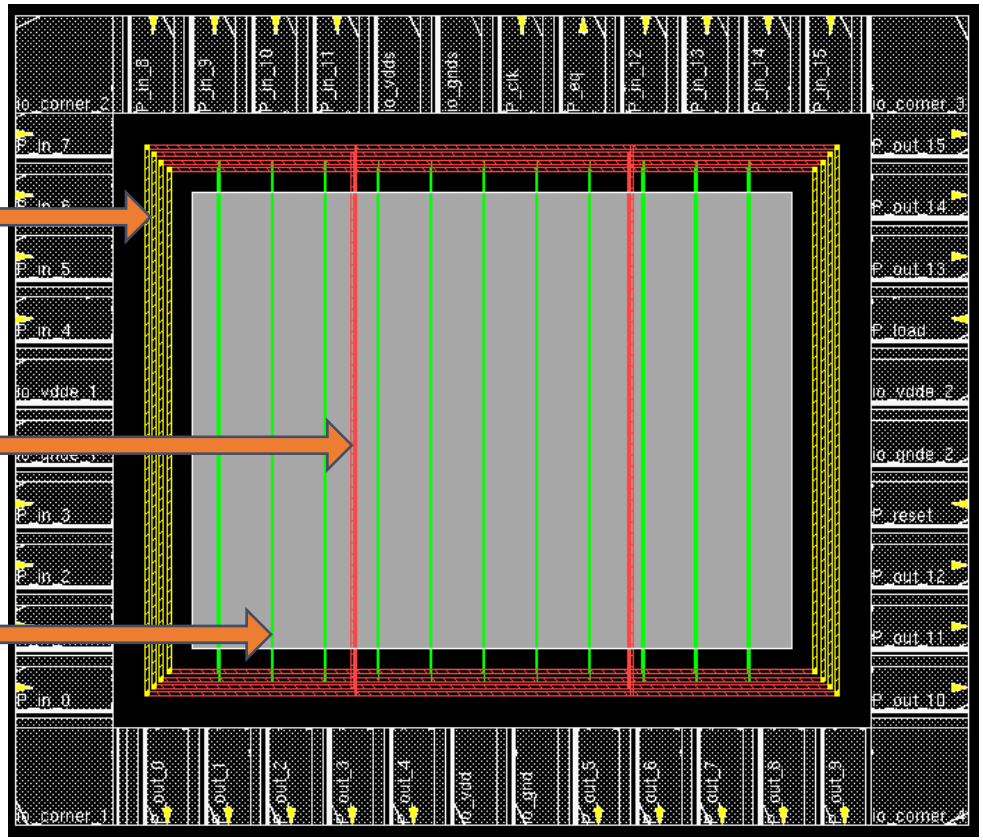
- ✓ 2 supply pairs VDDE and GNDE:



28nm FDSOI features during place and route

Floorplan and power plan generation:

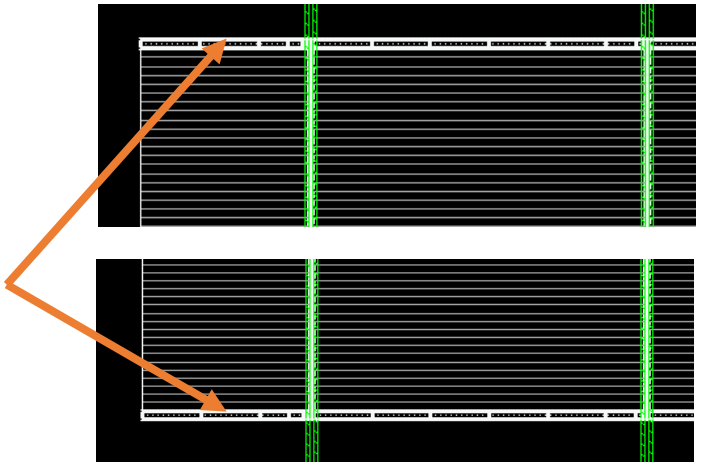
- ✓ 4 power rings (VDD, GND, VDDS, GNDS) drawn with thickest metal layers.
- ✓ 2 pairs of power stripes to feed the circuit with VDD and GND
- ✓ 1 pair of VDDS/GNDS stripes every 50µm spacing to feed body biasing voltages to standard cells



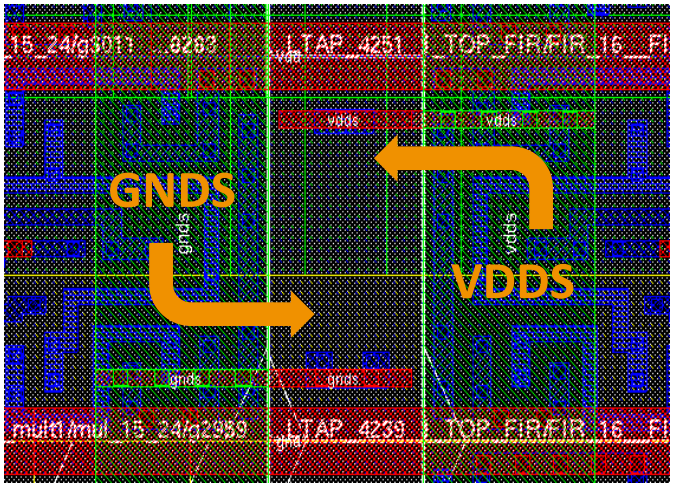
28nm FDSOI features during place and route

Core cells placement and routing:

- ✓ Fillers cells on top and bottom core rows (to meet particular DRC rules)



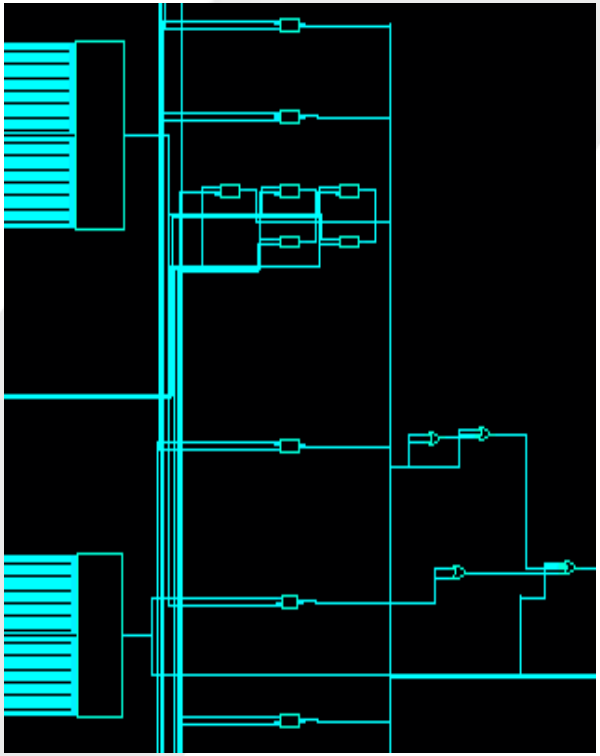
- ✓ Implementation of filler tap cells with separated power and ground rails (dedicated connections to bodies of standard cells)

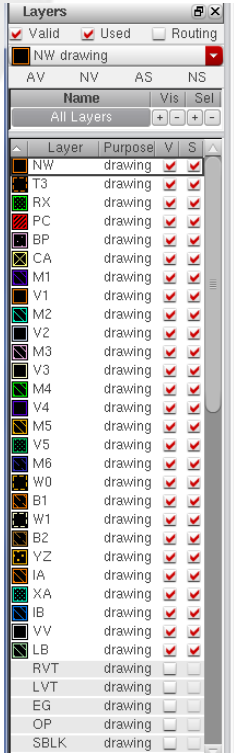


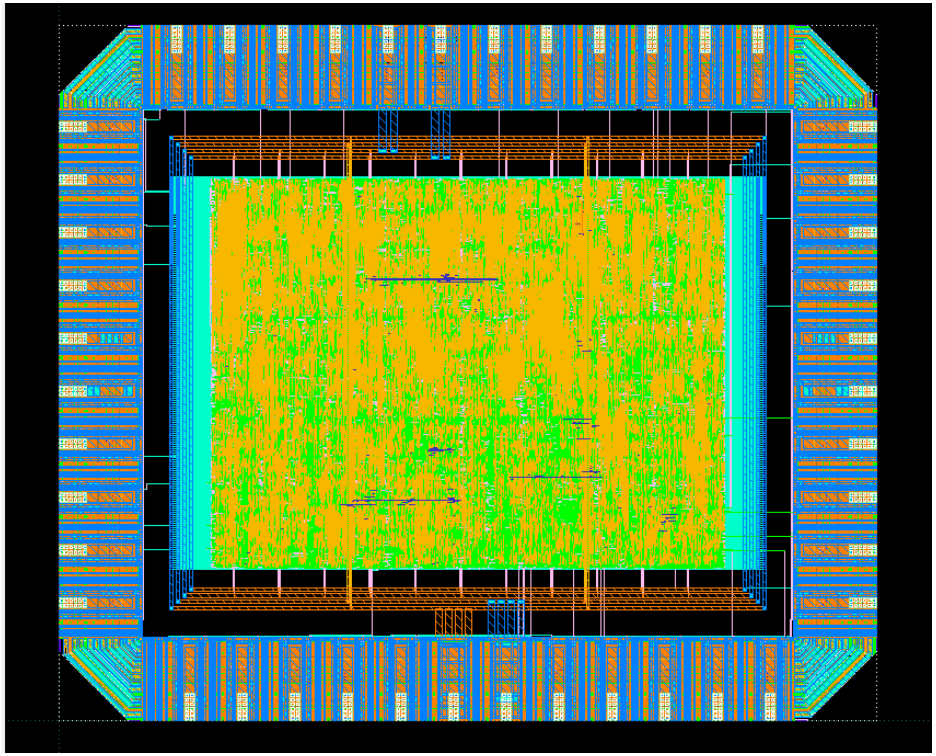
- ✓ Restriction of the tool to use the 8 first metal layers to route signals, and the 2 top layers for power.

Final verifications

- ✓ GDSII and netlist can be imported under Cadence Virtuoso 6.1.6:







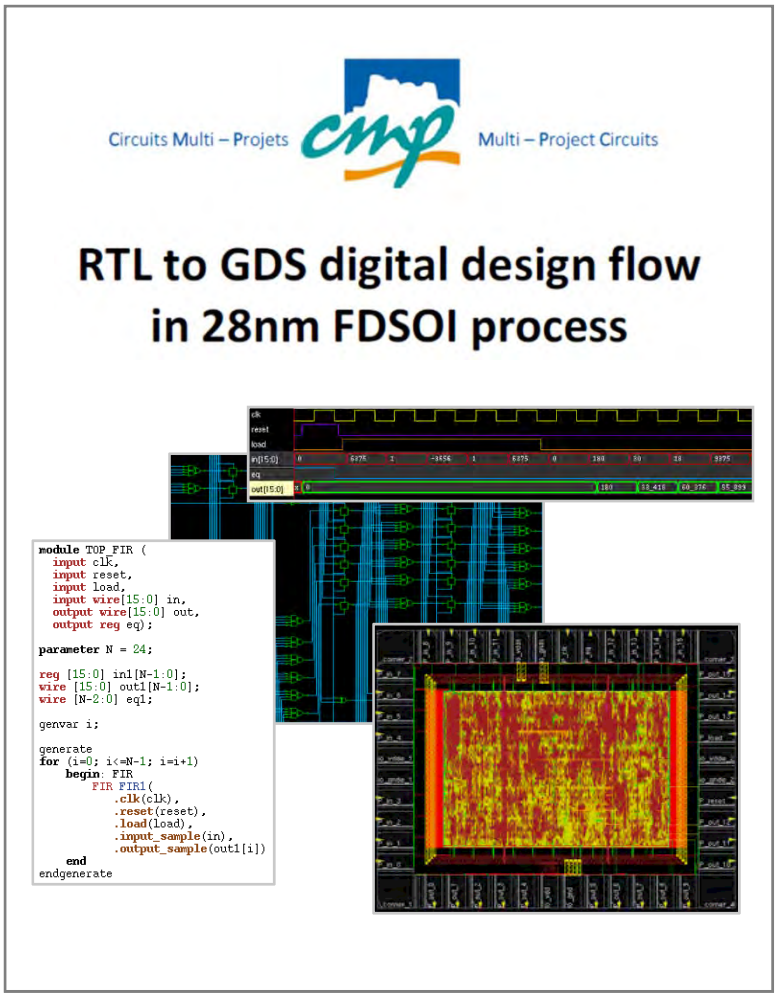
Schematic view

Layout view

- ✓ LVS and DRC verifications can be performed

Tutorial delivery

- ✓ 118 institutions received in 2016 this upgraded version of the tutorial
- ✓ Developed in CMOS28FDSOI technology, with PDK version 2.5.f
  - ✓ can be easily adapted for PDK 2.7.a
- ✓ A third tutorial release is planned Q2 2017 for latest CMOS28FDSOI PDK 2.9
  - ✓ migration from 10ML to **8 metal layers** stack,
  - ✓ **LVS** and **DRC** verifications.
- ✓ Still positive feedback from designers !

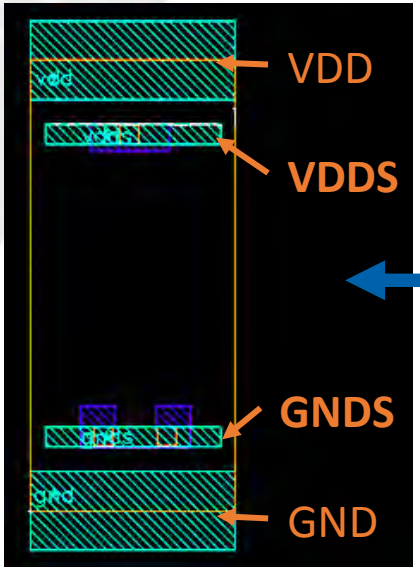


Thank you!

web

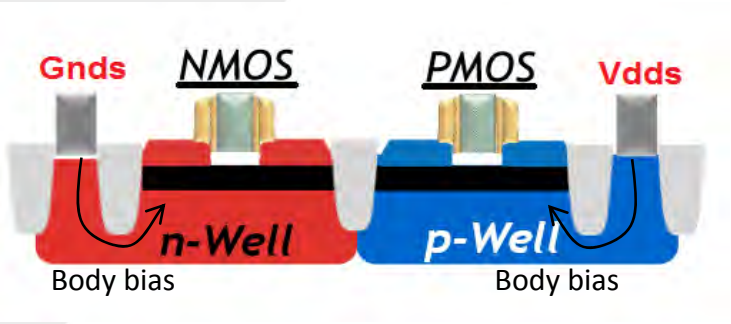
Body biasing methodology flow

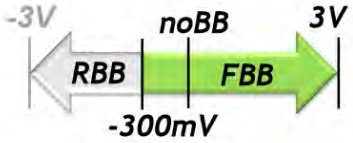
Body biasing in layout view:

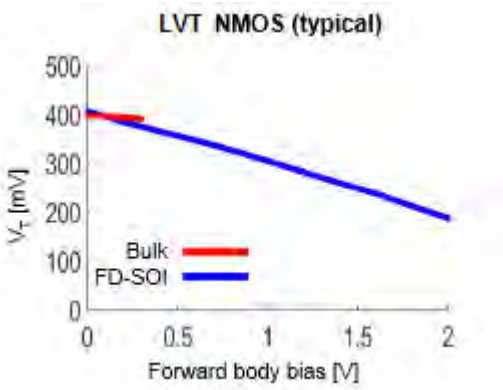


Filler tap cell with **separated power and ground rails**:
VDD/VDDS and GND/GNDS

Body biasing on LVT (flip-well) transistors:







➔ FBB or RBB: speed or leakage optimization

FBB: Forward Body Biasing RBB: Reverse Body Biasing