STMicroelectronics / Crolles site / Device Modelling
850 Rue J.Monnet - 38926 CROLLES CEDEX FRANCE

Aug, 8th 2005

CROLLES2 ALLIANCE
CMOS065

ST MODELS
Mismatch: Corner/Montecarlo simulations

CMOS065 MOS MODELLING TEAM

1. MISMATCH SIMULATIONS

1.1 Introduction

In order to simulate process corners with or without mismatch, or parametrised mismatch corners, a methodolody has been defined. The <family>_dev parameter (svtlp_dev for SVT LP devices for instance) is being used to define the simulation strategy. The pre-defined four strategies are:

- no mismatch: <family>_dev=0
- mismatch using Monte-Carlo simulations: <family>_dev=1 + instance parameter mismatch=1 + Monte-Carlo simulation (.mc in eldo)
- mismatch included in process corners: <family>_dev=2
- parametrised mismatch corners: <family>_dev=3

So as to easily implement the simulation strategy into the netlists, the <family>-dev parameter has been included in a library file (mismatch.lib), within four libraries. The libraries are:

- mismatch_no library: all <family>_dev parameters set to 0, no mismatch used
- mismatch_mc library: all <family>_dev parameters set to 1, mismatch through Monte-Carlo simulation
- mismatch_corner library: all <family>_dev parameters set to 2, global mismatch in corners
- mismatch_param library: all <family>_dev parameters set to 3, parametrised mismatch corners

1.2 Standard deviation calculation

The standard deviation for Vt and mobility mismatch variations are calculated using the following equations:

$$\sigma_{VT} = \frac{A_{VT}}{\sqrt{2 \cdot mult \cdot W \cdot L}} + B_{VT} \text{ and } \sigma_{\beta} = \frac{A_{\beta}}{\sqrt{2 \cdot mult \cdot W \cdot L}} + B_{\beta}$$

where (A_{vt}, B_{vt}) and (A_{β}, B_{β}) are extracted on a pair of transistors by the Electrical Characterization team.

Then, using the above σ values, the total mismatch variations are applied as follows:

$$VT = VT_{w\backslash\mathrm{o\ mismatch}} + \Delta_{VT} \text{ and } mobility = mobility_{w\backslash\mathrm{o\ mismatch}} \cdot (1 + \Delta_{\beta})$$

1.3 Mismatch into corners

1.3.1 Mismatch into corners for foundry alignment and Process Control Monitoring

During Process Control Monitoring, both process (global) and mismatch (local) variations are taken into account. Therefore, process and mismatch variances have to be added.

$$\sigma_{VT}^{total} = \sqrt{(\sigma_{VT}^{process})^2 + (\sigma_{VT}^{mismatch})^2}$$

This mismatch contribution is globally activated using the <family>_dev parameter (<family>_dev=2) . The parameter <family>_dev can be defined in the netlist or via the mismatch.lib file.

See graphic examples in the chapter 3

1.3.2 Parametrized mismatch variations into corners

Some applications (SRAM, DRAM, ...) require to simulate various mismatch states around a pre-defined process corner. Therefore, 2 instance parameters have been defined to specify the number of sigma for the VT mismatch variations (dvt_mdev) and the mobility mismatch variations (dmu_mdev).

$$VT = VT_{w\backslash\text{o mismatch}} + \text{sign*dvt\_mdev*}\sigma_{VT}^{mismatch},$$

and $$mobility = mobility_{w\backslash\text{o mismatch}} + \text{sign*dmu\_mdev*}\sigma_{\beta}^{mismatch}$$

Note: the sign is equal to -1 or 1 depending onn the following convention:

If dvt_dev (or dmu_dev) is negative |VT| (or mobility) is decreased.

These mismatch corner contributions are locally activated using the instance parameter “mismatch” (mismatch=1).

Moreover, they are also globally activated using the <family>_dev parameter (<family>_dev=3). The parameter <family>_dev can be defined in the netlist or via the mismatch.lib file.

See graphic examples in the chapter 3

1.4 Mismatch through Monte-Carlo simulations

For Monte-Carlo simulations, the standard deviation values are used in order to define two gaussian distributions for VT and mobility mismatch variations.

These variations are then added to the pre-defined corner as described in chapter 1.2.

The mismatch Monte-Carlo simulations can be locally activated using the instance parameter “mismatch” (mismatch=1).

Moreover, it is also globally activated using the <family>_dev parameter (<family>_dev=1). The parameter <family>_dev can be defined in the netlist or via the mismatch.lib file.

2. SUMMARY TABLE

Parameters					**Simulation result**
Global	*Local (instance parameters)*			Monte - Carlo mismatch	
<family>_dev	**mismatch**	**dvt_mdev**	**dmu_mdev**		
0	**0 or 1**	**x**	**x**	**x**	• **process corner**
1	*1*	*x*	*x*	*N*	• *process corner*
1	**1**	**x**	**x**	**Y**	• **runs centered on process corner - mismatch activated** • **first run=process corner**
2	**1**	**x**	**x**	**x**	• **mismatch into corners**
3	*1*	*not netlisted or equal to 0*	*not netlisted or equal to 0*	*x*	• *process corner*
3	*1*	*netlisted*	*not netlisted or equal to 0*	*x*	• *process corner + local Vt deviation*
3	*1*	*not netlisted or equal to 0*	*netlisted*	*x*	• *process corner + local mobility deviation*
3	**1**	**netlisted**	**netlisted**	**x**	• **process corner + local Vt & mobility deviations**

Notes:

- The useful model combinations are highlighted in bold characters in the table. The useless combinations are written in italic.
- x = no impact on simulation results, whether the parameter is netlisted or not
- For eldo, if you want to remove the nominal run of a montecarlo simulation, netlist '.mc nonom'

3. GRAPHIC PRESENTATIONS OF THE VARIOUS CORNERS

Notes:

- The figures 1, 2, 3 present the mismatch into corners (TSMC alignment). The mismatch into corners have to be understood as ‘TSMC-like’, they don’t aim to fully fit TSMC corners.
- The mismatch into corners aim to represent the corners of statistical process and mismatch spreads (in eldo, .mc with STAT model and mismatch).
- The figure 4 presents the position of the various corners when executing a mismatch monte-carlo simulation in a process corner. In particular, it shows the relative position of the process+mismatch corners vs mismatch deviations at the process corner.

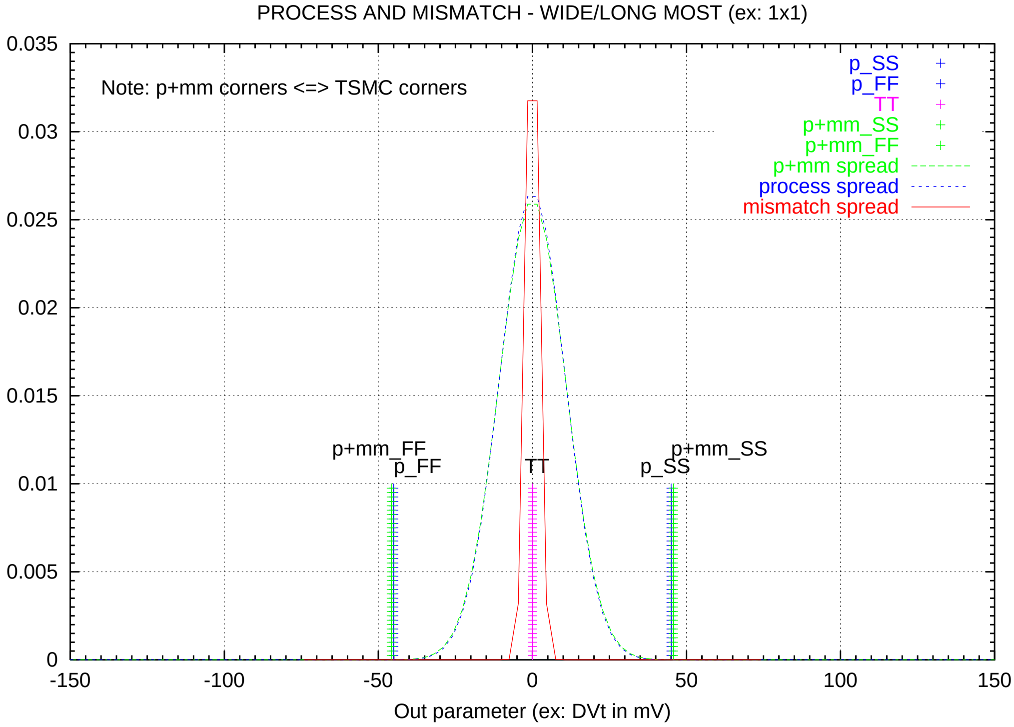

Figure 1 process corners vs process+mismatch corners for wide and long MOST

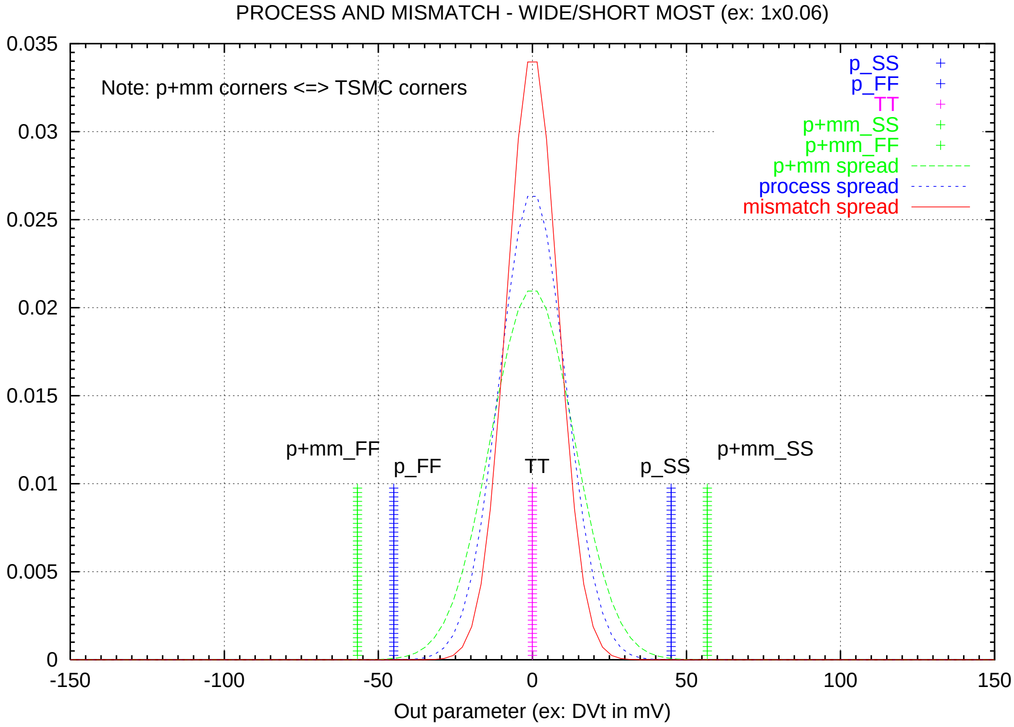

Figure 2 process corners vs process+mismatch corners for wide and short MOST

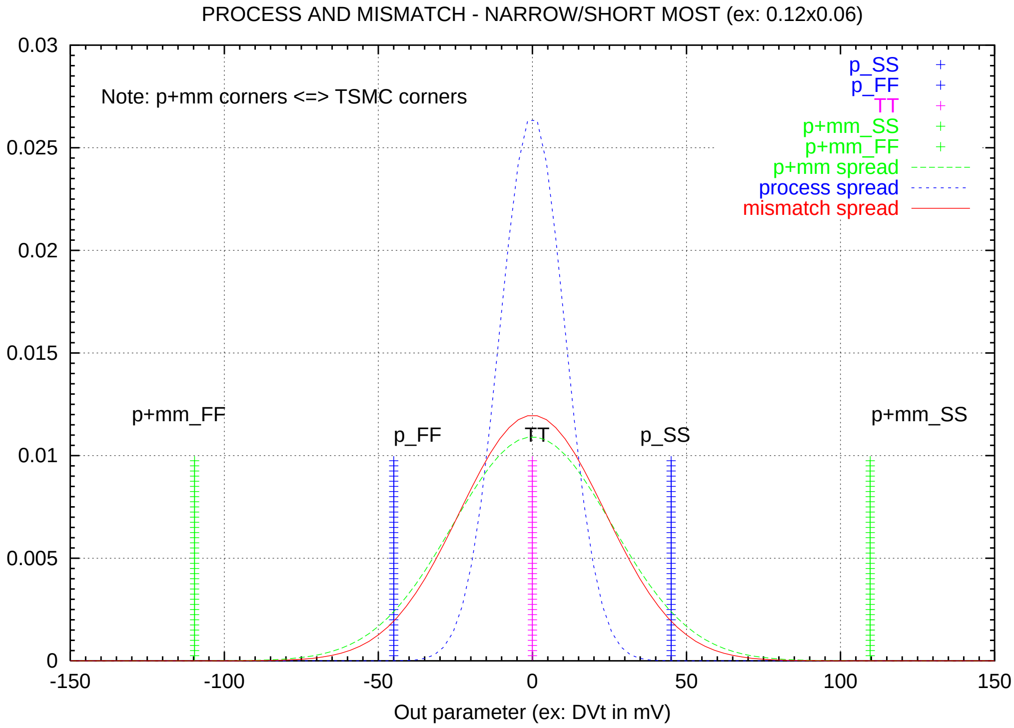

Figure 3 process corners vs process+mismatch corners for narrow and short MOST

Check latest version

‘

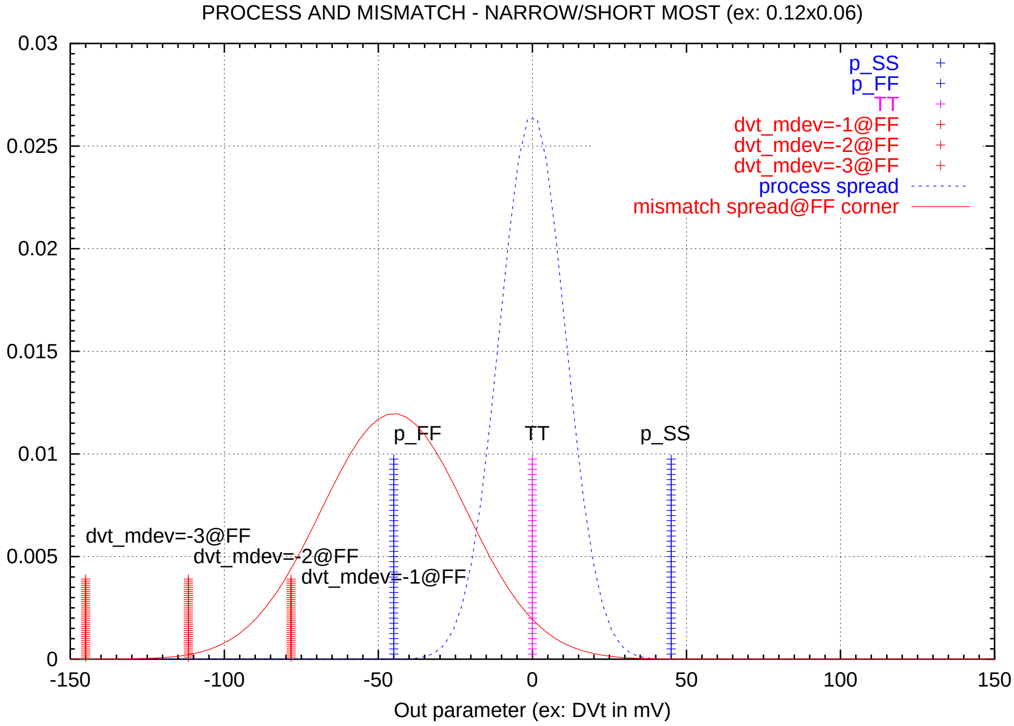

Figure 4 process/process+mismatch corners and mc mismatch distribution in corners