RFCELL · ENABLING PCELLS IN CMOS

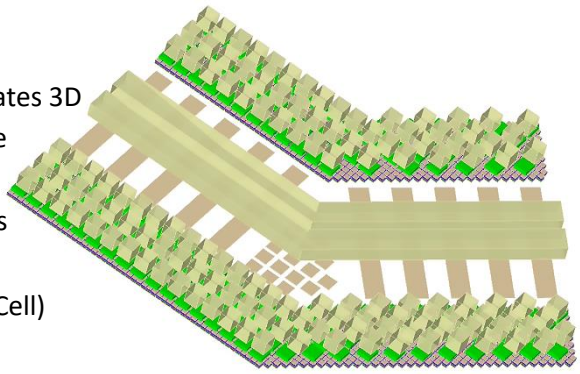
Description:

RFCELL is a Python-based framework (developed by Multifractal Semi.) for generating RF & mmWave parameterised-cells (PCELL's) with a specific focus on CMOS technology nodes. The framework provides high-level abstract types, which are ideally suited for defining RF & EM structures in an easy and intuitive way: without the loss of generality. The developer (RFIC engineer) can "think in 1D wire-frame mode" and focus on the design at hand, while the 3D details are auto-generated, in a consistent DRC compliant manner. RFCELL is process agnostic (DRC rules are provided in a process tech file) and with little additional effort from the designer, DRC & on-grid compliance are ensured.

Another key feature of RFCELL is the generation of layout simplification levels - enabling fast EM simulation as well as final tape-out-ready layout generation, all from the same source code. This eliminates the need for additional import/export steps between tools, which are error-prone and time-consuming.

RFCELL has been Si-proven in TSMC's 28 HPC+RF CMOS process and includes an extensive EM component library. We also provide a custom RFCELL component design service, in various CMOS nodes.

Features:

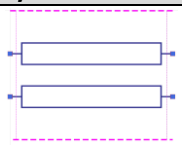
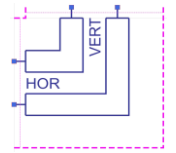
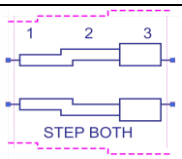
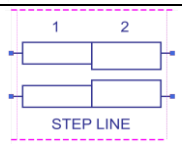
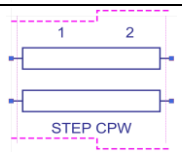
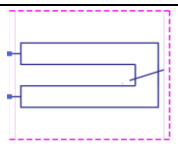
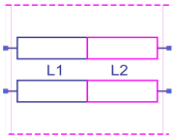
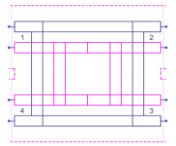
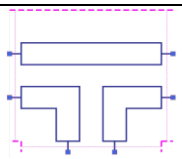
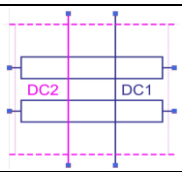
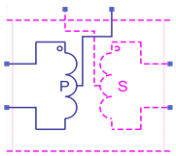


- Python-based: easy to code and read
- Enables rapid OA RF PCELL development
  - Code in "1D wire-frame" mode – RFCELL generates 3D
  - Define complex structures in a few lines of code
  - DRC clean and process agnostic: tech file
- Shortens development time from months to weeks
- Si-proven in 28CMOS
- Integrates with Keysight ADS AEL flow (not just PyCell)
- Integrates with Synopsys PyCell & Cadence

RFCELL library (*representative* sample):

A *sample* of our RFCELL library is shown below. Each component contains certain common objects: signal lines, shields (typically perforated planes), slow-wave (sw) fill regions & side boundary (cpw) regions. `*std_params` contains the following universal parameters associated with these objects.

Parameter <type>	Description	Example
`line_layer, cpw_layer, sw_layer, shield_layer <str>|<str/str>` or `list(<str>|<str/str>)`	Layers of various objects. If two layers defined then intermediate layers are stitched/drawn.	`"M7"; "AP/M7"`
`exclusion_layer <str>|<str/str>`	Layers to excl. from fill in defined boundary.	`"M6/M1"`
`line_width, line_gap <float>|list(<float>)`	Input/output signal line widths/gaps.	`10, 5`
`cpw_width, cpw_gap <float>`	Side boundary (ref. to as cpw) width/gap.	`25, 10`
`cpw_fill <bool>`	CPW can be either a solid or fill.	`True`
`sw_width, sw_edge_gap, sw_seperation <float>`	Fill/slow-wave structure parameters.	`4, 4, 2`
`simplify <int>`	0-3. 0: tape-out ready, 3: faster EM sim.	`0`

RFCELL	Description, parameters	Sim. model	Symbol
diff_straight_tl	Differential t-line *std_params, orth_or_diag <str>, length <float>	Circuit (0, 30-100 GHz) EM (0-200 GHz)	
diff_corner_tl	Differential t-line corner *std_params, hor_length <float>, vert_length <float>, mitter <float>, 135_or_90_bend <int>, cpw_gap_inner/outer <float>, cpw_width_inner/outer <float>	EM (0-200 GHz)	
diff_step_both_tl	Differential t-line step, with step in the CPW *std_params, length1/2/3 <float>, line_width_1/2 <float>, line_gap_1/2 <float>		
diff_step_line_tl	Differential t-line step, with no step in the CPW *std_params, length1/2 <float>, line_width_1/2 <float>, line_gap_1/2 <float>		
diff_step_cpw_tl	Differential CPW step, with step only in the CPW *std_params, length1/2 <float>, cpw_gap_1/2 <float>		
diff_stub_tl	Differential stub, both open/short *std_params, open_or_shorted <bool>, mitter <float>		
diff_layer_transition_tl	Differential t-line transition between layers *std_params, line_layers <str/str>		
diff_branchline	Differential branch-line coupler *std_params, vert/hor_length <float>, vert/hor_width <float>, vert/hor_gap <float>, feed_length <float>, mitter <float>		
diff_T_tl	Differential t-line T-junction *std_params		
diff_dc_bridge_tl	Differential t-line, with orthogonal DC cross *std_params, dc_bridge_layer <str,str/str>, dc_line_length <float>, dc_line_width <float>		
diff_trans	Differential transformer (1:1), with DC biasing taps *std_params, rotation_p/s <str>("NE","E"…), segment_length_p/s <float>, feed_length_p/s <float>, width_p/s <float>, signal_gap_p/s <float>, dc_access_type_p/s <int>, dc_length_p/s <float>	Circuit (0, 60-100 GHz) EM (0-200 GHz)	
via/fill	Rectangle filled with vias or fill *std_params	None	Layout only