MIXED SIGNAL PHASE FREQUENCY DETECTOR USING SKY130

Theory:

The Phase Frequency Detector belongs to a class of sequential Phase detectors with internal state. Assume that initially both outputs of the Phase Frequency Detector (PFD), UP and DOWN, are at 0s. When the Clock A rises first, the flip-flop triggered by the clock asserts UP high. When the Clock B rises later, the other flip-flop asserts DN as well. But then, the AND logic connected to the asynchronous reset input of the flip-flops deasserts both UP and DN signals to 0 as soon as they both reach 1s, returning the Phase Frequency Detector to the original state. The resulting difference in the UP and DN pulse widths corresponds to the timing difference between the two clocks' rising edges. This functionality can be used to detect both frequency and phase changes.

Schematic Diagram:

The schematic diagram of PFD is shown below:

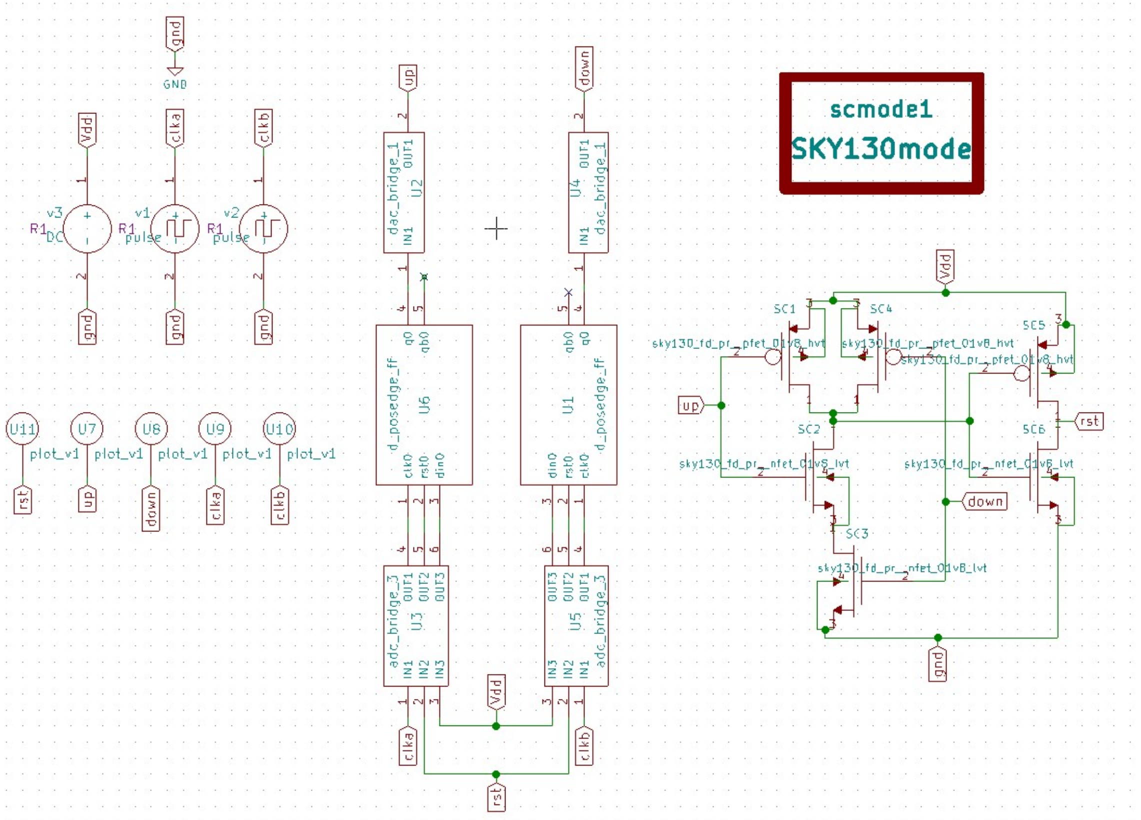


Figure 1: Schematic Diagram

Simulation Results:

1. **Ngspice plots**

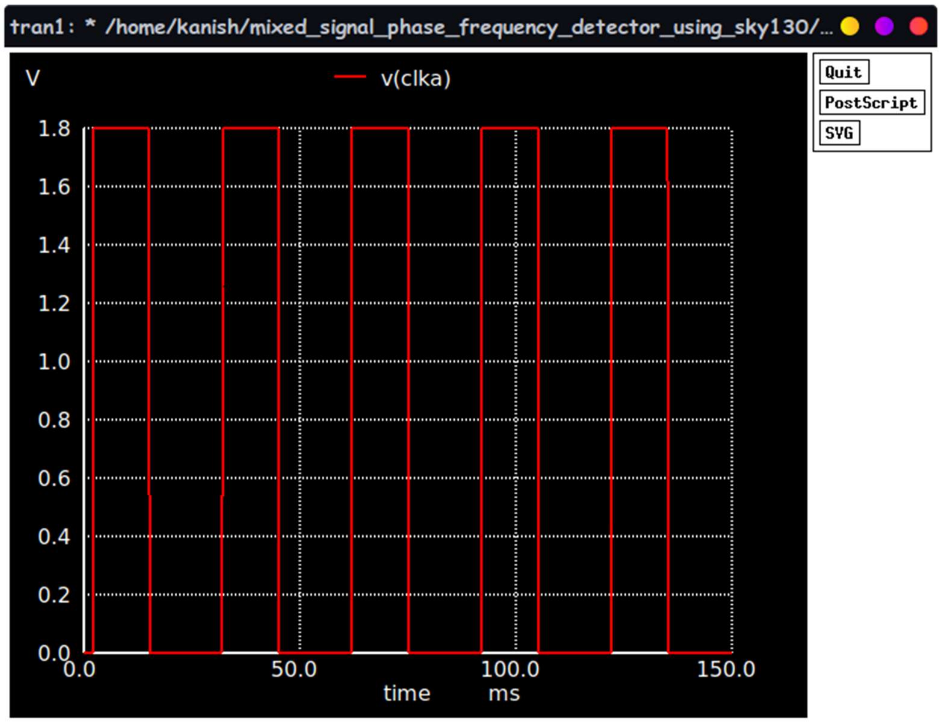


Figure 2: Clock A

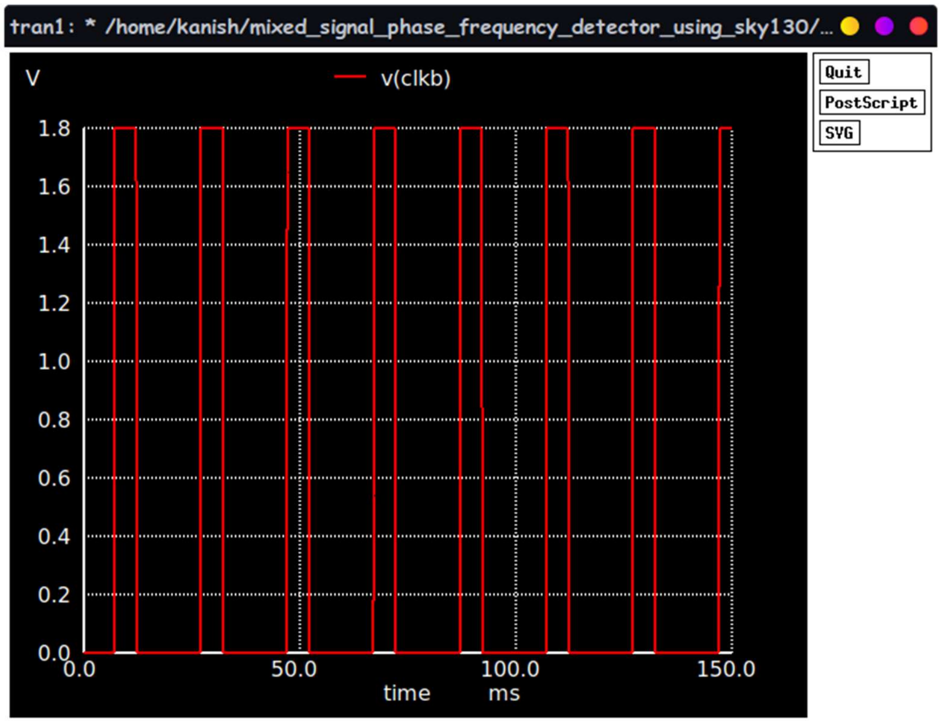


Figure 3: Clock B

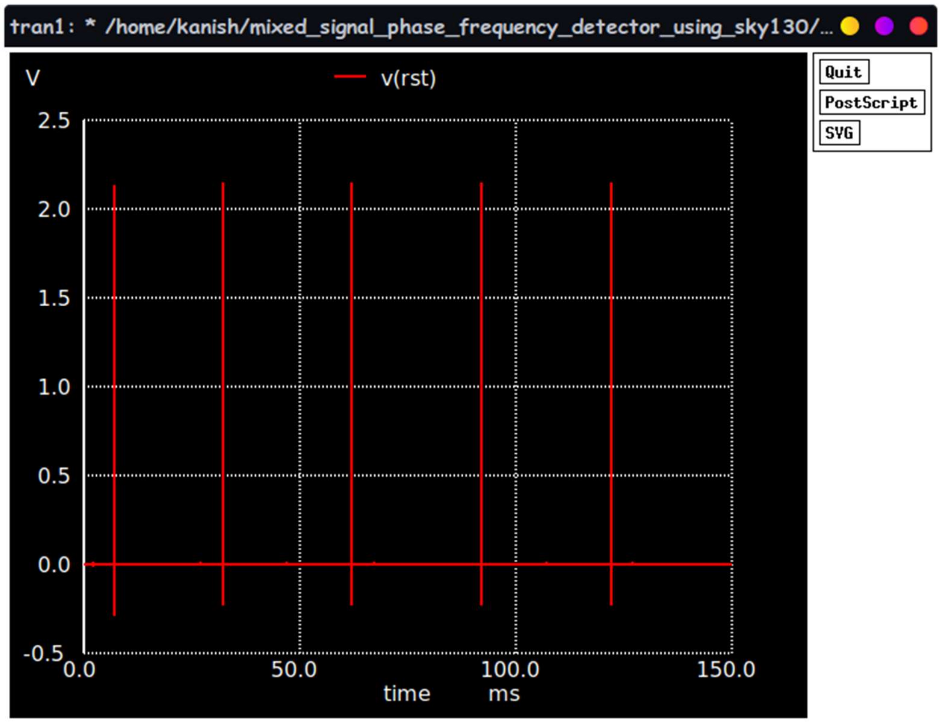


Figure 4: Reset Signal

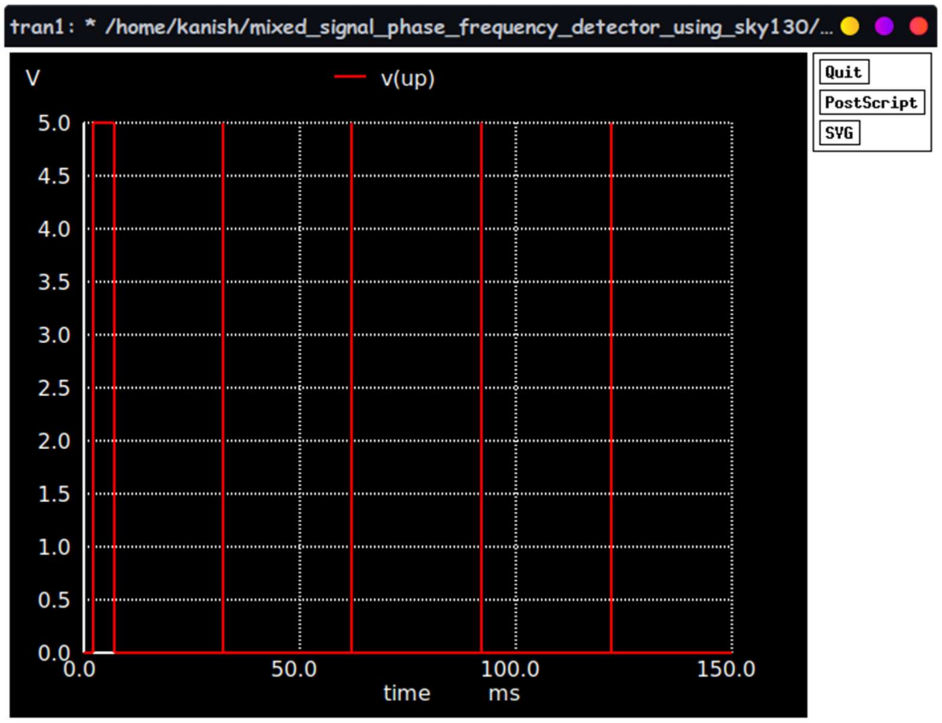


Figure 5: Up Output

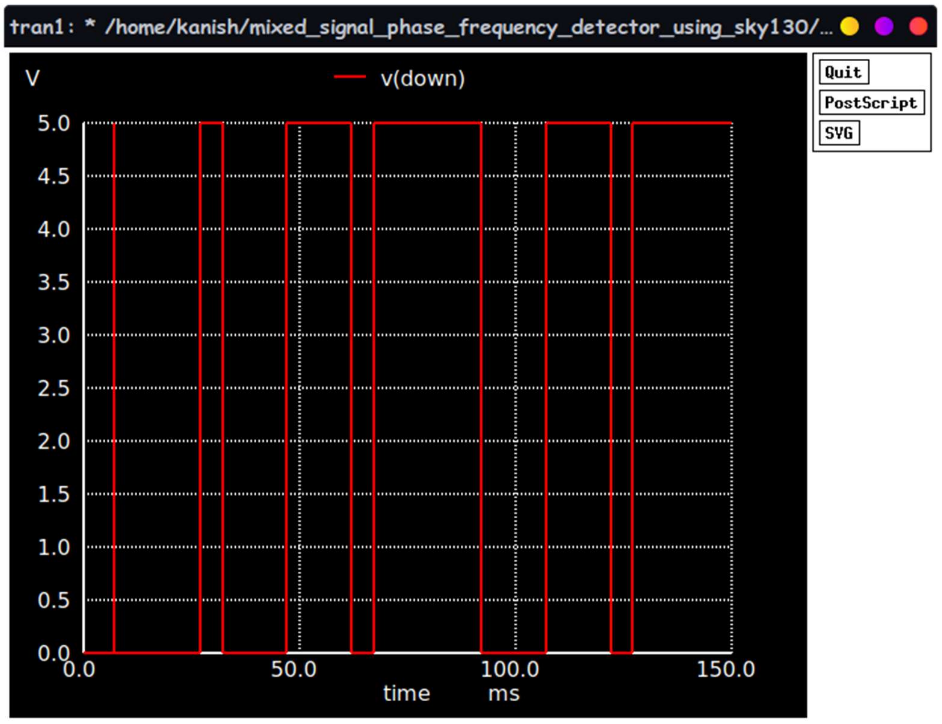


Figure 6: Down Output

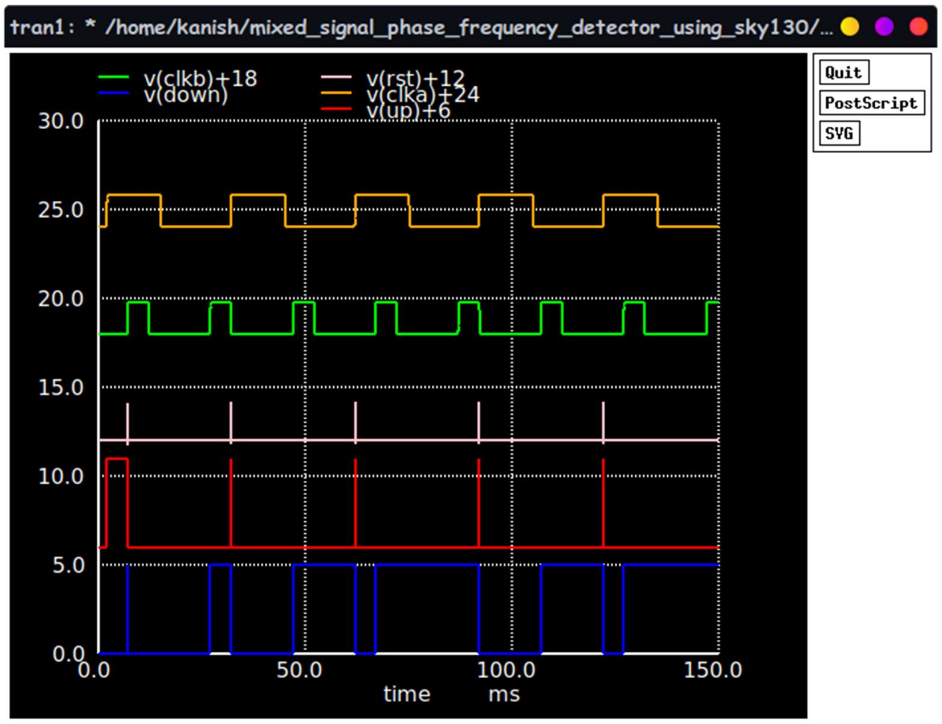
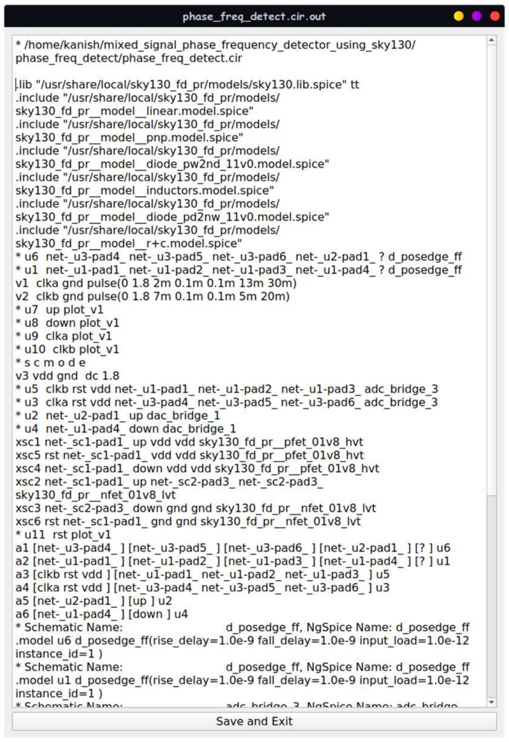


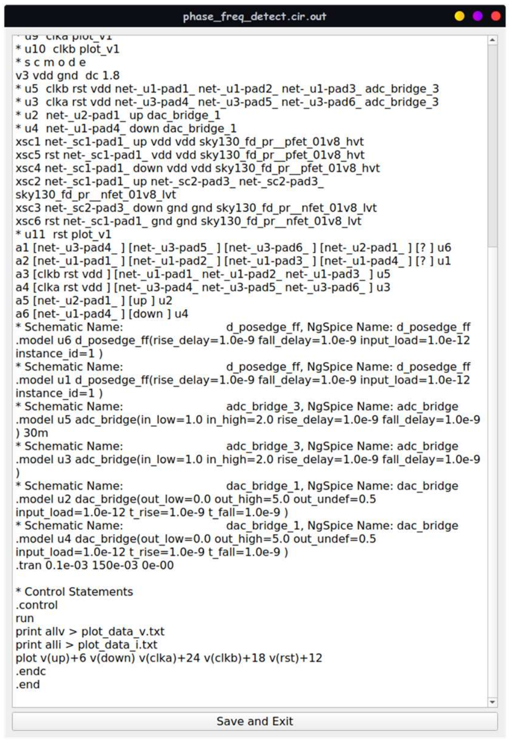
Figure 7: Combined Output

2. **Netlist Output**



```
* /home/kanish/mixed_signal_phase_frequency_detector_using_sky130/
phase_freq_detect/phase_freq_detect.cir

.lib "/usr/share/local/sky130_fd_pr/models/sky130.lib.spice" tt
.include "/usr/share/local/sky130_fd_pr/models/
sky130_fd_pr__model__linear.model.spice"
.include "/usr/share/local/sky130_fd_pr/models/
sky130_fd_pr__model__pnp.model.spice"
.include "/usr/share/local/sky130_fd_pr/models/
sky130_fd_pr__model__diode_pw2nd_11v0.model.spice"
.include "/usr/share/local/sky130_fd_pr/models/
sky130_fd_pr__model__inductors.model.spice"
.include "/usr/share/local/sky130_fd_pr/models/
sky130_fd_pr__model__diode_pd2nw_11v0.model.spice"
.include "/usr/share/local/sky130_fd_pr/models/
sky130_fd_pr__model__r+c.model.spice"
* u6  net-_u3-pad4_ net-_u3-pad5_ net-_u3-pad6_ net-_u2-pad1_ ? d_posedge_ff
* u1  net-_u1-pad1_ net-_u1-pad2_ net-_u1-pad3_ net-_u1-pad4_ ? d_posedge_ff
v1  clka gnd pulse(0 1.8 2m 0.1m 0.1m 13m 30m)
v2  clkb gnd pulse(0 1.8 7m 0.1m 0.1m 5m 20m)
* u7  up plot_v1
* u8  down plot_v1
* u9  clka plot_v1
* u10  clkb plot_v1
* s c m o d e
v3 vdd gnd  dc 1.8
* u5  clkb rst vdd net-_u1-pad1_ net-_u1-pad2_ net-_u1-pad3_ adc_bridge_3
* u3  clka rst vdd net-_u3-pad4_ net-_u3-pad5_ net-_u3-pad6_ adc_bridge_3
* u2  net-_u2-pad1_ up dac_bridge_1
* u4  net-_u1-pad4_ down dac_bridge_1
xsc1 net-_sc1-pad1_ up vdd vdd sky130_fd_pr__pfet_01v8_hvt
xsc5 rst net-_sc1-pad1_ vdd vdd sky130_fd_pr__pfet_01v8_hvt
xsc4 net-_sc1-pad1_ down vdd vdd sky130_fd_pr__pfet_01v8_hvt
xsc2 net-_sc1-pad1_ up net-_sc2-pad3_ net-_sc2-pad3_
sky130_fd_pr__nfet_01v8_lvt
xsc3 net-_sc2-pad3_ down gnd gnd sky130_fd_pr__nfet_01v8_lvt
xsc6 rst net-_sc1-pad1_ gnd gnd sky130_fd_pr__nfet_01v8_lvt
* u11  rst plot_v1
a1 [net-_u3-pad4_ ] [net-_u3-pad5_ ] [net-_u3-pad6_ ] [net-_u2-pad1_ ] [? ] u6
a2 [net-_u1-pad1_ ] [net-_u1-pad2_ ] [net-_u1-pad3_ ] [net-_u1-pad4_ ] [? ] u1
a3 [clkb rst vdd ] [net-_u1-pad1_ net-_u1-pad2_ net-_u1-pad3_ ] u5
a4 [clka rst vdd ] [net-_u3-pad4_ net-_u3-pad5_ net-_u3-pad6_ ] u3
a5 [net-_u2-pad1_ ] [up ] u2
a6 [net-_u1-pad4_ ] [down ] u4
* Schematic Name:                    d_posedge_ff, NgSpice Name: d_posedge_ff
.model u6 d_posedge_ff(rise_delay=1.0e-9 fall_delay=1.0e-9 input_load=1.0e-12
instance_id=1 )
* Schematic Name:                    d_posedge_ff, NgSpice Name: d_posedge_ff
.model u1 d_posedge_ff(rise_delay=1.0e-9 fall_delay=1.0e-9 input_load=1.0e-12
instance_id=1 )
* Schematic Name:                    adc_bridge_3, NgSpice Name: adc_bridge
```



```
* u9  clka plot_v1
* u10  clkb plot_v1
* s c m o d e
v3 vdd gnd  dc 1.8
* u5  clkb rst vdd net-_u1-pad1_ net-_u1-pad2_ net-_u1-pad3_ adc_bridge_3
* u3  clka rst vdd net-_u3-pad4_ net-_u3-pad5_ net-_u3-pad6_ adc_bridge_3
* u2  net-_u2-pad1_ up dac_bridge_1
* u4  net-_u1-pad4_ down dac_bridge_1
xsc1 net-_sc1-pad1_ up vdd vdd sky130_fd_pr__pfet_01v8_hvt
xsc5 rst net-_sc1-pad1_ vdd vdd sky130_fd_pr__pfet_01v8_hvt
xsc4 net-_sc1-pad1_ down vdd vdd sky130_fd_pr__pfet_01v8_hvt
xsc2 net-_sc1-pad1_ up net-_sc2-pad3_ net-_sc2-pad3_
sky130_fd_pr__nfet_01v8_lvt
xsc3 net-_sc2-pad3_ down gnd gnd sky130_fd_pr__nfet_01v8_lvt
xsc6 rst net-_sc1-pad1_ gnd gnd sky130_fd_pr__nfet_01v8_lvt
* u11  rst plot_v1
a1 [net-_u3-pad4_ ] [net-_u3-pad5_ ] [net-_u3-pad6_ ] [net-_u2-pad1_ ] [? ] u6
a2 [net-_u1-pad1_ ] [net-_u1-pad2_ ] [net-_u1-pad3_ ] [net-_u1-pad4_ ] [? ] u1
a3 [clkb rst vdd ] [net-_u1-pad1_ net-_u1-pad2_ net-_u1-pad3_ ] u5
a4 [clka rst vdd ] [net-_u3-pad4_ net-_u3-pad5_ net-_u3-pad6_ ] u3
a5 [net-_u2-pad1_ ] [up ] u2
a6 [net-_u1-pad4_ ] [down ] u4
* Schematic Name:                    d_posedge_ff, NgSpice Name: d_posedge_ff
.model u6 d_posedge_ff(rise_delay=1.0e-9 fall_delay=1.0e-9 input_load=1.0e-12
instance_id=1 )
* Schematic Name:                    d_posedge_ff, NgSpice Name: d_posedge_ff
.model u1 d_posedge_ff(rise_delay=1.0e-9 fall_delay=1.0e-9 input_load=1.0e-12
instance_id=1 )
* Schematic Name:                    adc_bridge_3, NgSpice Name: adc_bridge
.model u5 adc_bridge(in_low=1.0 in_high=2.0 rise_delay=1.0e-9 fall_delay=1.0e-9
) 30m
* Schematic Name:                    adc_bridge_3, NgSpice Name: adc_bridge
.model u3 adc_bridge(in_low=1.0 in_high=2.0 rise_delay=1.0e-9 fall_delay=1.0e-9
)
* Schematic Name:                    dac_bridge_1, NgSpice Name: dac_bridge
.model u2 dac_bridge(out_low=0.0 out_high=5.0 out_undef=0.5
input_load=1.0e-12 t_rise=1.0e-9 t_fall=1.0e-9 )
* Schematic Name:                    dac_bridge_1, NgSpice Name: dac_bridge
.model u4 dac_bridge(out_low=0.0 out_high=5.0 out_undef=0.5
input_load=1.0e-12 t_rise=1.0e-9 t_fall=1.0e-9 )
.tran 0.1e-03 150e-03 0e-00

* Control Statements
.control
run
print allv > plot_data_v.txt
print alli > plot_data_i.txt
plot v(up)+6 v(down) v(clka)+24 v(clkb)+18 v(rst)+12
.endc
.end
```

Conclusion:

Thus the Mixed Signal Phase Frequency Detector is implemented using Verilog and SKY130PDK in eSim.

References:

Neil H.E. Weste, David Money Harris —CMOS VLSI Design: A Circuits and Systems Perspective.