1. tcd example

1.1 gate tcd

```
Core(gps_baseband) {
  DesignInfo {
    design_id : gate;
    design_level : physical_block;
    ChildBlockModules {
    }
    Clocks {
      Clock(clk) {
        domain_label : clk;
        type : async_source;
        base_period : 3.00ns;
        period : 3.0;
        posedge_scannable_flop_count : 7762;
        negedge_scannable_flop_count : 0;
        posedge_non_scannable_flop_count : 0;
        negedge_non_scannable_flop_count : 0;
        active_high_latch_count : 0;
        active_low_latch_count : 0;
        posedge_memory_clock_input_count : 0;
        negedge_memory_clock_input_count : 0;
      }
    }
    DynamicDftSignals {
      DynamicDftSignal(async_set_reset_dynamic_disable) {
        Instance {
          type : pin;
          created_from_other_signals : on;
          add_hold_scan_cell : off;
          node_name :
tessent_persistent_cell_async_set_reset_dynamic_disable/Y;
        }
      }
      DynamicDftSignal(edt_clock) {
        Instance {
          type : pin;
          node_name :
wrp/gps_baseband_rtl1_tessent_ssn_scan_host_1_inst/edt_clock;
          At {
            ssh_dft_driver_node_name :
wrp/gps_baseband_rtl1_tessent_ssn_scan_host_1_inst/edt_clock;
          }
        }
      }
      DynamicDftSignal(edt_update) {
```

```
        Instance {
          type : pin;
          add_hold_scan_cell : off;
          node_name :
wrp/gps_baseband_rtl1_tessent_ssn_scan_host_1_inst/edt_update;
          At {
            ssh_dft_driver_node_name :
wrp/gps_baseband_rtl1_tessent_ssn_scan_host_1_inst/edt_update;
          }
        }
      }
      DynamicDftSignal(scan_en) {
        Instance {
          type : pin;
          add_hold_scan_cell : off;
          node_name :
wrp/gps_baseband_rtl1_tessent_ssn_scan_host_1_inst/scan_en;
          At {
            ssh_dft_driver_node_name :
wrp/gps_baseband_rtl1_tessent_ssn_scan_host_1_inst/scan_en;
          }
        }
      }
      DynamicDftSignal(shift_capture_clock) {
        Instance {
          type : pin;
          node_name :
wrp/gps_baseband_rtl1_tessent_ssn_scan_host_1_inst/shift_capture_clock;
          At {
            ssh_dft_driver_node_name :
wrp/gps_baseband_rtl1_tessent_ssn_scan_host_1_inst/shift_capture_clock;
          }
        }
      }
    }
  }
  Scan {
    allow_internal_pins : 1;
    is_hard_module : 1;
    exclude_from_concatenated_netlist : 1;
    internal_scan_only : 1;
    Mode(int_edt_mode) {
      type : internal;
      enable_dft_signal : int_edt_mode;
      traceable : 1;
      make_active_automatically : 1;
      EdtInstances {
        wrp/gps_baseband_rtl1_tessent_edt_c1_int_inst :
gps_baseband_rtl1_tessent_edt_c1_int;
      }
      OccInstances {
```

```
        wrp/gps_baseband_rtl1_tessent_occ_clk_inst :
gps_baseband_rtl1_tessent_occ;
      }
ScanEn(wrp/gps_baseband_rtl1_tessent_ssn_scan_host_1_inst/tessent_persistent
_cell_scan_en_buf/Y) {
        pipeline_count : 0;
        active_polarity : all_ones;
      }
Clock(wrp/gps_baseband_rtl1_tessent_ssn_scan_host_1_inst/clock_gen/clock_sig
nals/tessent_persistent_cell_shift_capture_clock_buf/Y) {
        off_state : 1'b0;
      }
      ChildBlockInstance(wrp/gps_baseband_rtl1_tessent_occ_clk_inst) {
        child_mode_name : "";
        enable_dft_signal : "";
      }
    }
    Mode(ext_edt_mode) {
      type : external;
      enable_dft_signal : ext_edt_mode;
      traceable : 1;
      make_active_automatically : 1;
      EdtInstances {
        wrp/gps_baseband_rtl1_tessent_edt_c1_ext_inst :
gps_baseband_rtl1_tessent_edt_c1_ext;
      }
ScanEn(wrp/gps_baseband_rtl1_tessent_ssn_scan_host_1_inst/tessent_persistent
_cell_scan_en_buf/Y) {
        pipeline_count : 0;
        active_polarity : all_ones;
      }
      Clock(clk) {
        off_state : 1'b0;
        is_capture_clock : true;
      }
Clock(wrp/gps_baseband_rtl1_tessent_ssn_scan_host_1_inst/clock_gen/clock_sig
nals/tessent_persistent_cell_shift_capture_clock_buf/Y) {
        off_state : 1'b0;
      }
    }
    OccModule(gps_baseband_rtl1_tessent_occ) {
      supports_add_core_instances : 1;
      Iproc(setup) {
        parameter_value_list : fast_capture_mode, 1;
      }
      OccShiftOnlyEn(shift_only_mode) {
        active_polarity : 1'b1;
      }
      Clock(slow_clock) {
        off_state : 1'b0;
      }
```

```
      ClockOut(tessent_persistent_cell_clock_out_mux/Y) {
        slow_clock_input : slow_clock;
        fast_clock_input : fast_clock;
      }
      FastCaptureClockEn(tessent_persistent_cell_cgc_fast_clock/FE) {
        clock_input : fast_clock;
      }
ShiftRegisterClockEn(occ_control/tessent_persistent_cell_cgc_SHIFT_REG_CLK/F
E) {
        clock_input : slow_clock;
      }
    }
  }
}
```