

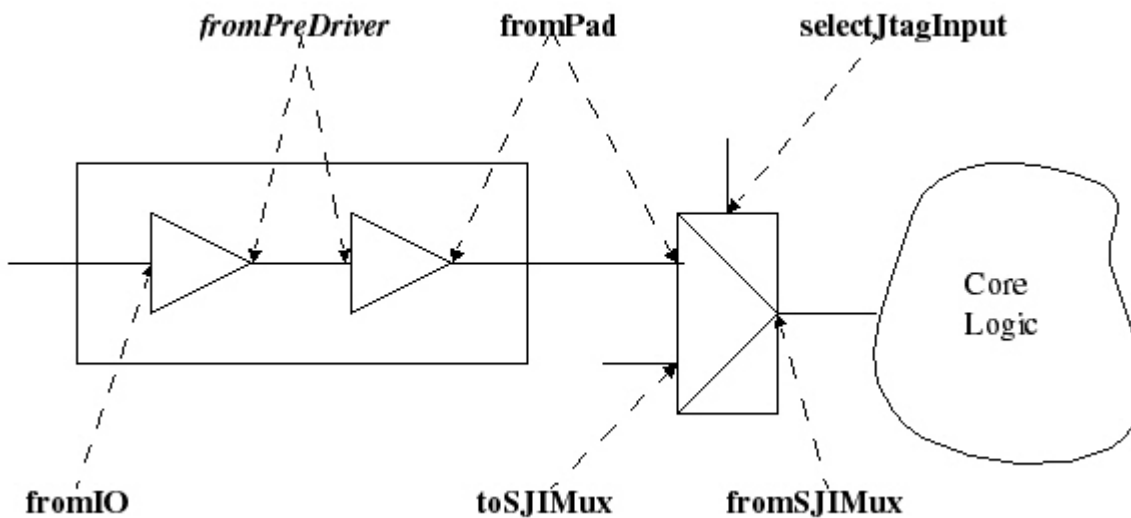
pad lib

1. LV FLOW lib

[ETAssemble Tool Reference.Pad Library.Function](#)

Function Type	Description
forceDisable	Disables the output driver when it is set to 1. Usually, this pin is directly connected to the ForceDisable signal of the TAP. The forceDisable function does not support an active-low polarity or a pre-existing connection.
fromPad, toPad	Input/Output connections from/to a pad and the core module. Typically, these connections are intercepted and the boundary scan cells are inserted.
samplePad	Specifies that the pin is a test-only buffered copy of the fromPad output pin. While the fromPad pin always carries the functional data to the core, the samplePad pin reports the top-level pin state to the boundary-scan cells or to any other embedded test structure without disturbing the functional path timing.
enableHigh, enableLow	Identify the active high or active low enable pins for bidirectional or output pads.
fromSJEMux	Specifies the output of the SJEMux and provides feedback to the boundary-scan enable (EN) cell.
toSJEMuxLow, toSJEMuxHigh	Specify the test enable signal from the boundary-scan EN cell.
toSJIMux, fromSJIMux, toSJOMux, fromSJOMux	Specify the output/input on the pad cells that are connected to the Select JTAG Input (SJI) or output (SJO) multiplexer.
selectJtagEnable	Specifies the connection for the SJEMux control signal, usually connected directly to the selectJtagOutput pin of the TAP, except when the pin is specified as an auxiliary output pin.
selectJtagInput, selectJtagOutput	Specify the connection for the control signal, usually coming from the TAP controller, which controls the updating of the boundary-scan cell's internal register.
fromIO	Specifies the input connection from an input or inout top pin to its pad. If the pad description has this cell pin function but neither toIO nor padIO cell pin function, then the BSDL top pin direction will be forced to input even if the direction of the associated top pin is inout.
toIO	Specifies the output connection from an output or inout top pin to its pad. If the pad description has this cell pin function but neither fromIO nor padIO cell pin function, then the BSDL top pin direction will be forced to output even if the direction of the associated top pin is inout.
padIO	Specifies the input/output connection from a top pin to its pad. The top pin can be any direction.
padIOInv	Specifies the input/output connection from the second top pin of a differential pin pair to its (differential) pad.
fromIOInv	Specifies the input connection from the second top pin of a differential pin pair which direction is input or inout to its (differential) pad.
toIOInv	Specifies the output connection from the second top pin of a differential pin pair whose direction is output or inout to its (differential) pad.

Function Type	Description
parametric	Specifies that the cell pin is used for parametric test.
DOT6-Related functions:	Group1: InitData, InitClk, TestData—This group is mandatory for all input and bidirectional AC pads.
	Group2: InitDataInv, TestDataInv—This group is mandatory for differential input and bidirectional AC pads.
	ACMode—This mode is optional. When this mode is present, ETAssemble adds the ACM subclass to the pad.
	For details, refer to the section Pad Library File in Support for IEEE 1149.6 Boundary Scan.



2. Tessent Cell Library

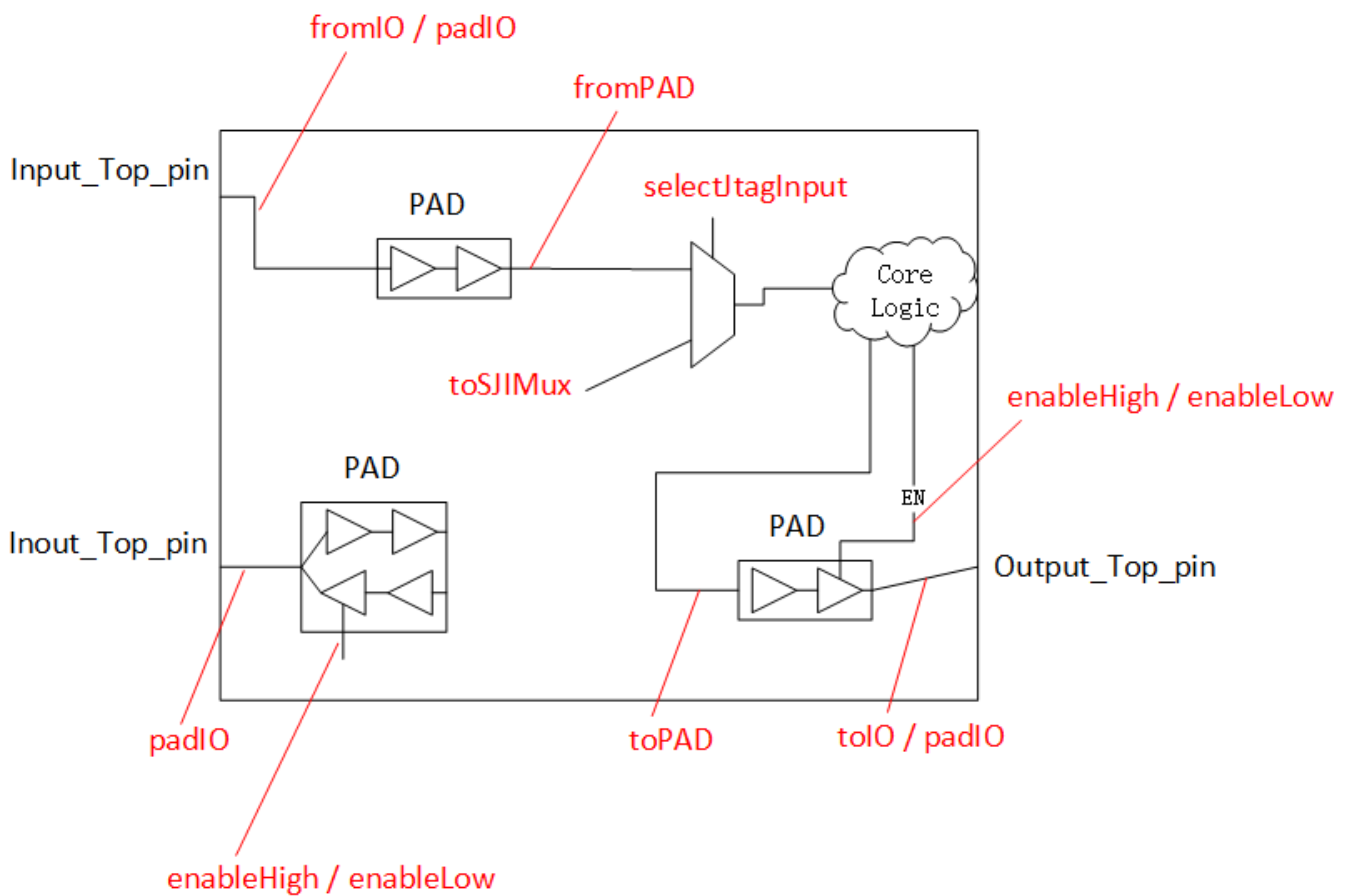
[Attribute Descriptions](#)

3. Tessent LV Flow Pad Library Mappings

[Tessent LV Flow Pad Library Mappings](#)

Tessent LV Flow Syntax (pad.library)	New Tessent Syntax	
forceDisable	pad_force_disable	
fromPad	pad_from_pad	
toPad	pad_to_pad	
fromIO	pad_from_io	
toIO	pad_to_io	
padIO	pad_pad_io	Specifies the input/output connection from a top pin to its pad
padIOInv	pad_pad_io_inv	
fromIOInv	pad_from_io_inv	
toIOInv	pad_to_io_inv	

Tessent LV Flow Syntax (pad.library)	New Tessent Syntax	
InitData	pad_init_data_dot6	
InitClk	pad_init_clock_dot6	
TestData	pad_test_data_dot6	
InitDataInv	pad_init_data_inv_dot6	
TestDataInv	pad_test_data_inv_dot6	
ACMode	pad_ac_mode_dot6	
enableHigh	pad_enable_high	Specifies the active high enable pins for bidirectional or output pads
enableLow	pad_enable_low	Specifies the active high enable pins for bidirectional or output pads
	pad_input_enable_high	
	pad_input_enable_low	



4. define a cell model

```

model model_name (list_of_pins) (
  model_source = verilog_module;
  cell_type = { clock_gating_and | clock_gating_or |
    buffer | inverter | or | and | nand | nor | xor | mux |
    clock_buffer | clock_inverter | clock_and | clock_or | clock_mux |
    clock_dff | clock_shaper | dff | latch | synchronizer_cell |
    scan_cell | pad | prohibited };

```

```

simulation_function = { clock_gating_and | clock_gating_or |
  func_only_clock_gating_and | func_only_clock_gating_or |
  clock_shaper_reset_active | clock_shaper_reset_inactive | buffer |
  inverter | or | and | nand | nor | xor | mux | dff | latch |
  synchronizer_cell | scan_cell | tie0 | tie1 } ;
pad_ac;
pad_nonjtag;
pad_ac_lp_time = <time_in_seconds>;
pad_ac_hp_time = <time_in_seconds>;
pad_ac_hp_on_chip;
bcell_lib_name = <bcell_lib_name>
nonscan_model = model_name | model_name (list_of_pins);
nonscan_equivalents = model_name1, modelname2 ...;
scan_equivalents = model_name1, modelname2 ...;
scan_length = integer;
gated_out_scan_cell;
mode(
  pad_ac;
  pad_ac_lp_time = <time_in_seconds>;
  pad_ac_hp_time = <time_in_seconds>;
  pad_ac_hp_on_chip [Yes | No];
  input (pin_name) (
    [pad_to_pad] [pad_pad_io][pad_pad_io_inv] [pad_enable_high]
    [pad_enable_low] [pad_to_sje_mux_low] [pad_to_sje_mux_high]
    [pad_to_sji_mux] [pad_to_sjo_mux] [pad_select_jtag_enable]
    [pad_select_jtag_in] [pad_select_jtag_out]
    [pad_two_state_output_enable_high]
    [pad_two_state_output_enable_low]
    [pad_init_data_dot6] [pad_init_data_inv_dot6]
    [pad_init_posedge_clock_dot6] [pad_init_negedge_clock_dot6]
    [pad_init_enable_high_dot6] [pad_init_enable_low_dot6]
    [pad_ac_mode_dot6]
    [pad_data_inv] [pad_diff_voltage] [pad_diff_current]
    [pad_from_io] [pad_from_io_inv] [pad_tied0] [pad_tied1]
    [pad_open])
  ) // end input section of mode

  input (pin_name2) ()
  ...
  input (pin_nameN) ()

  output(pin_name) (
    [pad_from_pad] [pad_pad_io] [pad_pad_io_inv]
    [pad_from_sje_mux] [pad_from_sji_mux] [pad_from_sjo_mux]
    [pad_sample_pad] [pad_data_inv] [pad_diff_voltage]
    [pad_diff_current] pad_open_drain] [pad_open_source]
    [pad_to_io] [pad_to_io_inv] [pad_pull0] [pad_pull1];
  ) // end output section of mode
  output (pin_name2) ()
  ...
  output (pin_nameN) ()

```

```

    inout(pin_name) (
        [pad_pad_io] [pad_pad_io_inv][pad_diff_voltage]
        [pad_diff_current] [pad_open_drain]
        [pad_open_source][pad_pull0] [pad_pull1]
    ) // end inoutsection of mode

    inout (pin_name2) ()
    ...
    inout (pin_nameN) ()

) // end mode

// Completed model level statements, including pad modes if any.
// Define model interface port directions, and attributes.

intern (intern_nodes) (intern_attributes)
input (pin_names) (list_of_input_attributes)
output (pin_names) (list_of_output_attributes)
inout (pin_names) (list_of_inout_attributes)

// Define hardware simulation function using cell library
// primitives and instances
(
    //hardware_statements, see Hardware Definitions
)

) // end model_name

```

5. input pad

```

model ipad
(PAD, C)
(
model_source = verilog_module;
cell_type = pad;

input (PAD) ( pad_from_io; )
output (C) ( pad_from_pad; )
(
instance = buf02 ipad_inst (PAD, C);
)
) // end model ipad

```

带EN控制

```

model ipad_ien
(PAD, C, IEN)
(
model_source = verilog_module;

```

```

cell_type = pad;

input (PAD) ( pad_from_io; )
input (IEN) ( pad_input_enable_high; )
output (C) ( pad_from_pad; )
(
// Input a 0 (low voltage) to core when input disabled.
primitive = _and mlc_and (PAD, IEN, C);
)
) // end model ipad_ien

```

差分PAD

```

model Diff_Pad
  (C,PADP,PADN)
  (
    cell_type = pad;

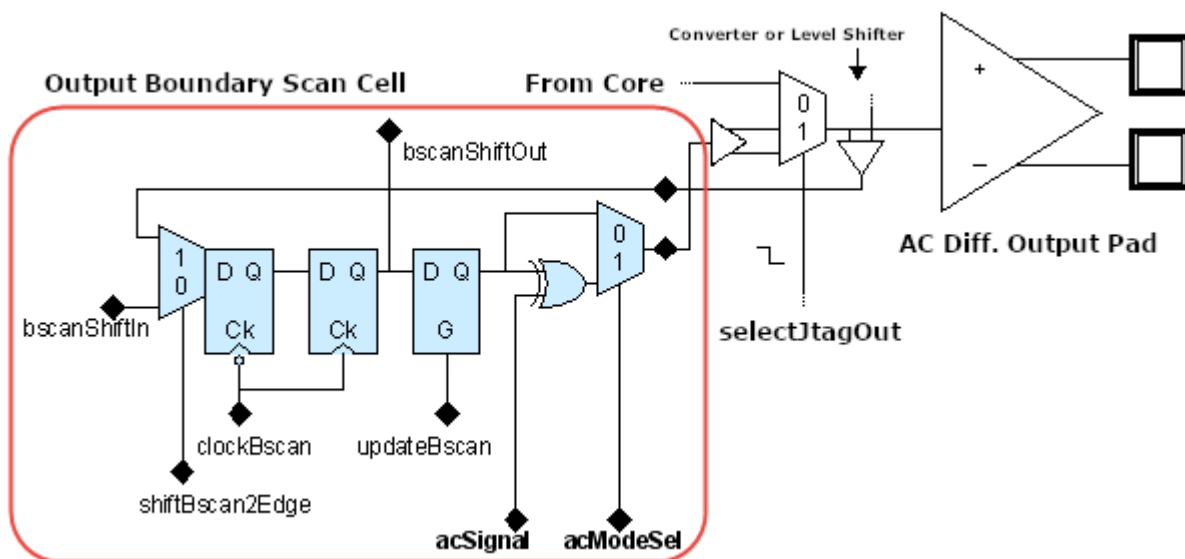
    input (PADP) ( pad_from_io; pad_diff_voltage)
    input (PADN) ( pad_from_io_inv; )
    output (C) ( pad_from_pad )

    (
primitive = _buf P1 (PADP, diff_net);
primitive = _inv P2 (PADN, diff_net);
primitive = _buf P3 (diff_net, C);

    )
  )
)

```

6. AC diff output pad



7. OD OS psd

属性pad_open_drain与pad_open_source生成出来的rtl电路区别：
仅在输出Z态的控制逻辑不一样：

- OD输出Z态时需要把数据输入force为1
- OS输出Z态时需要把数据输入force为0

```

1 module chip_top_rtl1_tessent_bscan_cell_bid_od_di (
2     fromCore,
3     selectJtagOutput,
4     toPad,
5     fromPad,
6     selectJtagInput,
7     forceDisable,
8     +-- 23 lines: clockBscan,-----+
9     reg bscanReg; // boundary scan register's output
10    reg updLatch; // update latch's output
11
12    assign captureMux = selectJtagInput ? SJO_Mux : SJI_Mux;
13    assign bscanShiftMux = shiftBscan2Edge ? bscanShiftIn : captureMux;
14    assign SJO_Mux = selectJtagOutput ? updLatch : fromCore;
15    assign toPad = SJO_Mux | forceDisable;
16    assign SJI_Mux = selectJtagInput ? updLatch : fromPad;
17    assign bscanShiftOut = retimeElem;
18
19    // bscan flop
20    always @(posedge clockBscan)
21        bscanReg <= bscanShiftMux;
22 +-- 11 lines: Retiming Latch-----+
23
24 module chip_top_rtl1_tessent_bscan_cell_bid_os_di (
25     fromCore,
26     selectJtagOutput,
27     toPad,
28     fromPad,
29     selectJtagInput,
30     forceDisable,
31     +-- 23 lines: clockBscan,-----+
32     reg bscanReg; // boundary scan register's outp
33     reg updLatch; // update latch's output
34
35     assign captureMux = selectJtagInput ? SJO_Mux : SJI_Mux;
36     assign bscanShiftMux = shiftBscan2Edge ? bscanShiftIn : captureMux;
37     assign SJO_Mux = selectJtagOutput ? updLatch : fromCore;
38     assign toPad = SJO_Mux & forceDisable;
39     assign SJI_Mux = selectJtagInput ? updLatch : f
40     assign bscanShiftOut = retimeElem;
41
42     // bscan flop
43     always @(posedge clockBscan)
44         bscanReg <= bscanShiftMux;
45 +-- 11 lines: Retiming Latch-----+

```

在数据输入pin上加入pad_data_inv与不加以区别：
OD情况：

```

1 module chip_top_rtl1_tessent_bscan_cell_bid_io_od_di (
2     fromCore,
3     selectJtagOutput,
4     toPad,
5     fromPad,
6     selectJtagInput,
7     forceDisable,
8     +-- 20 lines: clockBscan,-----+
9     wire captureMux; // Mux selects toPad or toCore as the capture
10    wire bscanShiftMux; // Mux selects bscanShiftIn or capture data
11    reg retimeElem; // Retiming element, after the bscan register
12    reg bscanReg; // boundary scan register's output
13    reg updLatch; // update latch's output
14
15    assign captureMux = selectJtagInput ? SJO_Mux : SJI_Mux;
16    assign bscanShiftMux = shiftBscan2Edge ? bscanShiftIn : captureMux;
17    assign SJO_Mux = selectJtagOutput ? updLatch : fromCore;
18    assign toPad = SJO_Mux | forceDisable;
19    assign SJI_Mux = selectJtagInput ? updLatch : fromPad;
20    assign bscanShiftOut = retimeElem;
21
22 module chip_top_rtl1_tessent_bscan_cell_bid_od_di (
23     fromCore,
24     selectJtagOutput,
25     toPad,
26     fromPad,
27     selectJtagInput,
28     forceDisable,
29     +-- 20 lines: clockBscan,-----+
30     wire captureMux; // Mux selects toPad or toCore as the ca
31     wire bscanShiftMux; // Mux selects bscanShiftIn or capture c
32     reg retimeElem; // Retiming element, after the bscan reg
33     reg bscanReg; // boundary scan register's output
34     reg updLatch; // update latch's output
35
36     assign captureMux = selectJtagInput ? SJO_Mux : SJI_Mux;
37     assign bscanShiftMux = shiftBscan2Edge ? bscanShiftIn : captur
38     assign SJO_Mux = selectJtagOutput ? updLatch : fromCore;
39     assign toPad = SJO_Mux | forceDisable;
40     assign SJI_Mux = selectJtagInput ? updLatch : fromPad;
41     assign bscanShiftOut = retimeElem;

```

OS情况：

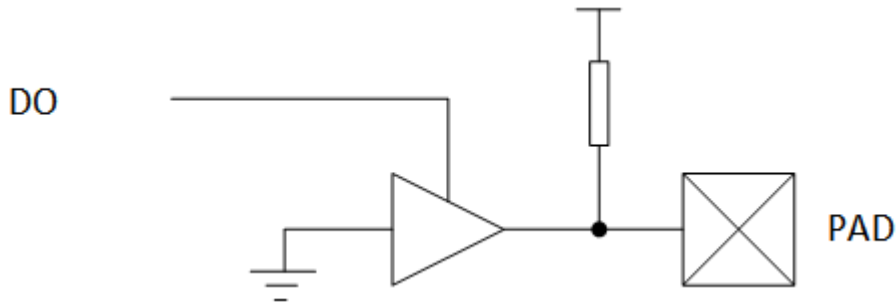
```

1 module chip_top_rtl1_tessent_bscan_cell_bid_io_os_di (
2     fromCore,
3     selectJtagOutput,
4     toPad,
5     fromPad,
6     selectJtagInput,
7     forceDisable,
8     +-- 20 lines: clockBscan,-----+
9     wire captureMux; // Mux selects toPad or toCore as the capture
10    wire bscanShiftMux; // Mux selects bscanShiftIn or capture data
11    reg retimeElem; // Retiming element, after the bscan register
12    reg bscanReg; // boundary scan register's output
13    reg updLatch; // update latch's output
14
15    assign captureMux = selectJtagInput ? SJO_Mux : SJI_Mux;
16    assign bscanShiftMux = shiftBscan2Edge ? bscanShiftIn : captureMux;
17    assign SJO_Mux = selectJtagOutput ? updLatch : fromCore;
18    assign toPad = SJO_Mux & ~forceDisable;
19    assign SJI_Mux = selectJtagInput ? updLatch : fromPad;
20    assign bscanShiftOut = retimeElem;
21
22    // bscan flop
23    always @(posedge clockBscan)
24        bscanReg <= bscanShiftMux;
25 +-- 12 lines: bscanReg <= bscanShiftMux;-----+
26
27 module chip_top_rtl1_tessent_bscan_cell_bid_os_di (
28     fromCore,
29     selectJtagOutput,
30     toPad,
31     fromPad,
32     selectJtagInput,
33     forceDisable,
34     +-- 20 lines: clockBscan,-----+
35     wire captureMux; // Mux selects toPad or toCore as th
36     wire bscanShiftMux; // Mux selects bscanShiftIn or captu
37     reg retimeElem; // Retiming element, after the bscan
38     reg bscanReg; // boundary scan register's output
39     reg updLatch; // update latch's output
40
41     assign captureMux = selectJtagInput ? SJO_Mux : SJI_Mux;
42     assign bscanShiftMux = shiftBscan2Edge ? bscanShiftIn : ca
43     assign SJO_Mux = selectJtagOutput ? updLatch : from
44     assign toPad = SJO_Mux & ~forceDisable;
45     assign SJI_Mux = selectJtagInput ? updLatch : fromP
46     assign bscanShiftOut = retimeElem;
47
48     // bscan flop
49     always @(posedge clockBscan)
50         bscanReg <= bscanShiftMux;
51 +-- 12 lines: bscanReg <= bscanShiftMux;-----+

```

7.1 output only 开漏 pad

只能驱动输出0，数据输入1时，输出0，数据输入0时高阻



```

model od_pad
  (DO, PAD)
  (
    model_source = verilog_module;
    cell_type = pad;
    mode(
      pin (DO) ( pad_to_pad; pad_data_inv;)
      pin (PAD) (pad_pad_io; pad_pull1; pad_open_source;)
    )
    input (DO) ( )
    inout (PAD) ( )
    (
      primitive = _pull mlc_pull_1 (1, PAD);

      primitive = _tsh mlc_tsh_1 (0, DO, PAD);
    )
  )
)

```

verilog mode

```

module od_pad (DO, PAD);
  input  DO;
  inout  PAD;

  pullup(PAD);

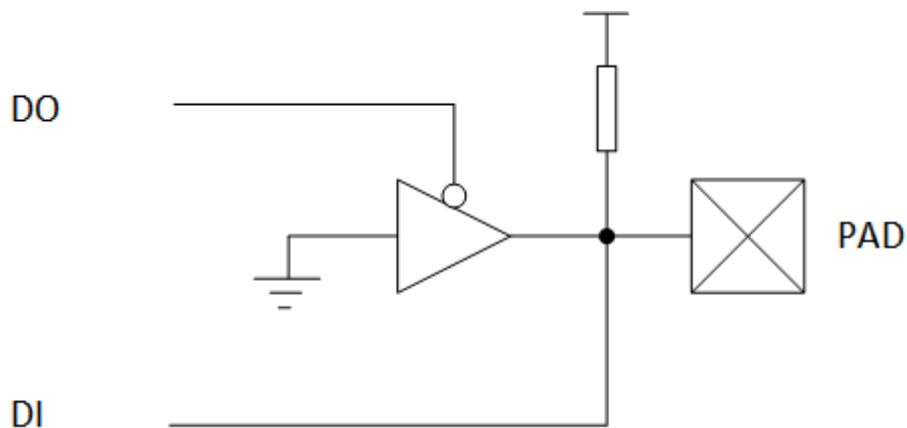
  assign PAD = (DO == 1'b1) ? 1'b0 : 1'bz;

endmodule

```

7.2 od pull1 pad

只能驱动输出0，数据输入0时，输出0，数据输入1时高阻



```

model od_pad
  (DI, DO, PAD)
  (
    model_source = verilog_module;
    cell_type = pad;

    mode(
      pin (PAD) (pad_pad_io; pad_pull1; pad_open_drain;)
      pin (DO) (pad_to_pad; )
      pin (DI) (pad_from_pad; )
    )

    input (DO) ( )
    inout (PAD) ( )
    output (DI) ( )
    (
      primitive = _buf mlc_buf_1 (PAD, DI);
      primitive = _pull mlc_pull_1 (1, PAD);

      primitive = _inv mlc_inv_1 (DO, DOinv);
      primitive = _tsh mlc_tsh_1 (DO, DOinv, PAD);
    )
  )

```

```

module od_pad (DI, DO, PAD);
output DI;
input DO;
inout PAD;

pullup(PAD);

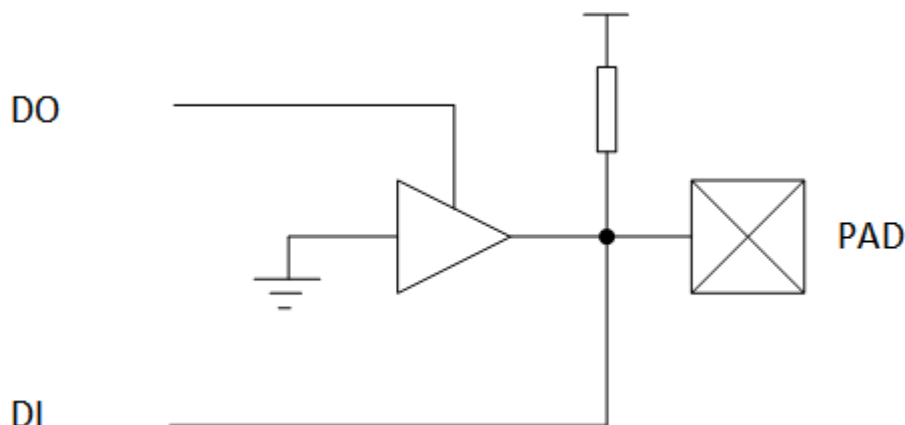
assign PAD = (DO == 1'b0) ? 1'b0 : 1'bz;
assign DI = PAD;

endmodule

```

7.3 od inv pull1 pad

只能驱动输出0，数据输入1时，输出0，数据输入0时高阻



```

model od_pad_inv
  (DI, DO, PAD)
  (
    model_source = verilog_module;
    cell_type = pad;

    mode(
      pin (PAD) (pad_pad_io; pad_pull1; pad_open_source;)
      pin (DO) (pad_to_pad; pad_data_inv;)
      pin (DI) (pad_from_pad; )
    )

    input (DO) ( )
    inout (PAD) ( )
    output (DI) ( )
    (
      primitive = _buf mlc_buf_1 (PAD, DI);
      primitive = _pull mlc_pull_1 (1, PAD);

      primitive = _inv mlc_inv_1 (DO, DOinv);
      primitive = _tsh mlc_tsh_1 (DOinv, DO, PAD);
    )
  )

```

```

module od_pad_inv (DI, DO, PAD);
output DI;
input DO;
inout PAD;

pullup(PAD);

assign PAD = (DO == 1'b1) ? 1'b0 : 1'bz;
assign DI = PAD;

```

```
endmodule
```

7.4 od inv pad

只能驱动输出0，数据输入1时，输出0，数据输入0时高阻

这种仿真会出错，因为pad_open_drain输出Z时会把DO force为1，但电路输出为0，比较不过。

这时候可以把生成出来的电路换成OS生成出来的电路，仿真可以PASS

```
model od_pad_inv
(DI, DO, PAD)
(
  model_source = verilog_module;
  cell_type = pad;
  mode(
    pin (PAD) (pad_pad_io; pad_open_drain;)
    pin (DO) (pad_to_pad; pad_data_inv;)
    pin (DI) (pad_from_pad; )
  )
  input (DO) ( )
  inout (PAD) ( )
  output (DI) ( )
  (
    primitive = _buf mlc_buf_1 (PAD, DI);

    primitive = _inv mlc_inv_1 (DO, DOinv);
    primitive = _tsh mlc_tsh_1 (DOinv, DO, PAD);
  )
)
```

```
module od_pad_inv (DI, DO, PAD);
output DI;
input DO;
inout PAD;

assign PAD = (DO == 1'b1) ? 1'b0 : 1'bz;
assign DI = PAD;
endmodule
```