

dft spec

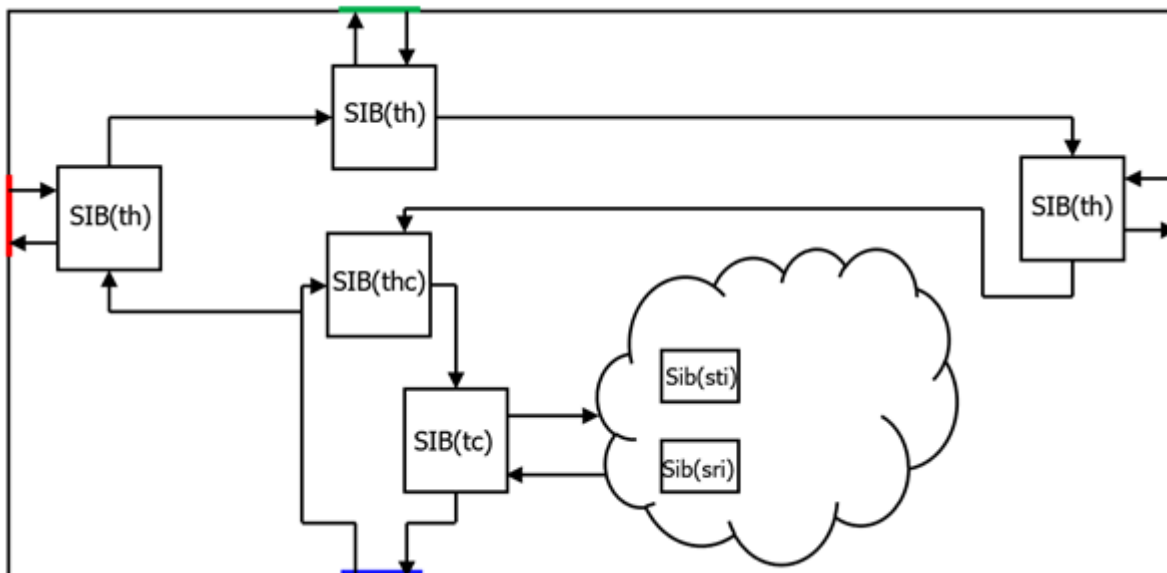
1. create_dft_specification

http://vmcc.vicp.net:9090/tessent_v2023.1_doc/htmldocs/mgchelp.htm#context=tshell_ref&id=142

```
create_dft_specification [-existing_ijtag_host_scan_in
host_scan_in_design_pin_spec]
[-existing_primary_tap_scan_out primary_tap_client_scan_out_design_pin_spec]
[-existing_bscan_host_scan_in bscan_host_scan_in_design_pin_spec]
[-tile_ijtag_host_list tile_ijtag_hosts]
[-stap_host_list stap_nodes]
[-active_high_compliance_enables enable_port_name ...]
[-active_low_compliance_enables enable_port_name ...]
[-sri_sib_list sri_sib_list]
[-sti_sib_list sti_sib_list]
[-replace]
```

1.1 example 4

example 4



```
set_dft_specification_requirements -design_type tile
check_design_rules
set spec [create_dft_specification -tile_ijtag_host_list {left top right}]
report_config_data $spec
```

```
DftSpecification(tile_core,gate) {
  IjtagNetwork {
    HostScanInterface(ijtag) {
```



```

}
TMSPort      tms      {
  Attribute forced_low_dft_signal_list = "tms_disable";
}
TRSTPort     trst     {
  Attribute connection_rule_option = "allowed_tied_high";
}
ToCaptureEnPort capture_dr_en;
ToShiftEnPort shift_dr_en;
ToUpdateEnPort update_dr_en;
ToResetPort  test_logic_reset { ActivePolarity 0; }
ToSelectPort host_1_to_sel { Source host_1_to_sel_int;
  Attribute connection_rule_option = "allowed_no_destination";
}
LogicSignal  host_1_to_sel_int { instruction == HOSTIJTAG_1; }
ScanInPort   host_1_from_so {
  Attribute connection_rule_option = "allowed_no_source";
}
ScanInPort   host_bscan_from_so {
  Attribute connection_rule_option      = "allowed_no_source";
  Attribute tessent_bscan_pipeline_stages = "0";
}
ToSelectPort host_bscan_to_sel { Source bscan_select_int;
  Attribute connection_rule_option = "allowed_no_destination";
  Attribute tessent_bscan_function = "select";
}
LogicSignal bscan_select_int {
  (instruction == EXTEST) ||
  (instruction == INTEST) ||
  (instruction == EXTEST_PULSE) ||
  (instruction == EXTEST_TRAIN) ||
  (instruction == SAMPLE) ||
  (instruction == PRELOAD) ;
}
DataOutPort  force_disable { Source force_disable_int;
  Attribute connection_rule_option = "allowed_no_destination";
  Attribute tessent_bscan_function = "force_disable";
}
LogicSignal  force_disable_int { instruction == HIGHZ; }
DataOutPort  select_jtag_input { Source select_jtag_input_int;
  Attribute connection_rule_option = "allowed_no_destination";
  Attribute tessent_bscan_function = "select_jtag_input";
}
LogicSignal  select_jtag_input_int { instruction == INTEST; }
DataOutPort  select_jtag_output { Source select_jtag_output_int;
  Attribute connection_rule_option = "allowed_no_destination";
  Attribute tessent_bscan_function = "select_jtag_output";
}
LogicSignal  select_jtag_output_int {
  (instruction == EXTEST) ||
  (instruction == EXTEST_PULSE) ||

```

```
(instruction == EXTEST_TRAIN) ||
(instruction == CLAMP) ||
(instruction == HIGHZ) ;
}
DataOutPort    extest_pulse { Source ext_test_pulse_int;
  Attribute connection_rule_option = "allowed_no_destination";
  Attribute tesseract_bscan_function = "extest_pulse";
}
LogicSignal ext_test_pulse_int { instruction == EXTEST_PULSE; }
DataOutPort    extest_train { Source ext_test_train_int;
  Attribute connection_rule_option = "allowed_no_destination";
  Attribute tesseract_bscan_function = "extest_train";
}
LogicSignal ext_test_train_int { instruction == EXTEST_TRAIN; }
DataOutPort fsm_state[3:0]{
  Attribute connection_rule_option = "allowed_no_destination";
  Attribute function_modifier = "tap_fsm_state";
  RefEnum    state_encoding;
}

Enum state_encoding {
  test_logic_reset    = 4'b1111;
  run_test_idle       = 4'b1100;
  select_dr           = 4'b0111;
  capture_dr          = 4'b0110;
  shift_dr            = 4'b0010;
  exit1_dr            = 4'b0001;
  pause_dr            = 4'b0011;
  exit2_dr            = 4'b0000;
  update_dr           = 4'b0101;
  select_ir           = 4'b0100;
  capture_ir          = 4'b1110;
  shift_ir            = 4'b1010;
  exit1_ir            = 4'b1001;
  pause_ir            = 4'b1011;
  exit2_ir            = 4'b1000;
  update_ir           = 4'b1101;
}

ScanInterface tap_client {
  Port tdi;
  Port tdo;
  Port tms;
}

ScanInterface host_ijtag_1 {
  Port host_1_from_so;
  Port host_1_to_sel;
}

ScanInterface host_bscan {
  Port host_bscan_to_sel;
```

```
Port host_bscan_from_so;
Port capture_dr_en;
Port shift_dr_en;
Port update_dr_en;
Port test_logic_reset;
Attribute          tesseract_is_bscan_host = "true";
}
Instance fsm Of tap_fsm {
  InputPort tck = tck;
  InputPort tms = tms;
  InputPort trst = trst;
}
ScanRegister instruction[3:0] {
  CaptureSource 4'b0001;
  ResetValue    4'b1111;
  ScanInSource  tdi;
  RefEnum       instruction_opcodes;
}
Enum instruction_opcodes {
  BYPASS          = 4'b1111;
  CLAMP           = 4'b0000;
  EXTEST         = 4'b0001;
  EXTEST_PULSE   = 4'b0010;
  EXTEST_TRAIN   = 4'b0011;
  INTEST         = 4'b0100;
  SAMPLE         = 4'b0101;
  PRELOAD        = 4'b0101;
  HIGHZ          = 4'b0110;
  HOSTIJTAG_1    = 4'b0111;
}
ScanRegister bypass {
  CaptureSource 1'b0;
  ScanInSource  tdi;
}
ScanMux IRMux SelectedBy fsm.irSel {
  1'b0 : DRMux;
  1'b1 : instruction[0];
}
ScanMux DRMux SelectedBy instruction {
  4'b1111 : bypass;
  4'b0000 : bypass;
  4'b0001 : host_bscan_from_so;
  4'b0010 : host_bscan_from_so;
  4'b0011 : host_bscan_from_so;
  4'b0100 : host_bscan_from_so;
  4'b0101 : host_bscan_from_so;
  4'b0110 : bypass;
  4'b0111 : host_1_from_so;
  'bX     : bypass;
}
}
```

```

Attribute      keep_active_during_scan_test = "true";
Attribute      tessent_instruction_reg      = "instruction";
Attribute      tessent_bypass_reg          = "bypass";
//Attribute    tessent_instrument_container =
"chip_top_rtl1_ijtag";
//Attribute    tessent_use_in_dft_specification = "false";
//Attribute    tessent_instrument_type      =
"mentor::ijtag_node";
//Attribute    tessent_instrument_subtype  =
"tap_controller";
//Attribute    tessent_signature          =
"774f0efdef9d5b074af5f536d5077f57";
}
Module tap_fsm {
  TCKPort      tck;
  TMSPort      tms;
  TRSTPort     trst;
  ToIRSelectPort irSel;
  ToResetPort  tlr;
}

```

1.3 tap_t

```

set spec [create_dft_specification -existing_ijtag_host_scan_in
tap/host_1_from_so \
                                -sri_sib_list occ -tile_ijtag_host_list
{r1}]
report_config_data $spec

```

```

DftSpecification(tap_t,rtl1) {
  IjtagNetwork {
    HostScanInterface(ijtag) {
      Interface {
        design_instance : tap;
        scan_interface : host_ijtag_1;
      }
      Sib(sri) {
        Attributes {
          tessent_dft_function : scan_resource_instrument_host;
        }
        Sib(occ) {
        }
      }
      Sib(thc) {
        Attributes {
          tessent_dft_function : tile_host_collector;
        }
        Sib(th_r1) {
          to_scan_in_feedthrough : pipeline;
          SecondaryHostScanInterface(r1) {

```



```

        DataInPorts {
            connection(0) : tap_t/select_jtag_output;
            connection(1) : tap_t/select_jtag_input;
            connection(2) : tap_t/force_disable;
            connection(3) : tap_t/extest_train;
            connection(4) : tap_t/extest_pulse;
        }
        reset_value : 5'b00000;
    }
}
Sib(sri_ctrl) {
    Tdr(sri_ctrl) {
        Attributes {
            tesseract_dft_function : scan_resource_instrument_dft_control;
        }
    }
}
}
}
HostScanInterface(tap) {
    Interface {
        tck : TCK;
        trst : TRST;
        tms : TMS;
        tdi : TDI;
        tdo : TDO;
    }
}
HostScanInterface(bscan) {
    Interface {
        design_instance : tap_t;
        scan_interface : H0;
    }
}
}
BoundaryScan {
    ijtag_host_interface : HostScanInterface(bscan);
    BoundaryScanCellOptions {
        REF_CLK : cclock;
        INCLK : cclock;
    }
}
}
}
}

```

2. get_config_elements

```

get_config_elements [name_patterns] [-hierarchical] [-in_wrappers
wrapper_object_spec]
[-partition partition] [-count] [-type type] [-filter filter]

```

```
[ -regexp ] [ -nocase ] [ -silent ]
```

假设spec如下：

```
tmp(1) {
  ABC (Def, 2) {
    prop1 : 1;
    Prop2 : 2;
  }
  abc(def, 1) {
  }
  abcdef {
  }
}
```

Case 1

This matches the wrappers with leaf name “abc” inside wrapper tmp(1) independent of whether it has an id. The matching is case-insensitive because only the id matching is case-sensitive.

```
get_config_elements abc -in tmp(1)
```

```
{/tmp(1)/ABC(Def,2) /tmp(1)/abc(def,1)}
```

Case 2

This matches the wrappers with the first id equal to “Def”. Notice that it does not match abc(def,1) as ids are matched considering casing unless the -nocase option is used.

```
get_config_elements *(Def,*) -in tmp(1)
```

```
{/tmp(1)/ABC(Def,2)} get_config_elements *(Def,*) -in tmp(1) -nocase
```

```
{/tmp(1)/ABC(Def,2) /tmp(1)/abc(def,1)}
```

Case 3

This matches elements below the top-level wrapper starting with a*.

```
get_config_element */a*
```

```
{/tmp(1)/ABC(Def,2) /tmp(1)/abc(def,1) /tmp(1)/abcdef}}
```

Case 4

This matches elements below the top-level wrapper starting with a* and having two ids.

```
get_config_element */a*(*,*)
```

```
{/tmp(1)/ABC(Def,2) /tmp(1)/abc(def,1)}
```

Case 5

This matches elements starting with 'p' anywhere inside tmp(1).

```
get_config_element p* -in tmp(1) -hierarchical
```

```
{/tmp(1)/ABC(DeF,2)/prop1 /tmp(1)/ABC(DeF,2)/Prop2}
```

3. get_name_list

```
foreach pat [get_name_list [get_config_element Patterns -hierarchical]] {  
  report_config_data $pat;  
}
```