ijtag

1. 添加TDR

1.1 JTAG TDR

即直接由TAP IR选择的TDR□不是在ijtag SIB下面的。

```
IjtagNetwork {
  HostScanInterface(id) {
    Tap(id) {
      HostIjtag(TDR1) {
        instruction_codes : 4'b0001;
        instruction_name  : TDR1;
        Tdr(tdr1) { // TDR就是直接在IR指令选
          length: 8;
          extra_bits_capture_value: self;
        }
      }
      HostIjtag(TDR2) {
        instruction_codes : 4'b0010;
        instruction_name  : TDR2;
        Tdr(tdr2) { // TDR就是直接在IR指令选
          length: 8;
          extra_bits_capture_value: self;
        }
      }
    }
  }
}
```

1.2 iJTAG TDR

```
IjtagNetwork {
  HostScanInterface(id) {
    Tap(id) {
      HostIjtag(ijtag) {
        instruction_codes : 4'b0011;
        instruction_name  : IJTAG;
        Sib(tdr1) {
          Tdr(tdr1) {
            length: 8;
            extra_bits_capture_value: self;
          }
        }
        Sib(tdr2) {
```

```
          Tdr(tdr2) {
            length: 8;
            extra_bits_capture_value: self;
          }
        }
      }
    }
  }
}
```

2. SIB

2.1 STI

The Sib(sti) provides access to the IJTAG network for all instruments in the current design that are to be scan tested along with the rest of the functional logic by the logictest modes such as EDT or logicBIST. The acronym “sti” refers to “Scan Tested Instrument”. As shown in Figure 13-18, that Sib provides isolation between its host port and its clients.

简单地讲STI这种类型的SIB□里面可以插入scan chain, 这样可以把SIB周边的接口用SCAN测试到。

其中MBIST & MBISR使用的SIB□就是STI中的一种。

```
      Sib(sti) {
        Attributes {
          tessent_dft_function : scan_tested_instrument_host;
        }
        Sib(mbist) {
        }
      }
```

Figure 13-18. Sib(sti) With Scan Isolation Chain

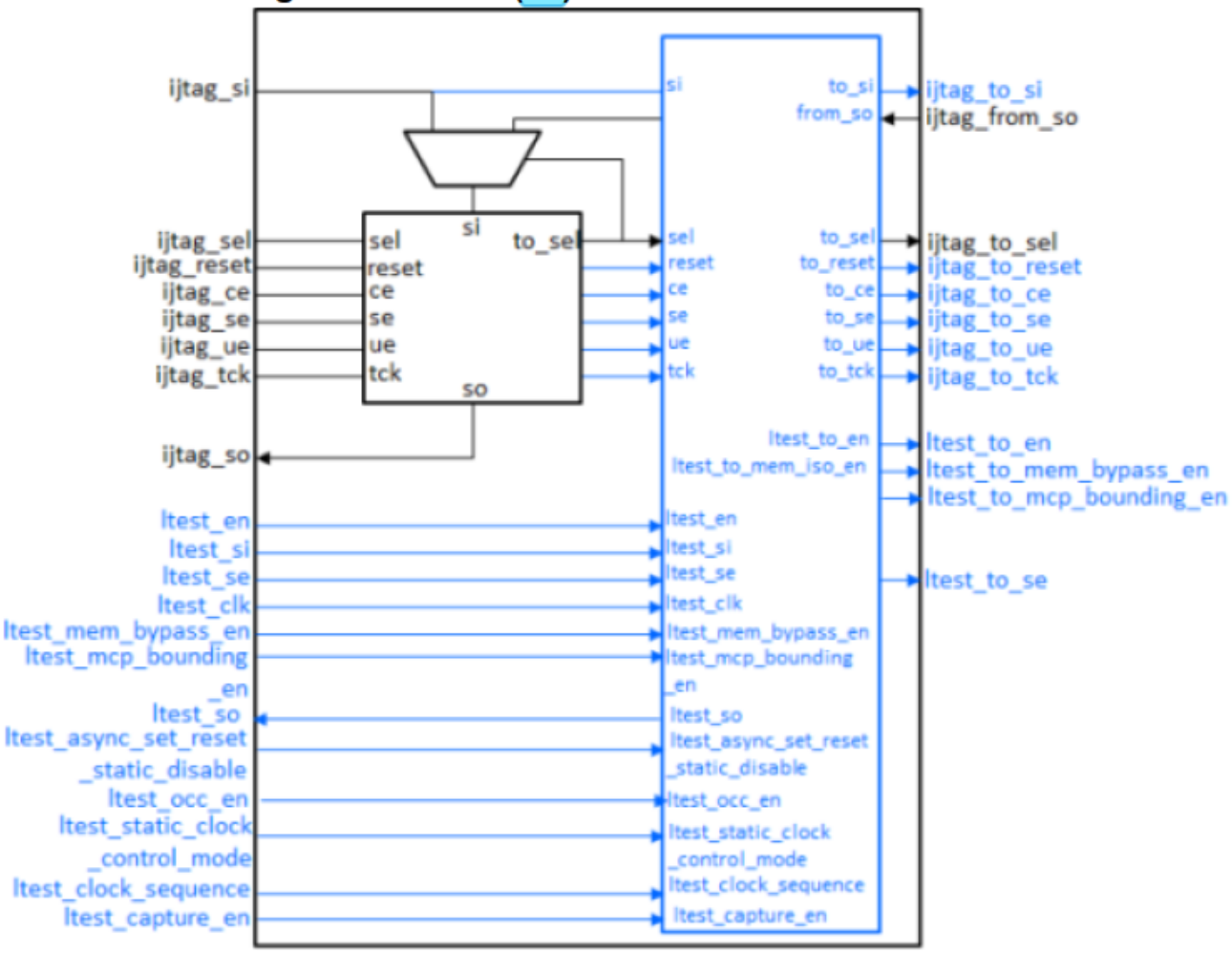
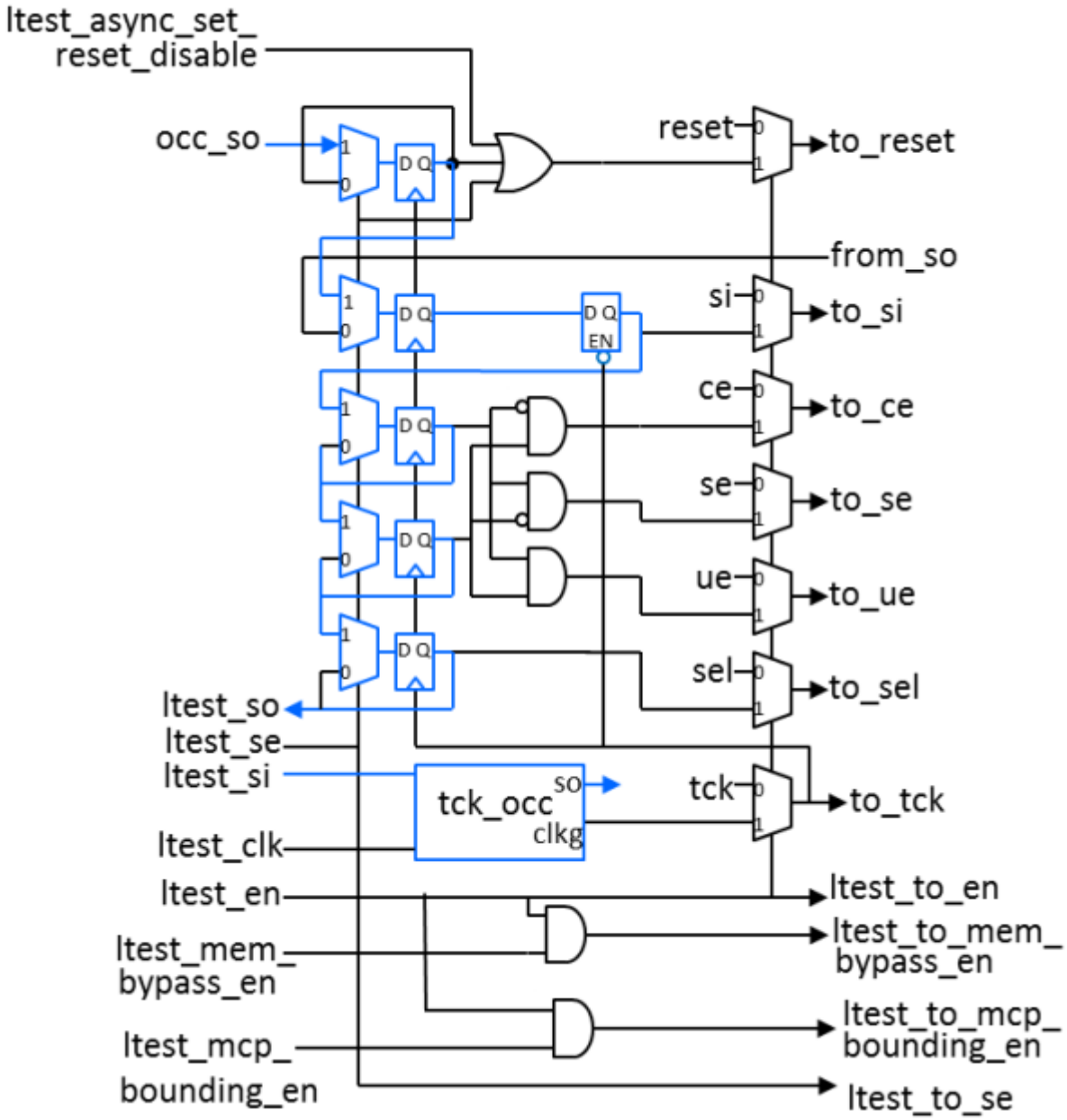


Figure 13-19. Added Scan Isolation Logic for SIB(sti)



2.2 SRI

SRI这种SIB就只是单纯的SIB， 在任何时候都可以用来配置TDR.

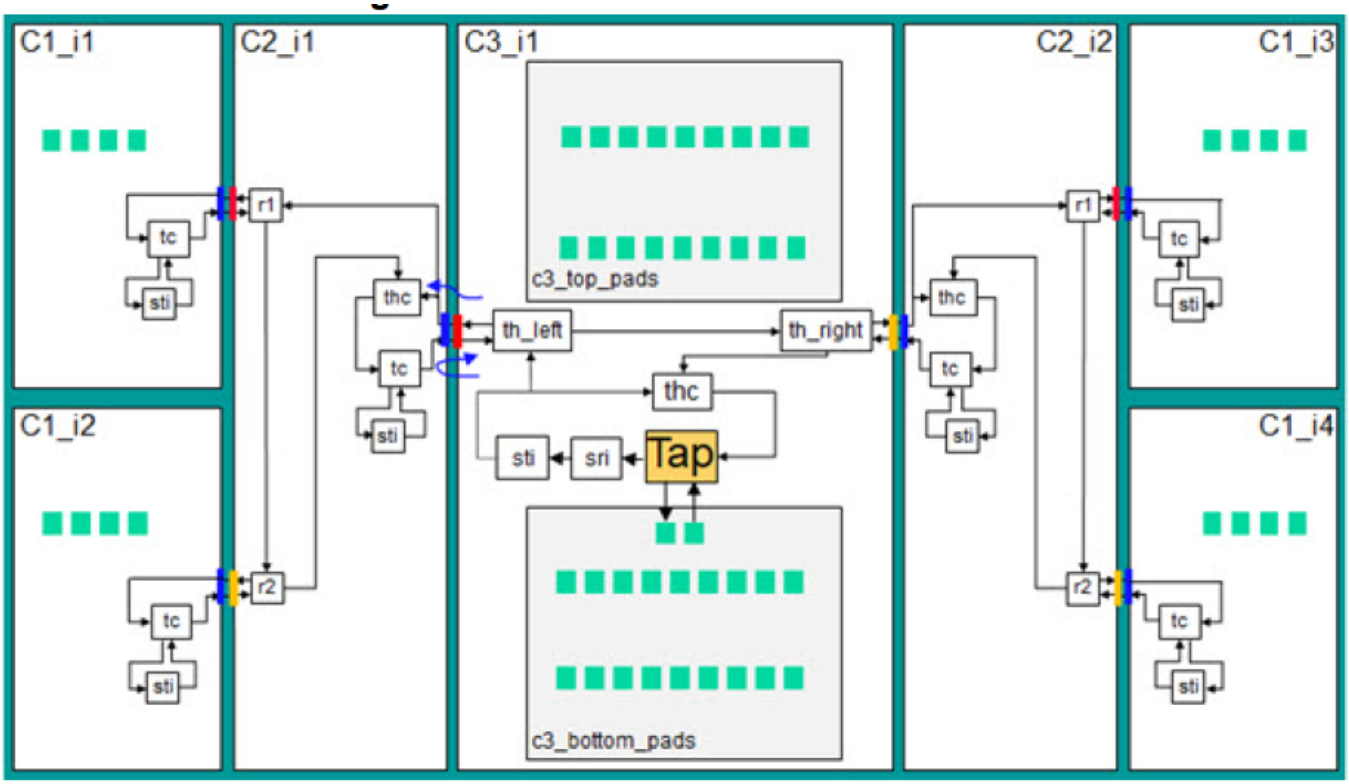
The Sib(sri) is used to provide access to the IJTAG network for child physical blocks, and for logictest instruments and Tdrs used to configure the logictest modes. The acronym “sri” refers to “Scan Resource Instrument”. These IJTAG nodes must remain active and undisturbed during the logictest modes, and are excluded from the logictest scan chains.

2.3 sri sti flow

用ijtag插sri组件，如tdr, icl等， 或sti组件时，为了spec有一个比较统一的hier path， 建议是用两次create spec的方式：

1. 先create sri sti spec
2. 再新的context design_id, 再create 分别的tdr等，在当前处hier path会有统一名字，方便脚本统一
    - HostScanInterface(sri)
    - HostScanInterface(sti)

3. tile flow




```
TC - Tile Client, 表示用于当前tile下面的jtag register SIB节点
THC - Tile Host Collector， 表示用于route到其它tile的router 总控制SIB节点
TH - Tile Host， 表示连接到其它tile的ijtag host控制SIB节点
STI - Scan Tested Instrument， 表示用于scan相关控制的SIB总控SIB节点，比如下面会挂载mbist mbisr等测试组件
SRI - Scan Resource Instrument, 用于连接其它的ijtag register的总控SIB节点，下面可以挂功能的各种SIB和ijtag register
```

```
set_dft_specification_requirements -design_type tile
```

```
set spec [create_dft_specification -tile_ijtag_host_list {r1 r2} -
sri_sib_list {occ edt} -sti_sib_list {mbist mbisr}]
report_config_data $spec
```

report config结果：

3.1 design_type is tile

SRI & STI是被包含在TC下面。

```
DftSpecification(a_t,rtl1) {
  IjtagNetwork {
    HostScanInterface(ijtag) {
      Sib(tc) {
        Attributes {
          tessent_dft_function : tile_client_sib;
        }
        to_scan_in_feedthrough : buffer;
        so_retiming : off;
        Sib(sti) {
          Attributes {
            tessent_dft_function : scan_tested_instrument_host;
          }
          Sib(mbist) {
          }
          Sib(mbisr) {
          }
        }
        Sib(sri) {
          Attributes {
            tessent_dft_function : scan_resource_instrument_host;
          }
          Sib(occ) {
          }
          Sib(edt) {
          }
        }
      }
      Sib(thc) {
        Attributes {
          tessent_dft_function : tile_host_collector;
        }
        Sib(th_r2) {
          to_scan_in_feedthrough : pipeline;
          SecondaryHostScanInterface(r2) {
          }
        }
        Sib(th_r1) {
          to_scan_in_feedthrough : pipeline;
          SecondaryHostScanInterface(r1) {
          }
        }
      }
    }
  }
}
```

3.2 design_type is standard

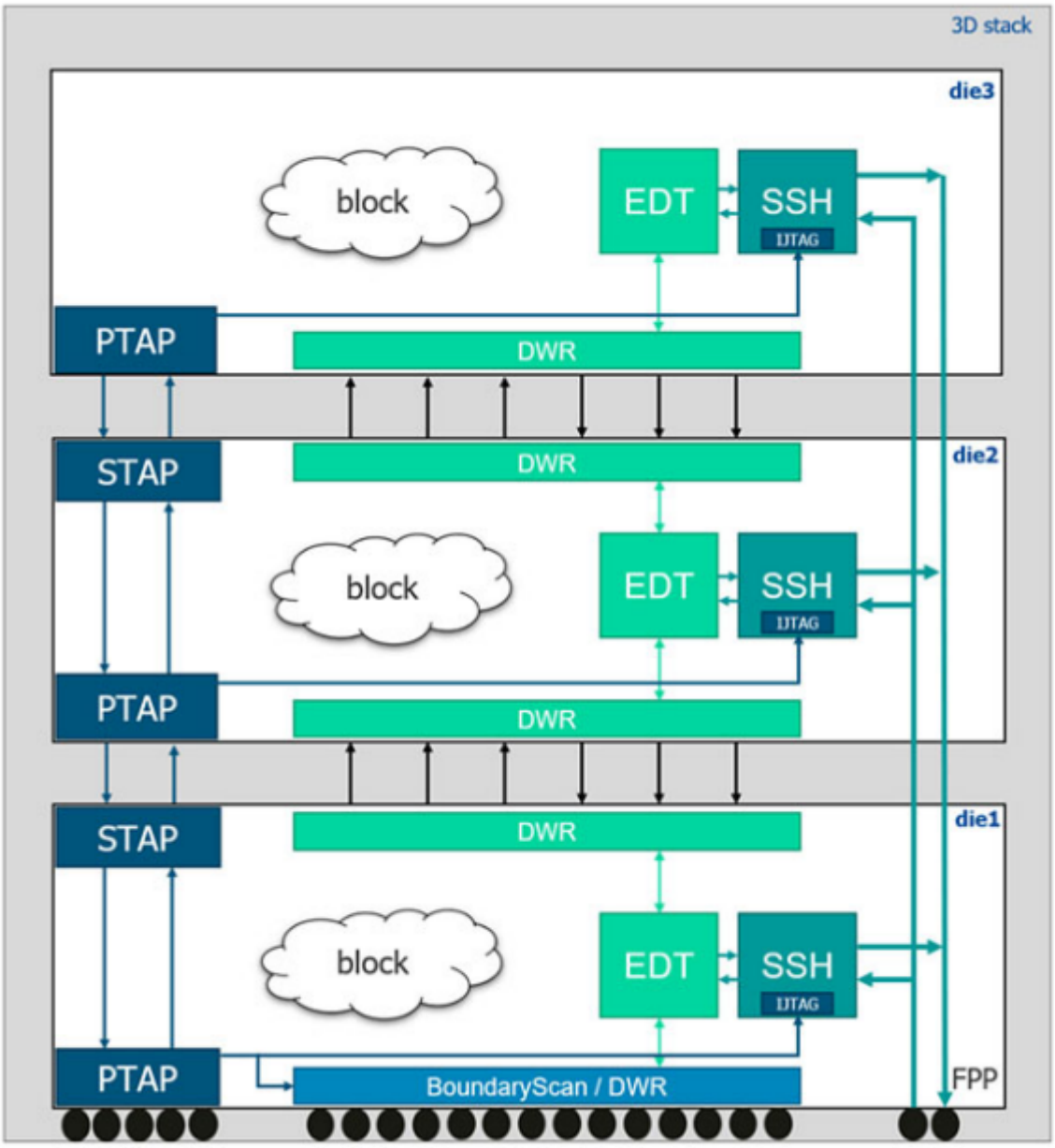
SRI & STI 与THC是平级结构，没有TC这个东西。

```
DftSpecification(a_t,rtl1) {
  IjtagNetwork {
    HostScanInterface(ijtag) {
      Sib(sti) {
        Attributes {
          tessent_dft_function : scan_tested_instrument_host;
        }
        Sib(mbist) {
        }
        Sib(mbisr) {
        }
      }
      Sib(sri) {
        Attributes {
          tessent_dft_function : scan_resource_instrument_host;
        }
        Sib(occ) {
        }
        Sib(edt) {
        }
      }
      Sib(thc) {
        Attributes {
          tessent_dft_function : tile_host_collector;
        }
        Sib(th_r2) {
          to_scan_in_feedthrough : pipeline;
          SecondaryHostScanInterface(r2) {
          }
        }
        Sib(th_r1) {
          to_scan_in_feedthrough : pipeline;
          SecondaryHostScanInterface(r1) {
          }
        }
      }
    }
  }
}
```

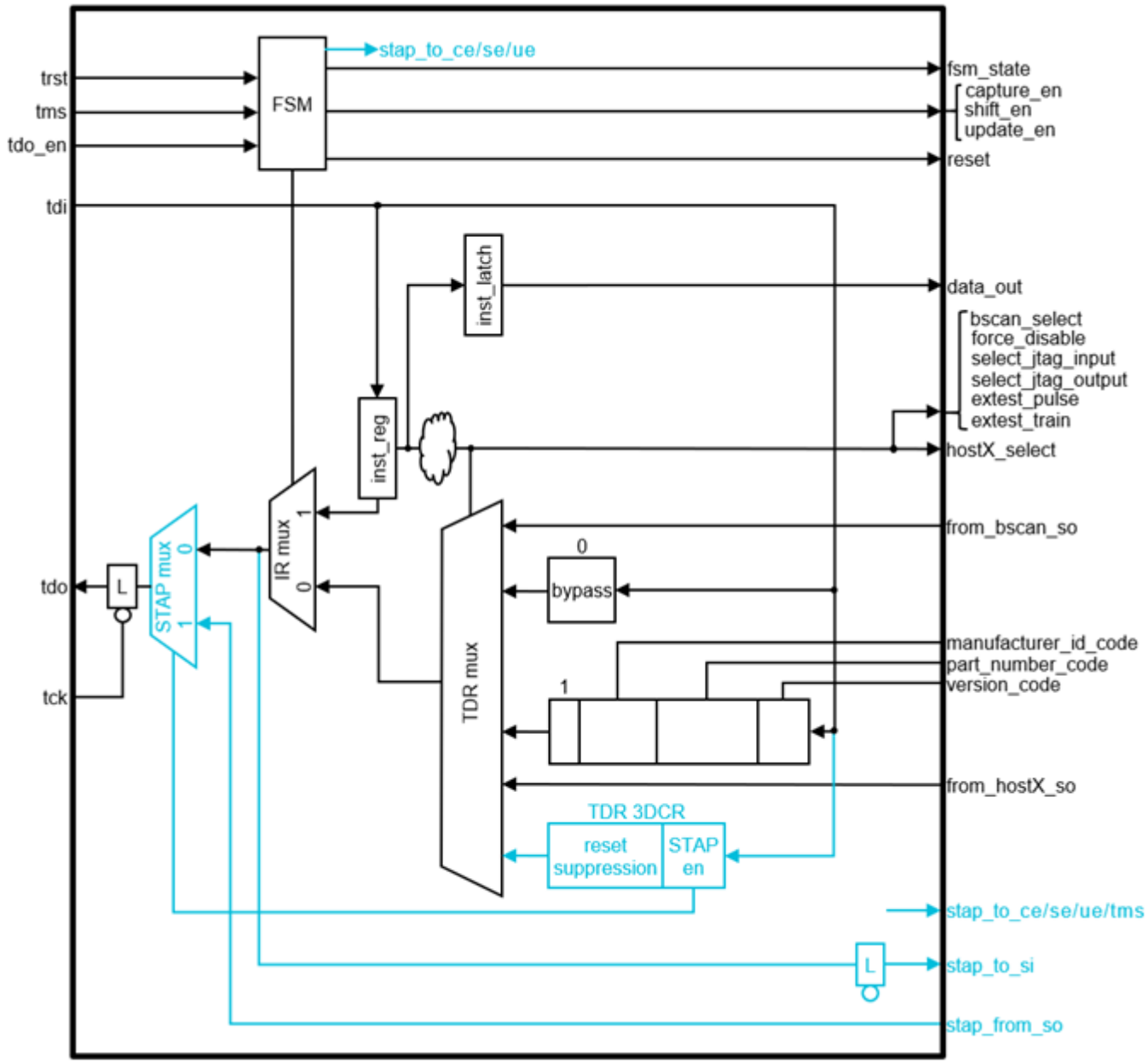
4. PTAP / STAP

PTAP/STAP参考1838协议，主要使用在3D封装项目上, 扩展出标准的5线JTAG接口，可以考虑用来接第三方的TAP□PHY/IP□□但注意这与标准1149.1不兼容。

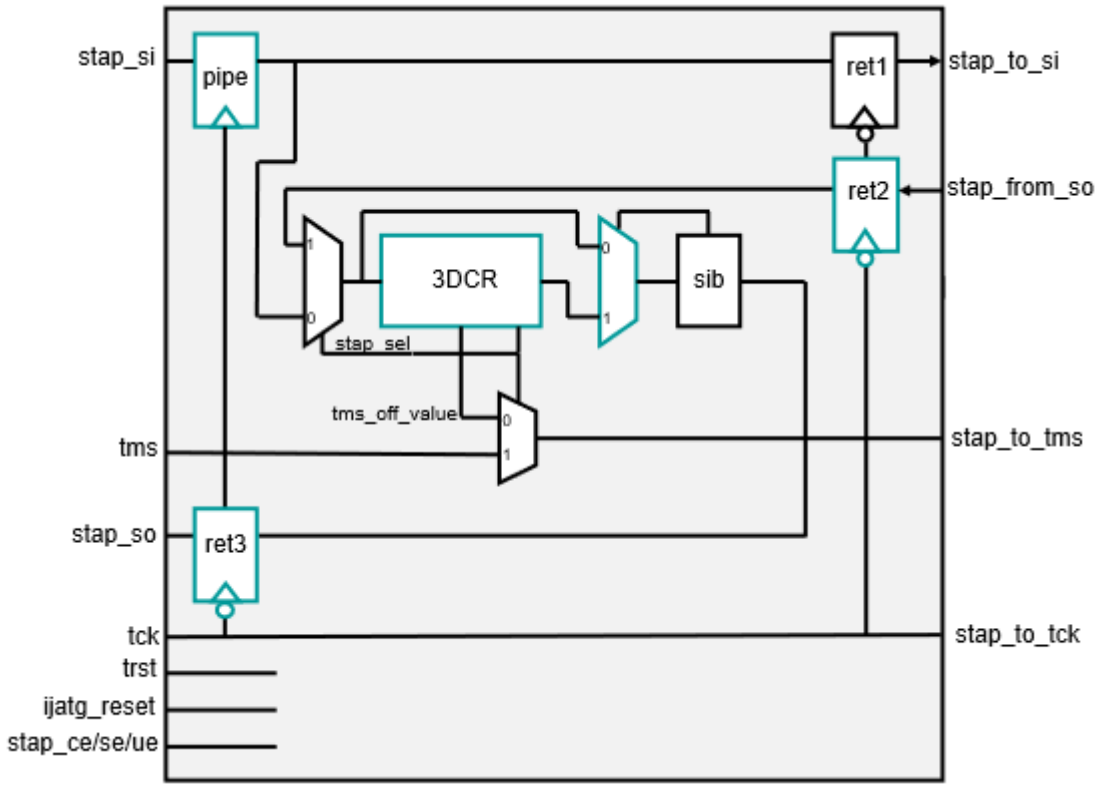
DIE内部用ijtag连接□3D DIE之间为了节省线接口一致使用PTAP/STAP连接□2D DIE之间使用标准1149.1串行连接。



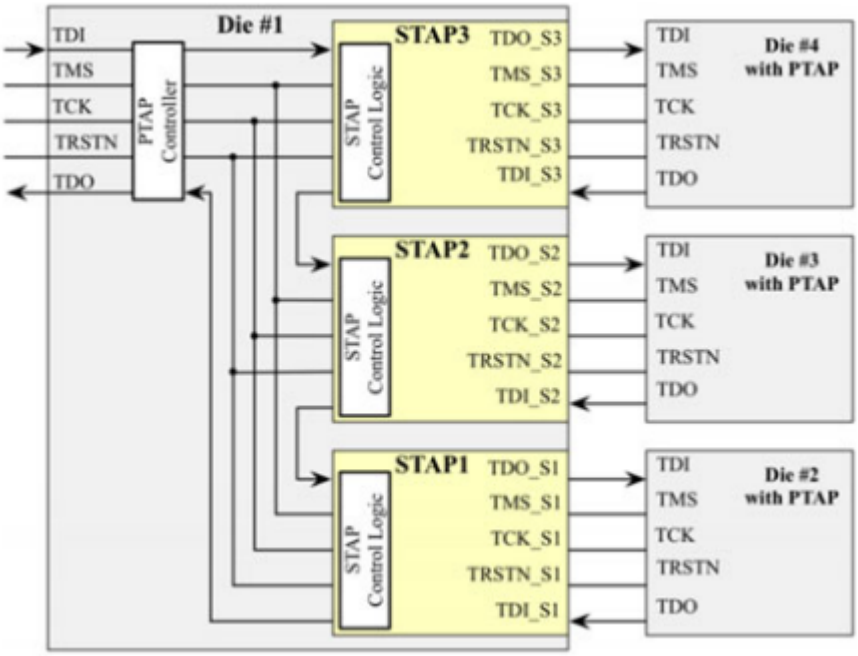
普通TAP改造为PTAP


trst
tms
tdo_en
tdi
tdo
tck
FSM
stap_to_ce/se/ue
fsm_state
capture_en
shift_en
update_en
reset
inst_latch
data_out
bscan_select
force_disable
select_jtag_input
select_jtag_output
extest_pulse
extest_train
inst_reg
hostX_select
IR mux
STAP mux
L
from_bscan_so
bypass
TDR mux
manufacturer_id_code
part_number_code
version_code
from_hostX_so
TDR 3DCR
reset suppression
STAP en
stap_to_ce/se/ue/tms
stap_to_si
stap_from_so

STAP框图



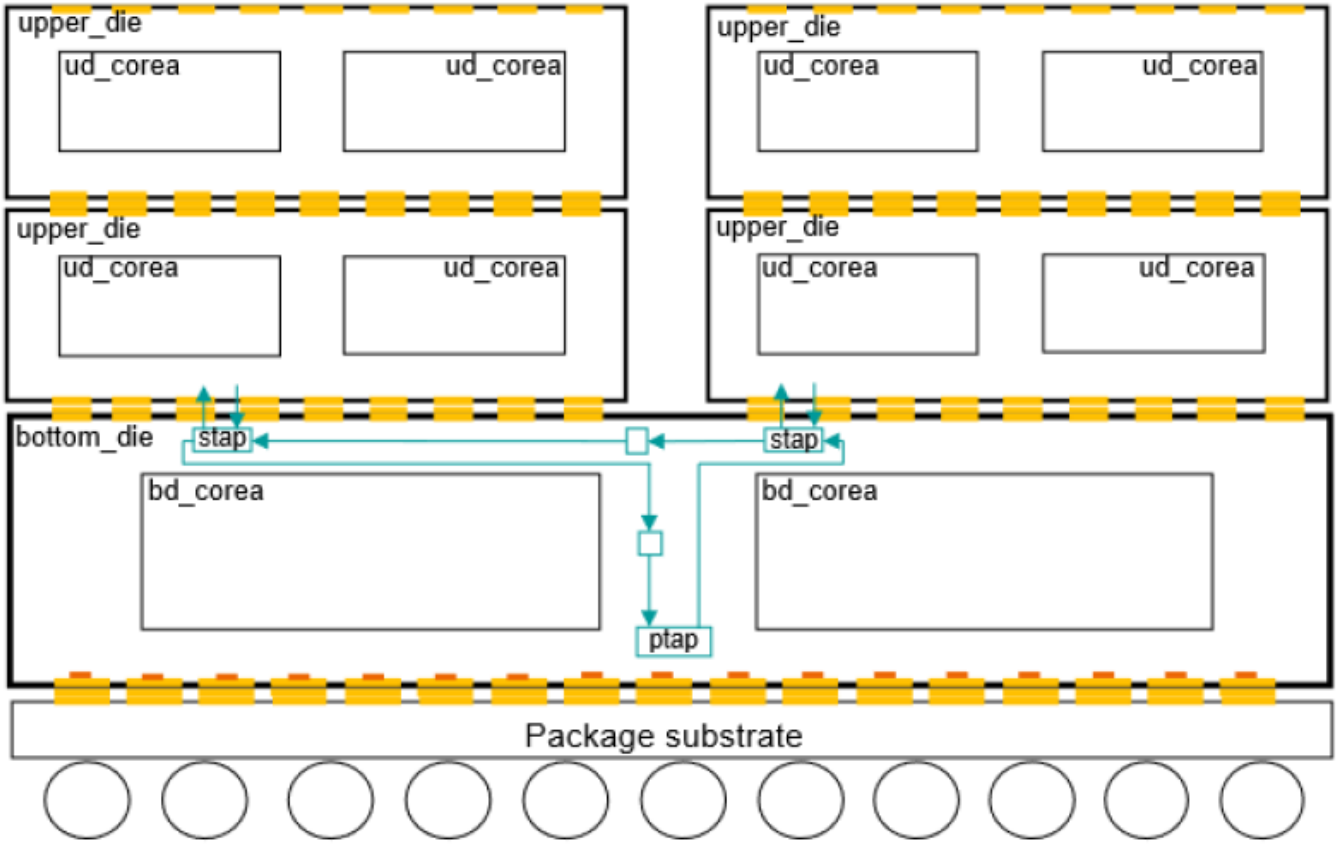
PTAP连3个STAP



4.1 example 1

一个PTAP接两个STAP

Figure 13-33. PTAP/STAP Network Inserted Within the Top Physical Blocks of a Multi-die Stack




```
## 从下往上看流向, si 到so的方向
Tap(main) {
  HostIjtag(1) {
   // Normal IJTAG network removed for clarity
  }
  HostBscan {
  }
  HostSTap { // 定义一个PTAP
    STapPipeline(end) {
    }
    STap(left) {     // 相当于实例化一个定义好的STAP进来
      SecondaryHostScanInterface(stap_left) {  // STAP连接到当前模块外面，扩展一个JTAG口
      }
    }
    STapPipeline(between) {
    }
    STap(right) {     // 相当于实例化一个定义好的STAP进来
      SecondaryHostScanInterface(stap_right) {  // STAP连接到当前模块外面，扩展一个JTAG口
      }
    }
  }
}
```

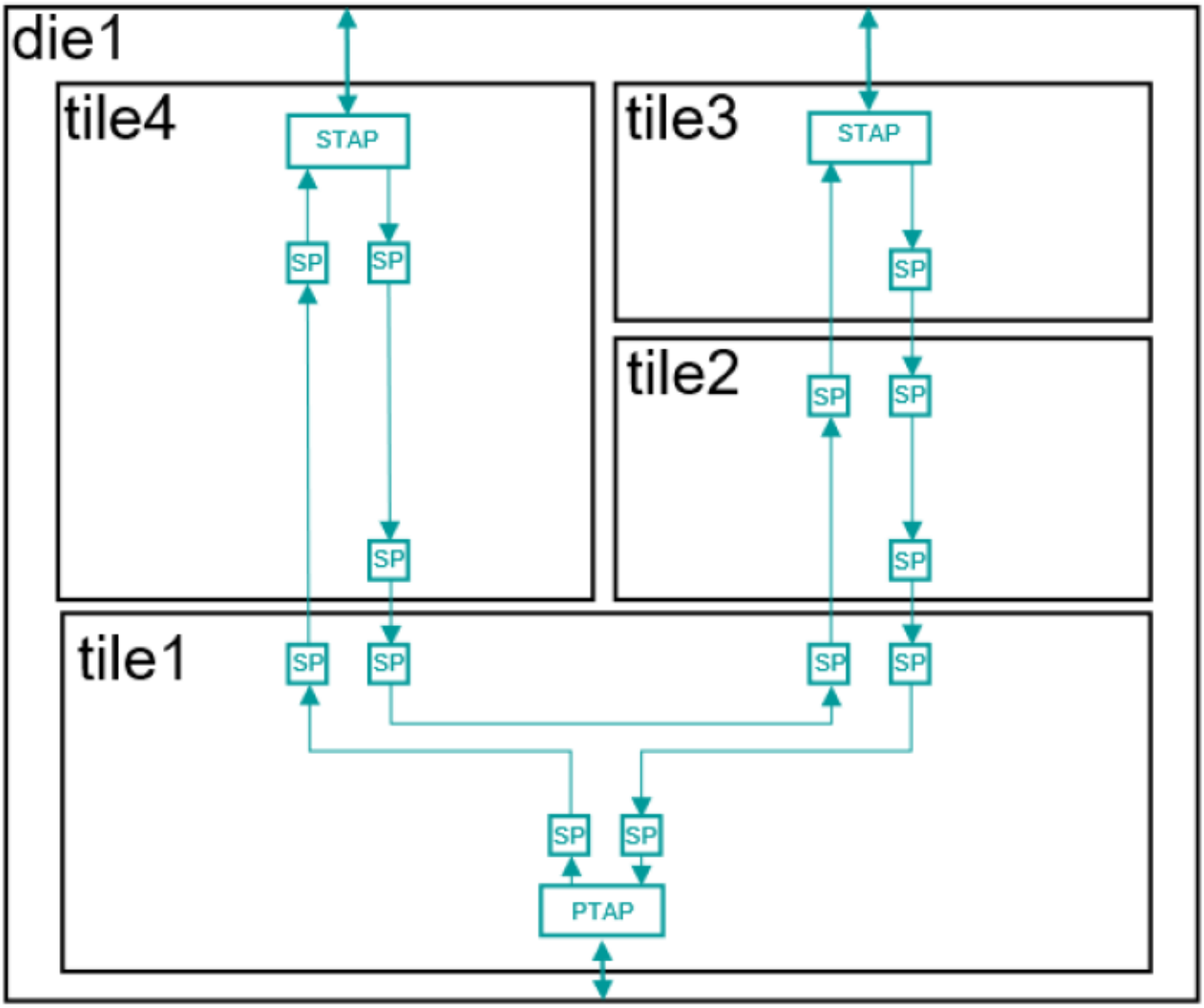
4.2 example 2

这个例子与上一个例子一样，都是一个PTAP接两个STAP接口，只不过PTAP与STAP在不同的tile，它们之间有连线穿过其它tile的情况，连线有pipeline结构。

因为STAP扩展的JTAG口是直接会连到PAD，应该是不能再添加pipeline结构，对于需要在芯片不同方向扩展JTAG口到芯片外的话，需要STAP放置在芯片的不同位置，STAP与PTAP之间的物理位置可能会是比较远，所以需要这样的分开的结构。

注，tile2, tile3, tile4里面只是描述是的STAP的连接结构，没有描述ijtag结构，可以再单独加一个HostScanInterface(ijtag)去描述。

Figure 13-34. PTAP/STAP Network Inserted into a Tiled Die



4.2.1 tile1

```
IjtagNetwork {
  HostScanInterface(tap) {
    Tap(main) {
      HostIjtag(1) {
```

```
          // Content removed for clarity
        }
        HostBScan {
        }
        HostSTap {
          STapPipeline(end) {
          }
          STapPipeline(from_right) {
          }
          SecondaryHostScanInterface(host_right_stap) {   // 开一个接口port□
外面可接一个STAP
            to_tck           : %s_to_tck;
          }
          STapPipeline(to_right) {
          }
          STapPipeline(from_left) {
          }
          SecondaryHostScanInterface(host_left_stap) {  //开一个接口port□外面
可接一个STAP
            to_tck           : %s_to_tck;
          }
          STapPipeline(to_left) {
          }

          STapPipeline(start) {
          }
        }
      }
    }
  }
```

4.2.2 tile2

```
  IjtagNetwork {
    HostScanInterface(stap) {
      Interface {
        tdi        : stap_tdi;
        tdo        : stap_tdo;
        tms        : stap_tms;
        tck        : stap_tck;
        trst       : stap_trst;
        reset      : stap_ijtag_reset;
        capture_en : stap_ijtag_ce;
        shift_en   : stap_ijtag_se;
        update_en  : stap_ijtag_ue;
      }
      STapPipeline(end) {
        so_retiming : off;
      }
      STapPipeline(from_host) {
```

```
    }
    SecondaryHostScanInterface(host_stap) {
      to_tck     : %s_to_tck;
    }
    STapPipeline(to_host) {
    }
  }
}
```

4.2.3 tile3

```
IjtagNetwork{
  HostScanInterface(stap) {
    Interface {
      tdi     : stap_tdi;
      tdo        : stap_tdo;
      tms        : stap_tms;
      tck        : stap_tck;
      trst       : stap_trst;
   reset      : stap_ijtag_reset;
      capture_en : stap_ijtag_ce;
      shift_en   : stap_ijtag_se;
      update_en  : stap_ijtag_ue;
    }
    STapPipeline(end) {
      so_retiming : off;
    }
    STap(host) {
      SecondaryHostScanInterface(stap_host) {
      }
    }
  }
}
```

4.2.4 tile4

```
IjtagNetwork {
  HostScanInterface(stap) {
    Interface {
      tdi        : stap_tdi;
      tdo        : stap_tdo;
      tms        : stap_tms;
      tck        : stap_tck;
      trst       : stap_trst;
      reset      : stap_ijtag_reset;
      capture_en : stap_ijtag_ce;
      shift_en   : stap_ijtag_se;
      update_en  : stap_ijtag_ue;
```

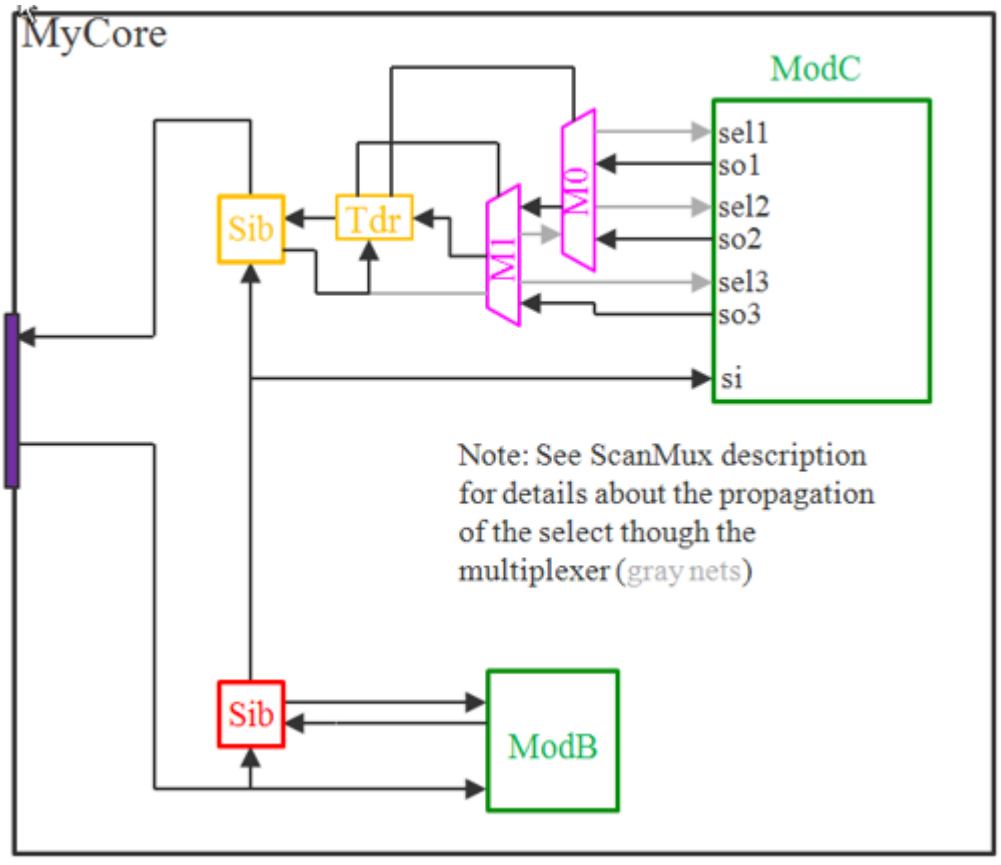
```
      }
      STapPipeline(end) {
        so_retiming : off;
      }
      STapPipeline(from_host) {
      }
      STap(host) {
        SecondaryHostScanInterface(stap_host) {
        }
      }
      STapPipeline(to_host) {
      }
    }
  }
```

5. DesignInstance

对于实例化已经存在ijtag接口的模块而言，需要用icl将其内部的结构描述出来。 有两种情况：

- 模块有多个scan_interface的， 理解为有多个平行的TDR
- 模块只有一个scan_interface的，理解为只有一个TDR

如下图所示：




```
DftSpecification(MyCore,rtl) {
  IjtagNetwork {
    HostScanInterface(purple) {
```

```
      Sib(orange) {
        Tdr(orange) {
          DataOutPorts {
            Count : 2 ;
          }
        }
        ScanMux(M1) {
          Select : tdr(orange)/DataOut(1) ;
          Input(0) {
            ScanMux(M0) {
              Select : tdr(orange)/DataOut(0) ;
              Input(0) {
                DesignInstance (u3/top_green) {
                  scan_interface : I1 ;
                }
              }
              Input(1) {
                DesignInstance (u3/top_green) {
                  scan_interface : I2 ;
                }
              }
            }
          }
          Input(1) {
            DesignInstance (u3/top_green) {
              scan_interface : I3 ;
            }
          }
        }
      }
      Sib(red) {
        DesignInstance (u1/bottom_green) {}
      }
    }
  }
}
```

如果在环境中，读取了预先写好的icl文件，那么工具将会自动地把这个模块给连接到ijtag network上去，但是结构会比较乱一些。

最好是自己去描述ijtag连接结构，不用工具自动生成的。

6. reset

ijtag tdr里面reset是用test-logic-reset来作为reset信号, -- 对应ICL ResetPort ToResetPort

而不是使用trst_n来作为reset这个是TAP的reset -- 对应ICL TRSTPort ToTRSTPort

```
DftSpecification(module_name,id) {
```

```
  IjtagNetwork {
    HostScanInterface(id) {
      Tdr(id) {
        so_retiming                  : on | off ;
        DataInPorts  {
        }
        DataOutPorts {
        }
        Interface  {
        }
        parent_instance              : instance_name ;
        leaf_instance_name           : leaf_instance_name ;
        keep_active_during_scan_test : on | off | auto ;
        length                       : integer ;      // default: auto
        extra_bits_capture_value     : 0 | 1 | self ;
        reset_value                  : binary ;       // default: auto
        Attributes {
          attribute_name             : attribute_value ;
        }
        DecodedSignal(signal_name) {
        }
      }
    }
  }
}
```

7. get_icl_objects

```
#获取指定icl module的scan interface列表
get_icl_objects -object_types icl_scan_interface_of_module -of_objects
[get_icl_modules $module_name]

#获取指定icl instance的scan interface列表
get_icl_objects -object_types icl_scan_interface -of_objects $instance

#获取指定icl instance下面的icl tdr register
get_icl_objects *tdr* -object_types icl_scan_register -below_instances
icl_instance_spec
```

8. 多TAP组织结构

8.1 通过单独port选择TAP

文档 example 2

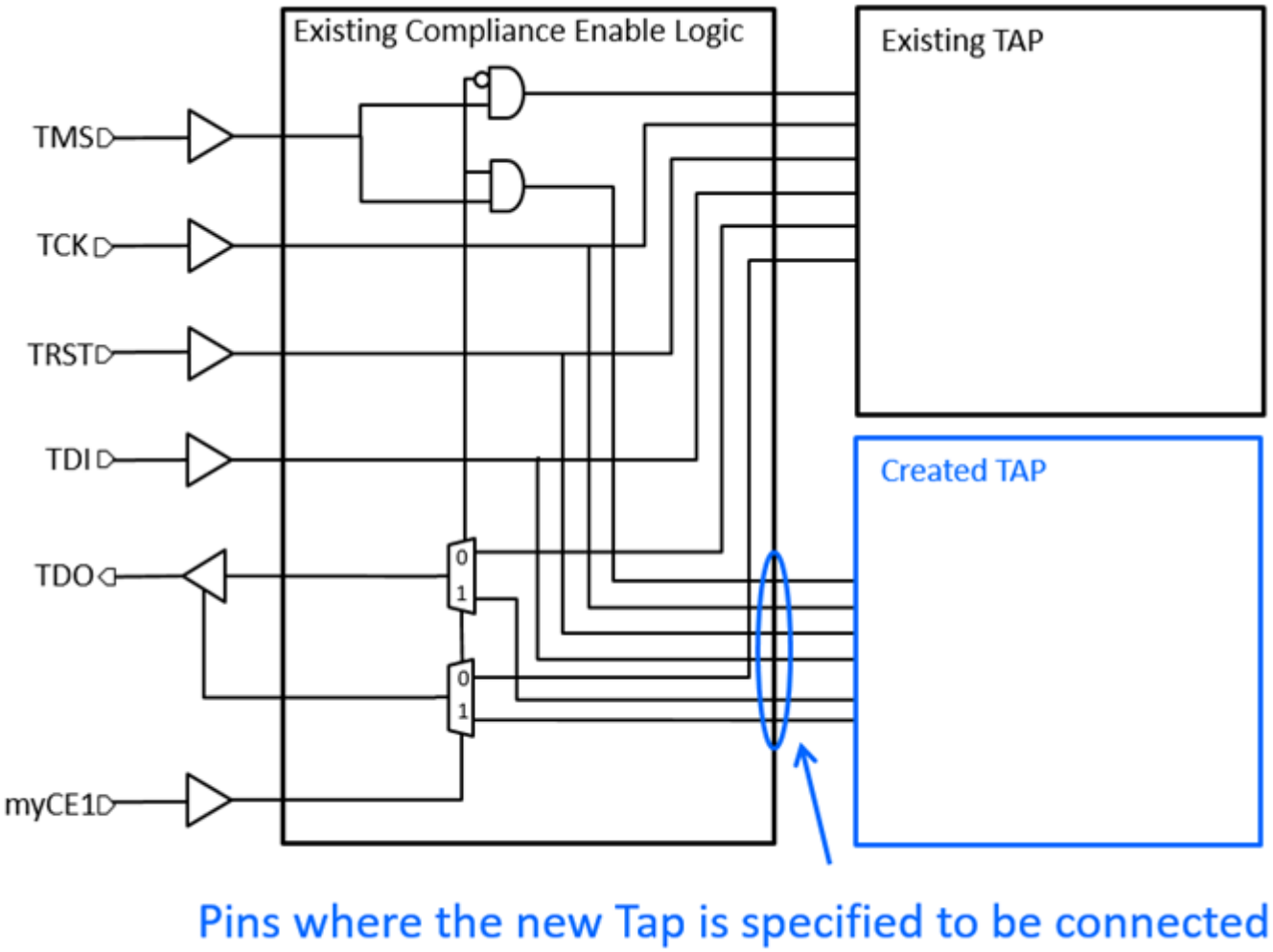
下面这个例子是已经在代码里面存在了一个TAP(比如CPU JTAG)，但是需要做bscan逻辑，就需要另外单独加一个TAP

两个TAP之间需要切换，是通过一个单独的管脚进行切换，需要设置是1切换到new TAP,还是0切换到new TAP

```
#0 切换到new tap
create_dft_specification -active_low_compliance_enables;

#1 切换到new tap
create_dft_specification -active_high_compliance_enables;
```

example采用myCE1==1切换到new TAP, new TAP的连接点是internal pin




```
# step 1: set the 'function' attribute on the top level TAP ports
set_attribute_value -name function -value trst jtag_trst
set_attribute_value -name function -value tdi  jtag_tdi
set_attribute_value -name function -value tms  jtag_tms
set_attribute_value -name function -value tck  jtag_tck
set_attribute_value -name function -value tdo  jtag_tdo

# Step 1b: declare your compliance enable port so it is documented in
#          the BSDL file
set_boundary_scan_port_options myCE1 -cell_options compliance_enable1
# Step 2: create an initial DFT specification
set_system_mode analysis
```

```
set spec [create_dft_specification]
# Step 3: add new HostScanInterface with the internal hookup points
read_config_data -in_wrapper ${spec}/IjtagNetwork -from_string {
   HostScanInterface(tap_internal) {
      Interface {
        tck :              jtag_and0/o ;
        trst :             jtag_and3/o ;
        tms :              jtag_and1/o ;
        tdi :              jtag_and2/o ;
        tdo :              jtag_buf5/a ;
        tdo_en :           jtag_buf4/a ;
        tdo_en_polarity : active_high ;
      }
   }
}
# step 4: move the Tap to the new HostScanInterface
move_config_element ${spec}/IjtagNetwork/HostScanInterface(tap)/Tap(main) \
       -in_wrapper ${spec}/IjtagNetwork/HostScanInterface(tap_internal)

# step 5: report the final DftSpecification.
report_config_data $spec
The resulting DFT specification looks like:
DftSpecification(top,rtl) {
  IjtagNetwork {
    HostScanInterface(tap) {
      Interface {
        tck : jtag_tck ;
        trst : jtag_trst ;
        tms : jtag_tms ;
        tdi : jtag_tdi ;
        tdo : jtag_tdo ;
      }
    }
    HostScanInterface(tap_internal) {
      Interface {
        tck : jtag_and0/o ;
        trst : jtag_and3/o ;
        tms : jtag_and1/o ;
        tdi : jtag_and2/o ;
        tdo : jtag_buf5/a ;
        tdo_en : jtag_buf4/a ;
        tdo_en_polarity : active_high ;
      }
      Tap(main) {
        HostIjtag(1) {
        }
        HostBscan {
        }
      }
    }
  }
```

```
  BoundaryScan {
    ijtag_host_interface : Tap(main)/HostBscan ;
    pin_order_file : ../data/top.pinorder ;
    BoundaryScanCellOptions {
      myCE1 : compliance_enable1 ;
    }
  }
}
```

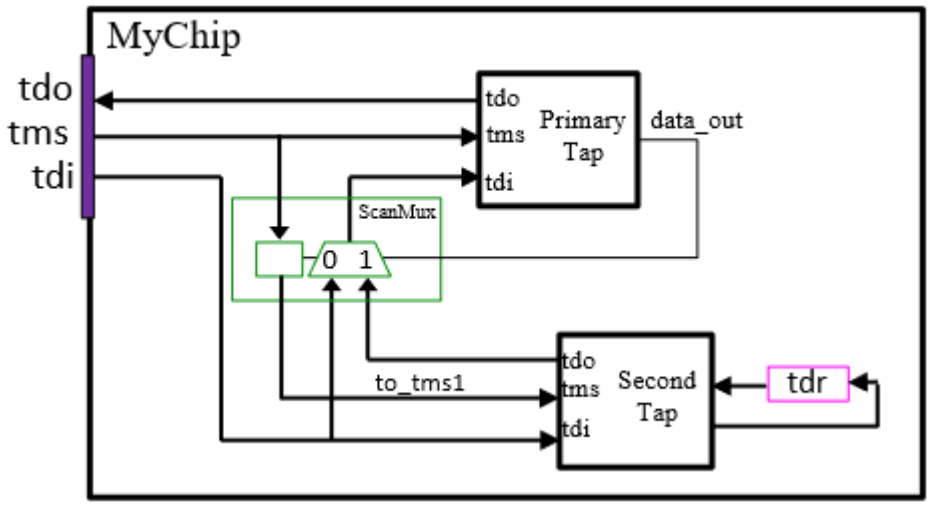
对应已存在的切换逻辑模块ICL描述如下

```
Module compliance_enable {
   // from the pads
   TCKPort     tck ;
   ScanInPort  tdi ;
   ScanOutPort tdo {
      Source ComplianceOutputSelect;
      Attribute forced_high_output_port_list = "tdo_en" ;
   }
   DataOutPort tdo_en {
      Attribute associated_scan_port_list = "tdo";
      Attribute connection_rule_option = "allowed_no_destination";
      Attribute function_modifier = "tdo_enable_active_high";
   }
   TMSPort     tms ;
   TRSTPort    trst {
      Attribute connection_rule_option = "allowed_tied_high" ;
   }
   DataInPort ce ;
   DataInPort tdo_en_other {
      Attribute associated_scan_port_list = "tdo_other";
      Attribute connection_rule_option = "allowed_no_source";
      Attribute function_modifier = "tdo_enable_active_low";
   }
   DataInPort tdo_en_ts {
      Attribute associated_scan_port_list = "tdo_ts";
      Attribute connection_rule_option = "allowed_no_source";
      Attribute function_modifier = "tdo_enable_active_low";
   }
   ScanInterface tap_pins {
      Port tdi ;
      Port tdo ;
      Port tms ;
      Port tck ;
      Port trst ;
   }
   // to the existing TAP
   ToTCKPort   tck_other ;
   ToTMSPort   tms_other ;
   ToTRSTPort  trst_other ;
   ScanInPort  tdo_other ;
```

```
   ScanOutPort tdi_other {
      Source tdi ;
   }
   ScanInterface tap_pins_other {
      Port tdi_other ;
      Port tdo_other ;
      Port tms_other ;
      Port tck_other ;
      Port trst_other ;
   }
   // to the Tessent Shell generated TAP
   ToTCKPort   tck_ts ;
   ToTMSPort   tms_ts ;
   ToTRSTPort  trst_ts ;
   ScanInPort  tdo_ts ;
   ScanOutPort tdi_ts {
      Source tdi ;
   }
   ScanInterface tap_pins_ts {
      Port tdi_ts ;
      Port tdo_ts ;
      Port tms_ts ;
      Port tck_ts ;
      Port trst_ts ;
   }
   // the mux for the tdo
   ScanMux ComplianceOutputSelect SelectedBy ce {
     1'b1 : tdo_ts ;
     1'b0 : tdo_other ;
   }
}
```

8.2 串行选择结构

这种结构比较好，与最新的PTAP-STAP结构可以考虑下用哪种比较好。

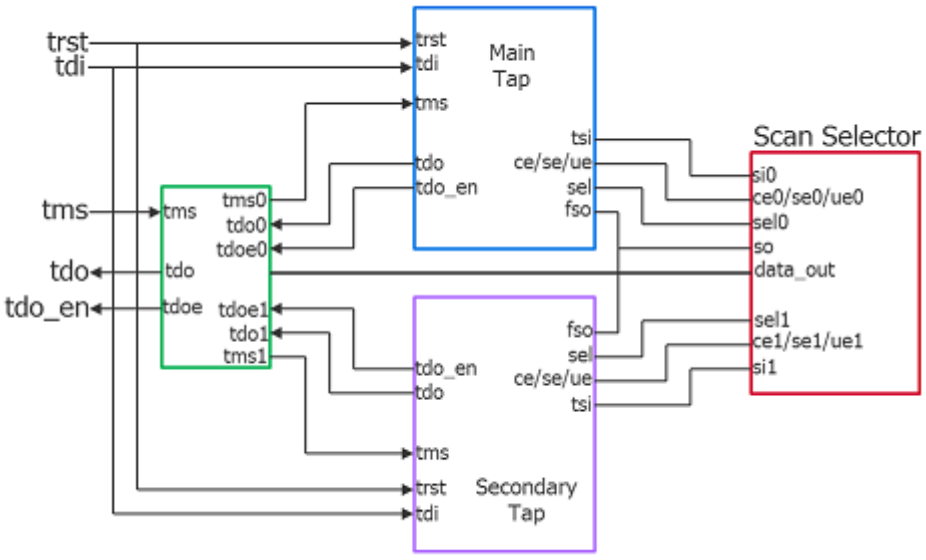


由其中一个TAP进行选择。

```
IjtagNetwork {
  HostScanInterface(purple) {
    Tap(main) {
      DataOutPorts {
        count : 1 ;
      }
    }
    ScanMux(green) {
      select : Tap(main)/DataOut(0) ;
      Input(1) {
        Tap(secondary) {
          HostIjtag(0) {
            Tdr(pink) {
            }
          }
        }
      }
    }
  }
}
```

8.3 并行选择结构

这种结构个人认为不太好。



由每个TAP都能访问的一个共公模块进行选择。

```
IjtagNetwork {
  HostScanInterface(tap) {
    Interface {
      tck : tck ;
      tdi : tdi ;
      tdo : tdo ;
```

```
        trst : trst ;
        tms : tms ;
      }
      ScanMux(tap_mux) {
        select : DesignInstance(scan_selector)/data_out ;
        Input(0) {
          Tap(main) {
            HostIjtag(0) {
              DesignInstance(scan_selector) {
                scan_interface   : Int0 ;
              }
            }
            HostIjtag(1) {
              tdr(1) {
                DataOutPorts {
                  count : 8 ;
                }
              }
            }
          }
        }
        Input(1) {
          Tap(secondary) {
            HostIjtag(0) {
              DesignInstance(scan_selector) {
                scan_interface   : Int1 ;
              }
            }
            HostIjtag(1) {
              tdr(2) {
                DataOutPorts {
                  count : 8 ;
                }
              }
            }
          }
        }
      }
    }
  }
```

```
Module scan_selector {
  TCKPort      tck;
  SelectPort   sel1;
  ScanInPort   si1;
  ShiftEnPort  se1;
  UpdateEnPort ue1;
  SelectPort   sel0;
  ScanInPort   si0;
  ShiftEnPort  se0;
  UpdateEnPort ue0;
```

```
  ScanOutPort  so1 { Source tdr; }
  ScanOutPort  so0 { Source tdr; }
  DataOutPort data_out { Source tdr; }
  ScanInterface Int1 { Port tck; Port sel1; Port si1;
                       Port se1; Port ue1; Port so1;}
  ScanInterface Int0 { Port tck;  Port sel0; Port si0;
                       Port se0; Port ue0; Port so0;}
  ScanMux M1 SelectedBy tdr {
    1'b0 : si0;
    1'b1 : si1;
  }
  ScanRegister tdr {
    ScanInSource M1;
    ResetValue 1'b0;
  }
}
```