DFA(Design Flow Automation) Release Notes

发布日期版本：2025/07/07

概述

本次DFA版本升级实现了设计流程自动化的全面增强，主要包含以下核心改进：

1. **标准化文件格式**
   - 规范了ticket文件(.stf)和definition文件(.dfd)的语法结构，支持参数引用、通配符匹配和追加赋值等高级特性
2. **深度系统集成**
   - 新增自动化DQI质量检测框架（报告抽取+状态判断）
   - 强化DVC版本控制集成（设计数据链接+自动化提交）
   - 优化Slurm任务调度支持（集群/本地模式）

功能更新说明

1. standard ticket format(.stf)文件格式规范化

```
[HEADER]
## DESCRIPTION : ticket name
## ARGUMENT    : < TXXX_XXX >, ex. T510_RCX
TICKET   = flow ticket name

## DESCRIPTION : flow name & run directory
## ARGUMENT    : < flow_name:run_dir_name >, ex. 510_RCX:rcxt_spef
FLOW_ID =  flow_reference_id:ticket_run_dir

## DESCRIPTION : techlib confilg file from TLM
## ARGUMENT    : < techlib config file name >, ex.dfalib.cfg
TECHLIB =  techlib_config_file

## DESCRIPTION : source path from DVC
## ARGUMENT    : < source path >, ex.phase/block/stage/v1
DVC_SRC =  design_source_version_path

## DESCRIPTION : destination path from DVC
## ARGUMENT    : < destination path >, ex.phase/block/stage/v2
DVC_DST =  design_dest_version_path

## DESCRIPTION : top module name
## ARGUMENT    : < top module name >, ex.design
DESIGN  =  top_module_name

[INPUT]
## DESCRIPTION : input file of this flow
## ARGUMENT    : < path | file_name  >, ex.NETLIST_FILE = netlist.v
input_ref_id1  = input_file_name
input_ref_id2  = input_dir_name
...

[OUTPUT]
## DESCRIPTION : output file of this flow
## ARGUMENT    : < path | file_name  >, ex.NETLIST_FILE = netlist.v
output_ref_id1 = output_file_name
output_ref_id2 = output_dir_name
```

```
...

[PARAMETER]
## DESCRIPTION : Script path of EDA tool
## ARGUMENT    : <eda_tool_path_script>, ex. encounter
TOOL = encounter

## DESCRIPTION : Set tool option
## ARGUMENT    : <eda_tool_options> ex. tool version, ...
TOOL_OPTION = -uv 9.10-e014_1 -u64

## DESCRIPTION : Specify queue name
## ARGUMENT    : [ Local | FE | BE ]
## if QUEUE = Local, only run job in local machine
QUEUE = Local

## DESCRIPTION: Set queue option
## ARGMENT    : <queue_options>   #ex. slurm's options
QUEUE_OPTION = --mem 64G -c 1    #slurm's options

## DESCRIPTION : PARAMETER of this flow
## ARGUMENT    : < path | parameter >, ex.rc_corner = Typ_85c
parameter_id1  = parameter_value1
parameter_id2  = parameter_value2
...
```

此外，该文件支持以下语法

- 此文件参数中可以直接引用其他参数的值，格式是“`${参数键名}`”。
- 在[INPUT] [OUTPUT] [PARAMETER]这三个字段中的参数支持以 `<step_dir_name>/input_id = 文件名` 这样的形式将参数指定给某子步骤，其中支持使用 `“ * ”` 对 `<step_dir>` 去进行匹配。
- 此文件参数支持以 `” \ “` 或 `“ref_id += xxx"` 的形式对参数进行追加赋值。

2. **flow step definition(.dfd)文件格式规范化**

```
###  单一STEP的dfd文件格式如下：  ###
DEFINE STEP <step_name>
INPUT  input_ref_id1 = input_file;
INPUT  input_ref_id1 = input_file;
OUTPUT output_ref_id = output_file;
OUTPUT output_ref_id = output_file;
PARAM  param_ref_id = default_value;
PARAM TOOL = encounter
PARAM TOOL_OPTION = -uv 9.10-e014_1 -u64
PARAM QUEUE = Local
PARAM QUEUE_OPTION = --mem 64 -core 1

PRE_CHECK pre_check_script
EXEC  $eda_cmd ....
POST_CHECK post_check_script
EXEC_DQI exec_dqi.script
END STEP

###  多个STEP或SUBFLOW的dfd文件格式如下：  ###
DEFINE FLOW <flow_name>
INPUT input_ref_id1 = input_file;
INPUT input_ref_id2 = input_file;
OUTPUT output_ref_id1 = output_file;
OUTPUT output_ref_id2 = output_file;
PARAM param_name = default_value;
```

```
STEP step_name step_id
> step_input = $input_file
< step_output = $output_file
+ param_name = $param_value
;
SUBFLOW flow_name flow_id
> step-input = $input_file
< step_output = $output_file
+ param_name = $param_value
;
END FLOW
```

此外，该文件支持以下语法

- 此文件参数中可以直接引用其他参数的值，格式是“`${参数键名}`”
- STEP / SUBFLOW部分语法说明如下
  - `step_name/flow_name`：此项内容为子流程的运行目录的名称，支持多层目录。ex.`01_RCX/cbest_125`
  - `step_id/flow_id`：此项为子流程对应的FLOW_ID
  - `> < +`：输入输出映射（`>`表示输入，`<`表示输出）以及参数传递（+表示参数）。

3. **Slurm集成优化**

通过在ticket中对QUEUE以及QUEUE_OPTION的设置实现与slurm的集成优化

- 支持QUEUE = < FE | BE >进行slurm队列调度，也支持QUEUE = Local进行本地运行
- 支持QUEUE_OPTION自定义slurm参数

4. **DQI集成优化**

新增自动化DQI报告抽取功能，此功能通过对以下文件的设置实现

- `dqi_ext.spec`：定义如何从EDA报告中抽取关键指标

```
# <category>/<DQI_ID>\tReport_path\tSimple_regex\tOnly_extract_first\tDefault_value\tComplex_match_script
361-SCAN_DC/SCAN_DC_INSERT_0001 REPORT/$RPT /^Total number of test points  : (\d+)/ true    0   none
361-SCAN_DC/SCAN_DC_INSERT_0002 REPORT/$RPT /^Number of Autofix test points: (\d+)/ false   0   none
```

字段说明

序号	字段名	描述
1	`<category>/<DQI_ID>`	类别和DQI_ID，用于输出路径和文件的名字。ex.510-RCX/RCX-001
2	`Report_path`	所需提取dqi的文件的路径，可以使用ticket和dfd文件中的变量
3	`Simple_regex`	用于提取值的正值表达式
4	`Only_extract_first`	< true | false >，用于判断是否只提取第一个匹配项
5	`Default_value`	如果找不到匹配项则使用这个默认值
6	`Complex_match_script`	当某个dqi value难以用正则表达式提取时，可以使用脚本提取value值，此项为脚本的路径填写栏位

dqi_ext.spec文件中的每一行都会提取出一份.val文件到::main/.dqi目录下，.val文件记录每个DQI指标的实际输出结果，其命名格式为`<DQI_ID>.val`，其文件内容格式如下：

```
 <VALUE>    <DATE_TIME>         <PATH>            <ATTACHMENT>
ex.5616    20250610_141306    /path/to/report:10      none
```

- `<DQI_ID>.dqi`：此文件中定义了各种状态条件（STATUS_NAME 与 CRITERIA），每个 `.dqi` 文件对应一个设计质量指标（DQI），定义其取值、单位及各个状态（如 Pass、Fail）的判断条件，以下为其文件结构：

```
###    单变量示例：   ###
DQI_ID          AREA
TITLE           Cell Area
DATA_TYPE       real
DATA_UINT       mm2
DESCRIPTION     Total standard cell area
REFERENCE       http://...

STATUS_NAME     pass_cond
STATUS      Pass
@CRITERIA   =   $AREA < 100.0
;

STATUS_NAME     fail_cond
STATUS      Fail
@CRITERIA   =   $AREA >= 100.0
;

DQI_END

###    多变量示例：   ###
STATUS_NAME     ok
STATUS      Pass
@CRITERIA   =   $TP1 == 0 && $TP2 == 0
;

STATUS_NAME     ng
STATUS      Fail
@CRITERIA   =   $TP1 > 0 || $TP2 > 0
;

DQI_END
```

字段说明

字段名	说明
`DQI_ID`	指标代号，应与文件名一致
`TITLE`	简短标题
`DATA_TYPE`	数据类型：`integer`、`real`、`bool`、`string`
`DATA_UNIT`	数据单位，如 `%`、`count`、`mm2` 等
`DESCRIPTION`	详细描述
`REFERENCE`	外部文件或网页参考

状态判断区

类别	运算符	示例
比较运算	`==`, `!=`, `<`, `>`, `<=`, `>=`	`$FOO > 100`
逻辑运算	`&&`, `||`, `!`	`$A > 0 && $B == 0`

类别	运算符	示例
算术运算	`+`, `-`, `*`, `/`	`($X + $Y) / 2 > 5`
括号	`(`, `)`	`($A > 10)`

准备好以上文件后即可在make run的EXEC_DQI步骤中自动抽取出含有流程中的重要指标报告的dqi_rpt.json文件，且此步骤会生成dqi_rpt.json文件的cksum，用于确定后续checkin至DVC中的dqi_rpt.json为同一文件

5. **DVC集成优化**

通过在ticket中指定`DVC_SRC`和`DVC_DST`实现同DVC的集成

- DVC_SRC指定该流程数据的来源版本，实现在build flow的时候建立以下链接，确保使用设计数据的版本正确

```
run_dir/
├── .design -> ${DVC_HOME}/phase/block/stage/version
├── inpNET_LIST -> .design/design.v
```

- DVC_DST指定该流程数据所需checkin的目标版本，通过make checkin实现将需要提交的设计数据checkin到DVC之中
  - make checkin执行前会先对dqi_rpt.json文件进行cksum比对，确认为正确版本后再checkin到DVC以及dqi2db之中
  - 除dqi_rpt.json文件外，还需checkin流程产出的设计数据，具体checkin项目需遵循`STG_REQ_FILE_LIST`这一清单中所定义的文件

6. **DFA命令**

目前DFA包含以下命令：

命令	功能
`dop ticket query`	查看DFA中心资料库可用ticket
`dop ticket checkout <ticket file name>`	将DFA中心资料库中的ticket checkout到本地
`dop ticket build <ticket file name>`	构建EDA工作环境

使用说明：

- **Source environment valuable：在使用DFA前先source位于DOP中心目录下的环境变量脚本，这样才能使用DFA命令**

```
source /path/to/DOP/cshrc_dop
```

- **Ticket query：查看DFA中心资料库下可用的ticket文件**

```
dop ticket query
```

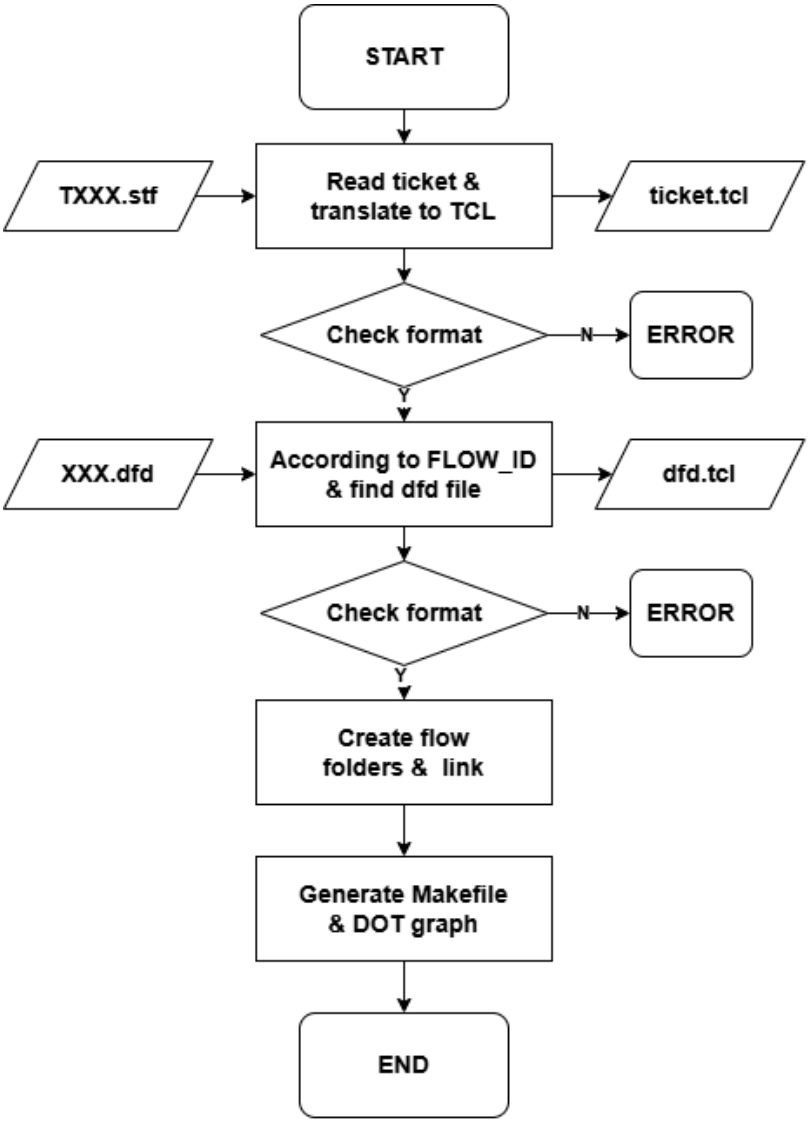
- **Ticket Checkout：从DFA中心资料库中下载PM/CAD准备的Ticket文件到本地工作执行环境**

```
dop ticket checkout <Ticket file name>
```

- **Edit Ticket：根据需求修改Ticket文件中的内容参数**
- **Ticket Build ： 根据Ticket定义的内容产生可执行的EDA工作环境（create run directory & Makefile）,包含把所需要的输入设计文档以及设计规范下载到工作执行环境，EDA工具运行流程也会根据Ticket文件中的参数进行调试，同时也会对资料完整性进行验证，使用以下命令:**

```
dop ticket build <Ticket file name>
```

即可展开流程，其中DFA流程展开的过程如下图所示：



之后会根据ticket以及definition文件的内容生成目录参考如下：

```
### 单个STEP的flow展开后目录结构 ###
ticket_run_dir
├──.techlib -> TLP/techlib_config_file
├──.design -> DVC/phase/block/stage/version
├──.script -> DFA/flow/flow_stage_id
├──.inp$input_ref_id1 -> .design/input_file_name
├──.out$output_ref_id1 -> ::main/output_file_name
└──::main/
      ├──script -> ../.script/flow_ref_id
      ├──input_file_name -> ../.inp$input_ref_id1
      └──Makefile -> script/Makefile.flow

### 多个STEP的flow展开后目录结构  ###
ticket_run_dir
├──.techlib -> TLP/techlib_config_file
```

```
├──.design -> DVC/phase/block/stage/version
├──.script -> DFA/flow/flow_stage_id
├──.inp$input_ref_id1 -> .design/input_file_name1
├──.inp$input_ref_id2 -> .design/input_file_name2
├──.out$output_ref_id1 -> step_dir1/.out$output_ref_id1
├──.out$output_ref_id2 -> step_dir2/.out$output_ref_id2
├──::main/
|      ├──script -> ../.script/flow_ref_id
|      ├──input_file_name1 -> ../.inp$input_ref_id1
|      ├──input_file_name2 -> ../.inp$input_ref_id2
|      ├──output_file_name1 -> ../.out$output_ref_id1
|      ├──output_file_name2 -> ../.out$output_ref_id2
|      └──Makefile -> script/Makefile.flow
├── step_dir1/
|   ├──.techlib -> ../.techlib
|   ├──.design -> ../.design
|   ├──.script -> ../.script
|   ├──.inp$input_ref_id1 -> ../input_file_name1
|   ├──.out$output_ref_id1 -> ::main/output_file_name1
|   └──::main/
|       ├──script -> ../.script/flow_ref_id
|       ├──input_file_name1 -> ../.inp$input_ref_id1
|       └──Makefile -> script/Makefile.flow
└── step_dir2/
    ├──.techlib -> ../.techlib
    ├──.design -> ../.design
    ├──.script -> ../.script
    ├──.inp$input_ref_id2 -> ../input_file_name2
    ├──.inp$output_ref_id1 -> ../step_dir1/.out$output_ref_id1      ### 各个STEP间的依赖关系通过链接构建
    ├──.out$output_ref_id2 -> ::main/output_file_name2
    └──::main/
        ├──script -> ../.script/flow_ref_id
        ├──input_file_name2 -> ../.inp$input_ref_id2
        └──Makefile -> script/Makefile.flow
```

会在::main目录下生成Makefile文件，参考如下：

```
### STEP下的Makefile文件 ###
FLOW          := 510-RCX
INPUT         := input_file_name
OUTPUT        := output_file_name
TOOL          := synopsys_starrc_shell_U
TOOL_OPTION   := -2022.12-SP5-2
QUEUE         := BE
QUEUE_OPTION  := --mem=64G -c 4
PRE_CHECK     := script/run_precheck
EXEC          := script/run_execute
POST_CHECK    := script/run_postcheck
EXEC_DQI      := script/run_execdqi
run :
        ../.dfa/dfa_run.sh
precheck :$(INPUT)
        $(PRE_CHECK) | tee precheck.log
$(OUTPUT) :precheck
        $(EXEC)  | tee execute.log
postcheck :$(OUTPUT)
        $(POST_CHECK) | tee postcheck.log
dqi :$(OUTPUT)
        dqi_extractor.py --spec ./script/dqi_ext.spec --dfd ../.dfa/dfd.tcl --ticket ../.dfa/ticket.tcl
        dqi_generate_rpt.py --flow_dqi_value_path ./.dqi --flow_dqi_path ${DFA_FOLDER}/dqi_def --output
./dqi_rpt.json
```

```
        cksum dqi_rpt.json > ../.dfa/.dqi_cksum
clean :
        rm -rf ./.dqi/* ./dfa.log ./dfa_run_*.log
        dfa_set_status.sh 00_dfa_Init
        script/f_flow_clean.run
rerun :
        make clean
        make run
kill :
        dfaPid=$$(cat ../.dfa/dfa.pid)
        kill -9 $$dfaPid
checkin :
        dfa_checkin.py

### 多个STEP流程下的顶层Makefile文件 ###
FLOW      := 521-DEF2SDF
INPUT     := A.v design.def design.spef.gz design.v
OUTPUT    := design.sdf.gz design.spef.gz
run:
        cd ../510-RCX/::main && make run
        cd ../511-SPEF2SDF/::main && make run
clean:
        cd ../510-RCX/::main && make clean
        cd ../511-SPEF2SDF/::main && make clean
kill:
        cd ../510-RCX/::main && make kill
        cd ../511-SPEF2SDF/::main && make kill
precheck:
        cd ../510-RCX/::main && make precheck
        cd ../511-SPEF2SDF/::main && make precheck
postcheck:
        cd ../510-RCX/::main && make postcheck
        cd ../511-SPEF2SDF/::main && make postcheck
dqi:
        cd ../510-RCX/::main && make dqi
        cd ../511-SPEF2SDF/::main && make dqi
checkin:
        cd ../510-RCX/::main && make checkin
        cd ../511-SPEF2SDF/::main && make checkin
rerun: clean run
design.sdf.gz: A.v design.spef.gz design.v
        cd ../511-SPEF2SDF/::main && make design.sdf.gz
design.spef.gz: A.v design.def
        cd ../510-RCX/::main && make design.spef.gz
```

- **Execute Ticket： 调用所需的EDA工具执行设计，产出设计输出文档，切换到::main目录下执行以下命令：**

```
make run
```

- **Checkin Design：将该流程产生的设计数据以及抽取的重要设计资料上传至DVC的设计资料库中心**

```
make checkin
```

附录

DFA下载方式：

```
git clone https://github.com/icdop/dop.git
or
git clone https://gitee.com/icdop/dop.git
```

用户使用手册：

https://lyg365.sharepoint.com/sites/cad/Shared%20Documents/Forms/AllItems.aspx?id=%2Fsites%2Fcad%2FShared%20Documents%2FDOP%2FDFA%2FDFA%20User%20Manual%E2%80%94%E2%80%94for%20CAD%2FDFA%20User%20Manual%E2%80%94%E2%80%94For%20CAD%5F20250707v1%2Epdf&viewid=63537b0b%2Dbdf6%2D4006%2D9eb3%2D39061696a4f5&parent=%2Fsites%2Fcad%2FShared%20Documents%2FDOP%2FDFA%2FDFA%20User%20Manual%E2%80%94%E2%80%94for%20CAD