CTB Flash Integration

Table of contents

CTB Integration as part of FLASH

at this document we will define how CTB will be integrated as part of FLASH TRACE CONVERSION FLOW.

Introduction

to scale CTT technology, CTT atomic patterns are required to be easily generated for all content type and flows, based no MPE current pattern-generation flow, integrating CTB as part of trace convert flow & TRC2ATE is the best approach to accomplish this, since there are several trace-generation flows but a single pattern generation flow, therefore, integrating CTB tool as part of TRACE-CONVERT flow will encapsulate the differences and make the marking flow simpler & eaiser.

CTB-WRAPPER Version integration into FLASH

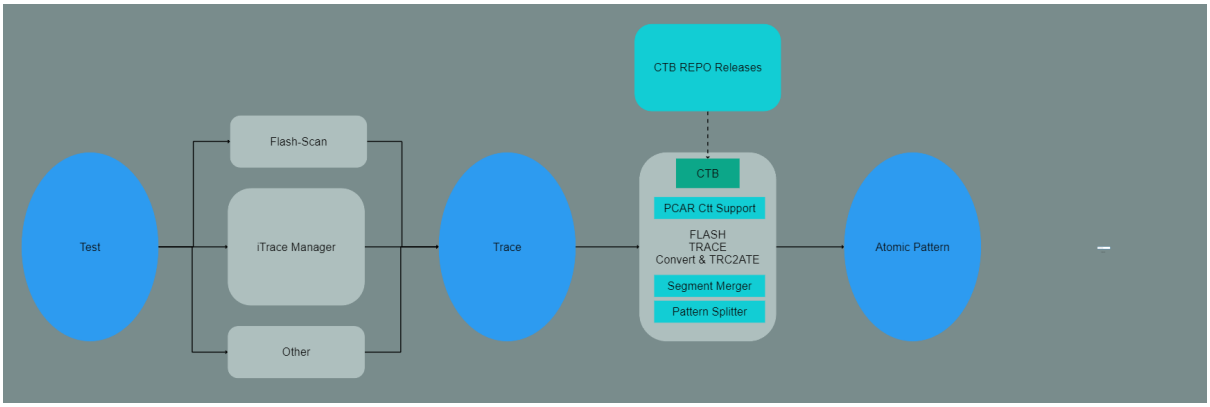
TRACE_CONVERT/TRACE2ATE will call a "ctb-wrapper" tool providing Project & Step as well, the wrapper will resolve the acutal CTB version to be executed and perform the execution, the motivation of using ctb-wrapper tool is to encapsulate CTB releases from FLASH and keep the product teams to be the decesion makers.

Pre Requisite

users must have access to CTB CRT releases area Group Permissions: **soc**

Flow Diagram

CTB will be integrated at the pattern conversion tools and will handle all the traces regardless of the trace-generated flow (iTraceMgr,FlashScan, other..)



high_level_flow.png

CTB integration in details

```
ctb-wrapper mark --cfg [CFG_FILE JSON] --stil [STIL] --metadata [METADATA JSON] --rundir [RUNDIR] --result_name [ctb_mark_result.json]
```

Inputs

CFG File

the configuration file will include all parameters required for CTB to be executed, as follows:

```
{
  "product": "LNL",
  "step": "B0",
  "tag" : null,
  "version_override": null,
  "custom_marker": "MARKER_ALIAS_NAME",
  "backdoor1": "value1"
}
```

STIL

full path to the stil file to be marked

METADATA File

the metadata file will contain the pattern name disassemble information in both bit & full value

```
{
  "naming_metadata": {
    "namefield1": {
      "bit": "bit_value",
      "full": "full_value"
    },
    "namefield2": {
      "bit": "bit_value2",
      "full": "full_value2"
    }
```

```
    }
}
```

RUNDIR

full path to a folder where CTB can create logs and other files required for the transformation

Result Name

the name of the file that will hold the result information of the marking - default: ctb_mark_result.json the file will be located at the supplied RUNDIR (e.g. RUNDIR/ctb_mark_result.json)

Outputs

CTB will create a JSON file that contains full path to the marked STIL, exit-status and error message incase of any issue

Exit Codes:

Exit Code	Description
0	SUCCESS
Other	FAILURE

Output Json structure

in-case of success run:

```
{
  "preMarkInputFile": "FULL_PATH_TO_INPUT_FILE",
  "postMarkInputFile": "FULL_PATH_TO_OUTPUT_FILE",
  "exitStatus" : 0,
  "errorMessage" : null
}
```

in-case of failure run:

```
{
  "preMarkInputFile": "FULL_PATH_TO_INPUT_FILE",
  "postMarkInputFile": null,
  "exitStatus" : "4",
```

```
  "errorMessage" : "Failed to find a marker script to handle the input
stimulus"
}
```

Opens

- define how CTB version override will be supported?
- define how CTB backdoor flags can be supplied from the user