TCE User Group Meeting
Digital Result Memory Handling in SMT 7.2.1.X

Presented by: Stephen Fu/Tan Kheng How

25-Nov-2013

Overview

- Background
- Current situation / issues
- New feature overview
  - Enhance the algorithm to allocate for memory chunks
  - Result memory reservation via device parameter
  - Dynamic result memory resize via FW command / TM API
  - Automatic result memory handling for digital capture
- Discussions

Background

How Smartest stores digital results (error map, digital capture) in digital channel memory:

- The test processor has **an internal error map**. It is 2KB (16K bit) in size for PinScale and is 8KB (64K bit) for SmartScale. When the internal error map is not big enough, the user can switch to use the **external memory (digital channel memory)** for result storage by issuing the FW command: DCRT RAM, (port list);

- The result area is dynamically allocated for external error map, digital capture. The software tries to find the maximum **available common memory block across all channels** that are part of the selected port(s), in the currently present memory layout .

Background

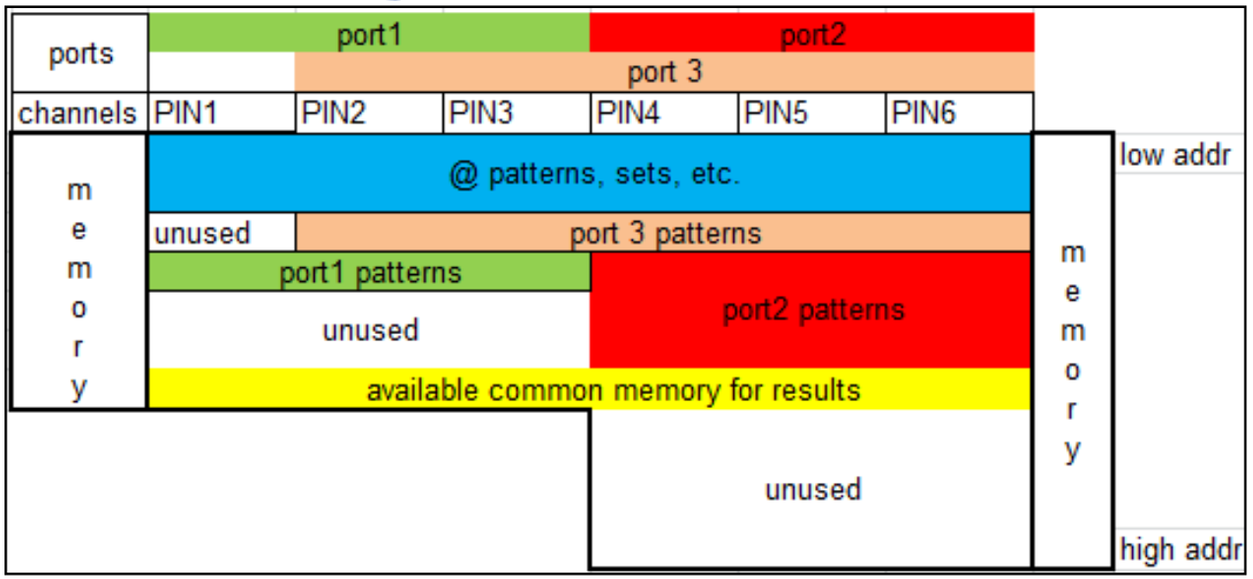
V93k smartest **digital channel memory** management:

- Digital channel memory management is founded on per-pin architecture. It is licensing based.

- Digital channel memory is used for storage of vectors, sequencer programs, sets (timing, level, etc.), results, etc.

- Memory management has been continuously enhanced: Pattern download sequence was based on name. It was enhanced to download “@” pattern first. Since Smartest 7, a new and complicated memory management algorithm is used to minimize “memory holes” as much as possible.

Vector Memory Allocation



- VM + SM + sets + results
- VM depth could be different per pin
- Combination of Multiport and Single Port
- Memory Pooling (Sharing VM across pins in same test processor)
- “Available Common Memory for Results” -> Free unallocated address area across all the pins

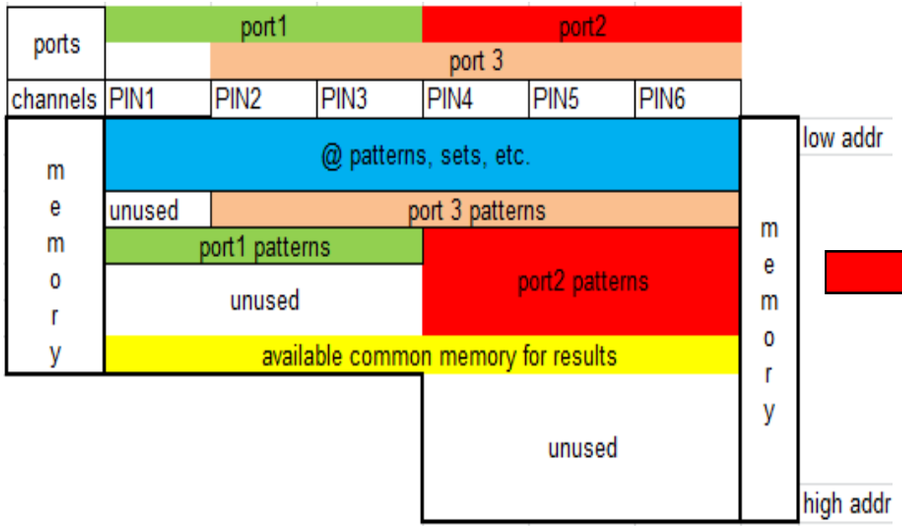
Need to optimize “Available Common Memory for Results” to “squeeze in” more vectors across the pins

New Feature Overview: Enhance the algorithm to allocate for memory chunks

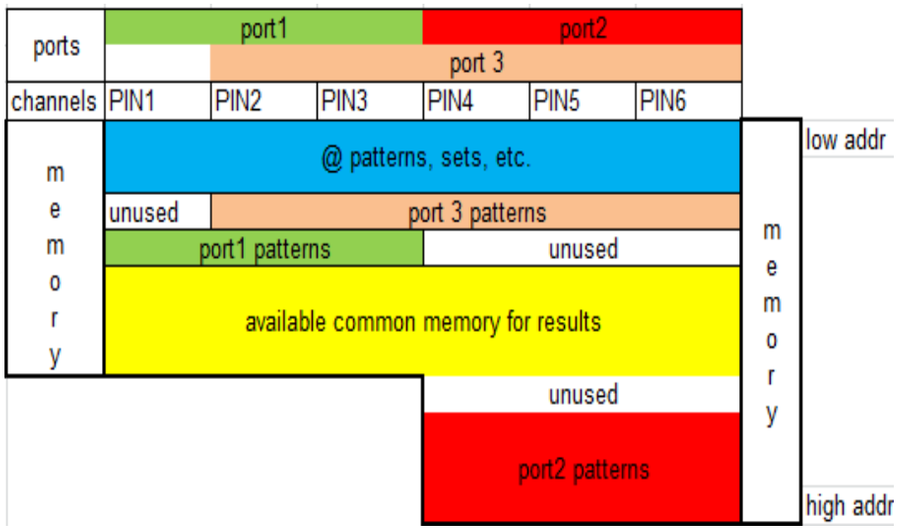
Enhance the algorithm to add a strategy for allocations for memory chunks which cover only a subset of channels with greater memory depth

This enhancement has been implemented in 7.2.0.5.

Before 7.2.0.5



Since 7.2.0.5



As you can see in this example, port2 patterns (red block) are downloaded differently, so that the available common memory (yellow block) for all pins is larger.

Current situation/issues (7.1.4.x)

- Memory location for results must be re-calculated for every test execution -> noticeable performance impact when switching from internal error map to external memory.

  i.e. switching error map recording to RAM, test time will increase as system is calculating for the remaining space during every execution.

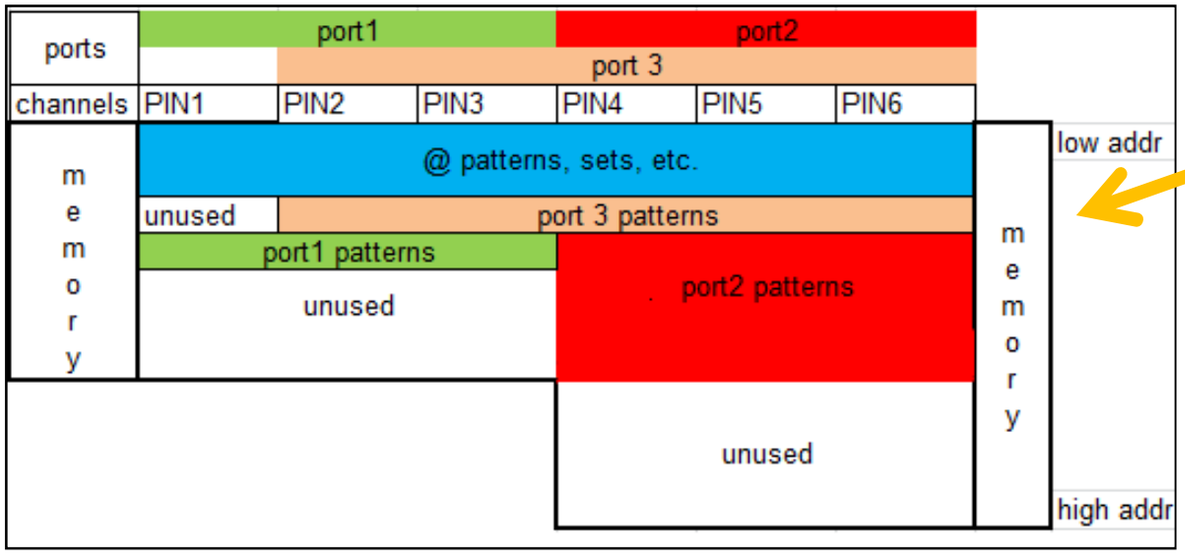
  CR-77457 [scan_tml] scan_tml has large test time hit for good parts over ac_tml FunctionalTest

- Temporary workarounds (RRDF) revealed SMT issues such as crashes and greatly decreased usability (requirement to execute testsuites as group due to automatic removal of all user defined result areas at the start of each execution and having to use FW commands at all).

  CR-79588 "RRUD RRDF workaround causing testflow execution crash" which was found on SMT-7.1.4.6 and is fixed on SMT-7.1.4.8 and SMT-7.2.0.6.

Current situation/issues (7.1.4.x)

- Result module is only able to use free space currently available without memory layout optimization -> often leads to no space for results while each pin still has free memory (just not at a common address).



No “Available Common Memory for Results”

- As above example, the user will get a runtime error like: `"The Acquire target for port '@‘ is not supported ".`

- **CR-80144 “What is "The Acquire target for port '@' is not supported."?”** which was found on SMT-7.1.4.6 and is fixed on SMT-7.1.4.8 and SMT-7.2.0.6.

Summary of workarounds

RRDF/RRUD

- This workaround use “RRDF” to “defragment” common address area across all the pins
- After that, “RRUD” will undefined the area.
- VM will remained “defragmented” after “RRUD”, allowing user to squeeze more common area

Pros:

- Using Default Result Area, easier to use during debug

Cons:

- Result area recalculated during each execution, may cause test time impact

RRUD/RRDF/RRSC

- This workaround defined Result Area through “RRDF” FW command
- RRSC FW command will activate Result Area defined by “RRDF” FW command.
- In other words, user defined Result Area has been used, instead of Default Result Area

Pros:

- No need to recalculate result area after each execution

Cons:

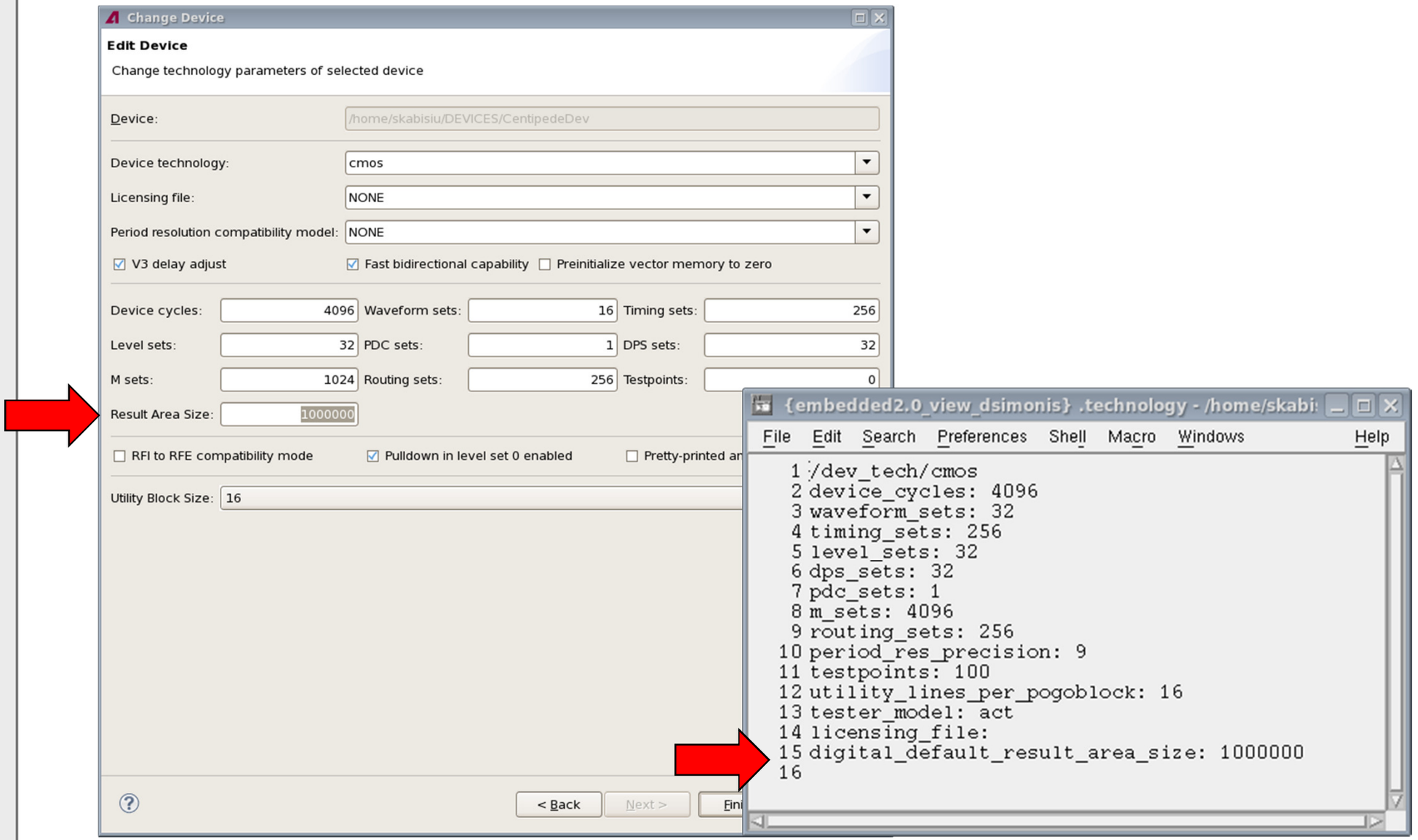
- Need to execute RRDF/RRSC FW command testsuites before executing normal testsuite.

New Feature Overview: (1) result memory reservation via device parameter – 7.2.1.0

- Motivation:
  - Need one base solution that can be used in all scenarios (including those where the other additional convenience features are not applicable)
- Usage:
  - Specify the amount of memory to be reserved for digital results in the .technology file via the following parameter:
  - digital_default_result_area_size: 1000000
  - Parameter can be edited via the 'Edit / Create Device' dialogs
  - Calculations:
    - Acquire Fail mode: 11 bytes / failure (worst case – very sparse error distribution).
      - Example: to ensure capture of the first 50K failures, 50K Fails * 11 Bytes/Fail = 550KB of memory must be reserved
    - Acquire All mode: Size is calculated based on the cycle range to be recorded and the result x-mode and edge resolution (1 / 2 bit mode):
      - bytes_required = num_cycles * x-mode * bits_per_result / 8 bits / byte
      - Example: Max cycle range to be captured is 80K cycles in x-4 (one bit per edge) result mode. Bytes_required = 80K * 4 * 1 / 8 = 40KB

- **Not intended to be used during test program development unless a commonly applicable value has been determined**
- **This approach is alternative to original "RRDF/RRUD" or "RRUD, RRDF, RRSC" workaround**

New Feature Overview: (1) result memory reservation via device parameter – 7.2.1.0



New Feature Overview: : (2) dynamic result area resize via FW command / TM API – 7.2.1.0

- Each Testsuite can increase the size of the default result area via FW command or TM API
- FW Usage:
  - RSCF “”, DIG_EMAP_RESULT_AREA_SIZE,<size>,(@)
  - Based on result area configuration set API, empty set name used to refer to default set. Pin list parameter must be ‘@’ for the default set.
- TM API Usage
  - Based on the RESULT_AREA_CONFIG_SET API, which now allows modification of the default set

```
14 void setupEmapForNumFails(int maxFailCycles) {
15
16   RESULT_AREA_CONFIG_SET defaultResultArea("");
17
18   // configure the default result area size for all pins
19   int requiredSpace = maxFailCycles * 11;
20   defaultResultArea.pin("@").setErrorMapResultAreaSize(requiredSpace);
21 }
```

New Feature Overview: (2) dynamic result area resize via FW command / TM API – 7.2.1.0

- Advantages:
  - Independent test suite development possible – no global merge required
  - Result area is dynamically resized when a testsuite requests a larger default result area size
  - After one testflow execution, maximum result memory requirement has been determined and no additional memory re-size operations are needed, giving the same optimal performance thereafter (as .technology)
- Disadvantage:
  - Result memory usage is not visible for ‘tat’ license generator utility under standard operating mode (as setup is only loaded but not executed by ‘tat’, therefore the dynamic result memory requirement is not considered)
  - Workaround: query actual result area size after one complete testflow execution and use this value for the .technology parameter (while this value may change as testsuites are added, it provides a better value than the assumed size of zero)

```
task: RSCF? "",,(@)
RSCF "",DIG_EMAP_RESULT_AREA_SIZE,1000000,(p)
```

New Feature Overview: automatic result memory handling for Digital Capture – 7.2.1.1

- SmarTest will handle result memory management for digital capture without test engineer interaction
- As patterns with digital capture definitions are loaded, the maximum memory requirement for each capture pin is calculated based on each vector variable definition and immediately allocated.
- Advantages:
  - Completely automatic (other than having to enable this feature)
  - Low memory conditions, which previously only showed up as run time errors during execution – e.g. 'not all samples could be captured', are now detected at pattern load time
- Limitation:
  - Vector variables must be saved in the pattern files (std case), i.e. no dynamic vector variable generation

Discussions

Q&A / Comments / Feedback