Automating the formal verification sign-off flow of configurable digital IP’s

Giovanni Auditore, Giuseppe Falconeri
STMicroelectronics
Catania (Italy)
Wondering where is Catania...?

...just follow the smoke signals!
Agenda

• Introduction
• Configurable IP formal verification
• The execution challenge
• Coverage driven approach
• Real case study
• Real case study results
• Conclusions
Configurable IP formal verification

\[ RT_{TOT} = \sum_{i=1}^{N} RT_i \]

\[ COV_{TOT} = \bigcup_{i \in [1..N]} COV_i \]
The execution challenge

Assuming $RT_{AVVERAGE} = 6 \text{ hours}$

• $N = 4 \Rightarrow 1 \text{ day}$ 😊

• $N = 256 \Rightarrow 64 \text{ days}$ ☠️ => Increase parallelism by 8 $\Rightarrow \sim 1 \text{ week}$ 😞

• $N = 96 \times 2 \times 16 \times 2^{96} \times 2^{96} \times 2^{96} \times 2^{96} \Rightarrow \text{Simply ... NO WAY!!!}$

Need a compromise
The execution challenge

Reduce Run Time

- Increase parallelism
- Cost increase
- Lower Coverage
- Compromise quality

Reduce Number of Runs
Coverage driven approach

• Define the coverage metric target also based on IP configuration parameters
• Define constraints on illegal parameter values combinations
• Apply randomization to extract a subset of configurations which maximize the agreed coverage target
• Execute formal runs only on the identified subset
• Automate the flow to reduce manual intervention
• Add to the standard sign-off metrics also the configuration metrics
Full coverage metrics driven approach

EXECUTION MANAGER

START

Constrained random configuration

Constrained random configuration generation test

Executable configuration related scripts

Configuration formal verification execution

Full metrics Coverage Database

Configuration metrics collection

Formal metrics collection

Coverage met?

YES

Rank

DONE

NO
Configuration metrics driven approach

EXECUTION MANAGER

START

Constrained random configuration generation

NO

Constrained random configuration generation test

YES

Config Cov met?

NO

Full metrics Coverage Database

EXECUTABLE configuration related scripts

Configuration metrics collection

Rank

Configuration formal verification execution

Review configuration coverage and assertion set and loop again

Coverag e met?

NO

DONE

YES
Configurations ranking flow

for gen in `seq 1 $maxGenNb`; do
  ### Call Specman, load the configuration e code file, execute generation N Times
  This step will create the yml and the vsif files of each generated config

  ### Call IMC to merge the coverage and generate report

  ### Evaluate the metric achieved, extracted by parsing the report log

  ### Stop if the target metric was reached, otherwise increase N and loop
  done

  ### Extract ranked configurations

  ### Create the top vsif file, importing only ranked configurations vsif files

  ### Run top.vsif file in vManager
# Real case study: Event Controller IP

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Parameter Description</th>
<th>Parameter Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>nb_of_events</td>
<td>Number of events</td>
<td>16 to 96</td>
</tr>
<tr>
<td>nb_of_cpu</td>
<td>Number of CPU interrupt controller</td>
<td>1 to 4</td>
</tr>
<tr>
<td>trig_cfg</td>
<td>Type of event, a selection among two different kind of events</td>
<td>96 bits mask (only the lower Number of Events bits are valid)</td>
</tr>
<tr>
<td>cpu_rxev_en</td>
<td>Optional propagation of enabled events to dedicated CPU outputs</td>
<td>4 bits mask (only the lower Number of CPU bits are valid)</td>
</tr>
<tr>
<td>rxev_cfg</td>
<td>Mask bit vector for events enabled to CPU propagation</td>
<td>96 bits mask (only the lower Number of Events bits are valid)</td>
</tr>
<tr>
<td>tz_cfg</td>
<td>Mask bit vector for events which implement AHB5 TrustZone security protection</td>
<td>96 bits mask (only the lower Number of Events bits are valid)</td>
</tr>
</tbody>
</table>
Real case study: Event Controller IP

```c
struct event_config_s {
    rtl_version : rtl_version_t;
    event_index : uint(bits:7);
    trig_cfg : bit;
    trig_is_port : bool;
    keep trig_is_port == (read_only(event_index) < 16);
    rxev_cfg : bit;
    next_trig_cfg : bit;
    keep soft next_trig_cfg == 0;
    next_rxev_cfg : bit;
    keep soft next_rxev_cfg == 0;
    tz_cfg : bit;
};

cover cover_event_config_e is {
    item event_index using per_instance, ignore=(event_index>95);
    item trig_cfg;
    item rxev_cfg;
    item next_trig_cfg using no_collect;
    item next_rxev_cfg using no_collect;
    item tz_cfg;
    cross trig_cfg, rxev_cfg ;
    cross trig_cfg, next_trig_cfg;
    cross rxev_cfg, next_rxev_cfg;
};

cover config_cover_e is {
    item nb_of_events using
    ignore - (nb_of_events>95 or nb_of_events<16),
    illegal - (nb_of_events<16),
    ranges - {
        range([16]    , "Minimum number of events", UNDEF, 1);
        range([17..31], "Low number of events", UNDEF, 2);
        range([32..57], "Medium number of events", UNDEF, 2);
        range([58..94], "High number of events", UNDEF, 2);
        range([95]    , "Maximum number of events", UNDEF, 1);
    };
    item nb_of_cpu;
    item cpu_rxev_en using
    ranges - {
        range([0]    , "ALL OFF", UNDEF, 1);
        range([1]    , "ONLY CPU0 ON", UNDEF, 1);
        range([2]    , "ONLY CPU1 ON", UNDEF, 1);
        range([4]    , "ONLY CPU2 ON", UNDEF, 1);
        range([8]    , "ONLY CPU3 ON", UNDEF, 1);
        range([3]    , "ONLY CPU0 CPU1 ON", UNDEF, 1);
        range([6]    , "ONLY CPU1 CPU2 ON", UNDEF, 1);
        range([0xC]  , "ONLY CPU2 CPU3 ON", UNDEF, 1);
        range([0xF]  , "ALL ON", UNDEF, 1);
        range([5,7,9,0xA,0xB,0xD,0xE]  , "SOME ON", UNDEF, 1);
    };
};
```

Full metrics approach results

<table>
<thead>
<tr>
<th>Name</th>
<th>Combined Average Grade</th>
<th>Combined Covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMETRICS</td>
<td>99.98% *</td>
<td>60188 / 60206 (99.97%)</td>
</tr>
<tr>
<td>1 Functional requirements verification plan</td>
<td>99.98% *</td>
<td>60188 / 60206 (99.97%)</td>
</tr>
<tr>
<td>1.1 HW Generics</td>
<td>99.78% *</td>
<td>1080 / 1098 (98.36%)</td>
</tr>
<tr>
<td>1.1.1 HW Generics cover</td>
<td>99.57% *</td>
<td>1078 / 1096 (98.36%)</td>
</tr>
<tr>
<td>1.1.1.1 nb_of_event</td>
<td>✓ 100%</td>
<td>5 / 5 (100%)</td>
</tr>
<tr>
<td>1.1.1.2 nb_of_cpu</td>
<td>✓ 100%</td>
<td>4 / 4 (100%)</td>
</tr>
<tr>
<td>1.1.1.3 cpu_rxev_en</td>
<td>✓ 100%</td>
<td>10 / 10 (100%)</td>
</tr>
<tr>
<td>1.1.1.4 trg_cfg</td>
<td>✓ 100%</td>
<td>1 / 1 (100%)</td>
</tr>
<tr>
<td>1.1.1.5 rxev_cfg</td>
<td>✓ 100%</td>
<td>2 / 2 (100%)</td>
</tr>
<tr>
<td>1.1.1.6 tz_cfg</td>
<td>✓ 100%</td>
<td>2 / 2 (100%)</td>
</tr>
<tr>
<td>1.1.1.7 priv_cfg</td>
<td>✓ 100%</td>
<td>2 / 2 (100%)</td>
</tr>
<tr>
<td>1.1.1.8 nb_ioports</td>
<td>✓ 100% *</td>
<td>5 / 5 (100%)</td>
</tr>
<tr>
<td>1.1.1.9 Cross_dependency_trig_rxev</td>
<td>98.68%</td>
<td>345 / 350 (98.57%)</td>
</tr>
<tr>
<td>1.1.1.10 Cross_dependency_contiguous_trig</td>
<td>98.68%</td>
<td>330 / 335 (98.51%)</td>
</tr>
<tr>
<td>1.1.1.11 Cross_dependency_contiguous_rxev</td>
<td>97.89%</td>
<td>372 / 380 (97.89%)</td>
</tr>
<tr>
<td>1.1.2 HW Generics checks</td>
<td>✓ 100%</td>
<td>2 / 2 (100%)</td>
</tr>
<tr>
<td>1.2 AIEC registers access</td>
<td>✓ 100%</td>
<td>45294 / 45294 (100%)</td>
</tr>
<tr>
<td>1.3 Clocking strategy</td>
<td>✓ 100%</td>
<td>21 / 21 (100%)</td>
</tr>
<tr>
<td>1.4 Input events trigger</td>
<td>✓ 100%</td>
<td>528 / 528 (100%)</td>
</tr>
<tr>
<td>1.5 IOPORT inputs selection</td>
<td>✓ 100%</td>
<td>4128 / 4128 (100%)</td>
</tr>
<tr>
<td>1.6 System wakeup on inputs events</td>
<td>✓ 100%</td>
<td>1705 / 1705 (100%)</td>
</tr>
<tr>
<td>1.7 CPU wake up on inputs events</td>
<td>✓ 100%</td>
<td>3481 / 3481 (100%)</td>
</tr>
<tr>
<td>1.8 Configurable events output interrupt to all CPU's</td>
<td>✓ 100%</td>
<td>672 / 672 (100%)</td>
</tr>
<tr>
<td>1.9 CPU output event on inputs events</td>
<td>✓ 100%</td>
<td>3279 / 3279 (100%)</td>
</tr>
</tbody>
</table>
## Configuration metrics approach results

### vPlan Hierarchy

<table>
<thead>
<tr>
<th>Name</th>
<th>Combined Average Grade</th>
<th>Combined Covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>C7AMBA_AIEC_V2_2_VMETRICS</td>
<td>98.66% *</td>
<td>60199 / 60206 (99.99%)</td>
</tr>
<tr>
<td>Functional requirements verification plan</td>
<td>98.66% *</td>
<td>60199 / 60206 (99.99%)</td>
</tr>
<tr>
<td>HW Generics</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.1 HW Generics cover</td>
<td>99.92% *</td>
<td>1091 / 1098 (99.36%)</td>
</tr>
<tr>
<td>1.1.1 nb_of_event</td>
<td>100% *</td>
<td>5 / 5 (100%)</td>
</tr>
<tr>
<td>1.1.2 nb_of_cpu</td>
<td>100%</td>
<td>4 / 4 (100%)</td>
</tr>
<tr>
<td>1.1.3 cpu_rxev_en</td>
<td>100%</td>
<td>10 / 10 (100%)</td>
</tr>
<tr>
<td>1.1.4 trg_cfg</td>
<td>100%</td>
<td>1 / 1 (100%)</td>
</tr>
<tr>
<td>1.1.5 rxev_cfg</td>
<td>100%</td>
<td>2 / 2 (100%)</td>
</tr>
<tr>
<td>1.1.6 tz_cfg</td>
<td>100%</td>
<td>2 / 2 (100%)</td>
</tr>
<tr>
<td>1.1.7 priv_cfg</td>
<td>100%</td>
<td>2 / 2 (100%)</td>
</tr>
<tr>
<td>1.1.8 nb_ioports</td>
<td>100% *</td>
<td>5 / 5 (100%)</td>
</tr>
<tr>
<td>1.1.9 Cross_dependency_trig_rxev</td>
<td>99.47%</td>
<td>348 / 350 (99.43%)</td>
</tr>
<tr>
<td>1.1.10 Cross_dependency_contiguous_trig</td>
<td>99.21%</td>
<td>332 / 335 (99.1%)</td>
</tr>
<tr>
<td>1.1.11 Cross_dependency_contiguous_rxev</td>
<td>99.47%</td>
<td>378 / 380 (99.47%)</td>
</tr>
<tr>
<td>1.1.2 HW Generics checks</td>
<td>100%</td>
<td>2 / 2 (100%)</td>
</tr>
<tr>
<td>1.2 AIEC registers access</td>
<td>88% *</td>
<td>45294 / 45294 (100%)</td>
</tr>
<tr>
<td>1.3 Clocking strategy</td>
<td>100%</td>
<td>21 / 21 (100%)</td>
</tr>
<tr>
<td>1.4 Input events trigger</td>
<td>100%</td>
<td>528 / 528 (100%)</td>
</tr>
<tr>
<td>1.5 IOPORT inputs selection</td>
<td>100%</td>
<td>4128 / 4128 (100%)</td>
</tr>
<tr>
<td>1.6 System wakeup on inputs events</td>
<td>100%</td>
<td>1705 / 1705 (100%)</td>
</tr>
<tr>
<td>1.7 CPUUn wakeup on inputs events</td>
<td>100%</td>
<td>3481 / 3481 (100%)</td>
</tr>
<tr>
<td>1.8 Configurable events output interrupt to all CPU's</td>
<td>100%</td>
<td>672 / 672 (100%)</td>
</tr>
<tr>
<td>1.9 CPUUn output event on inputs events</td>
<td>100%</td>
<td>3279 / 3279 (100%)</td>
</tr>
</tbody>
</table>
### Real case study results

<table>
<thead>
<tr>
<th>Approach kind</th>
<th>Number of generated configurations</th>
<th>Number of executed configurations</th>
<th>Total run time</th>
<th>Configuration coverage metrics</th>
<th>Full Coverage metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full metrics driven</td>
<td>30</td>
<td>30</td>
<td>~ 180h</td>
<td>98.36%</td>
<td>99.97%</td>
</tr>
<tr>
<td>Configuration metrics driven</td>
<td>440</td>
<td>22</td>
<td>~ 125h</td>
<td>99.36%</td>
<td>99.99%</td>
</tr>
</tbody>
</table>
Conclusions

Focusing on configuration coverage helps to formalize a shared verification objective for configurable digital designs.

Random generation and coverage metrics collection allows to maximize the verification robustness and reduces the execution cost needed to achieve the agreed target result.

Automation reduces human intervention in a long lasting task allowing the same to progress in background.
Weakness

- Selected configuration coverage comes from human agreement
- Some properties in different configurations runs may share the same cone of logic

Future developments

- Reduction algorithms of a N-dimensional space can be applied, using the concept of distance between configurations
- Take advantage of data collection to skip useless properties execution and further reduce the run time

... and maybe more. Stay tuned!
Thank you!