2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

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

Giovanni Auditore, Giuseppe Falconeri

STMicroelectronics

Catania (Italy)

accellera SYSTEMS INITIATIVE

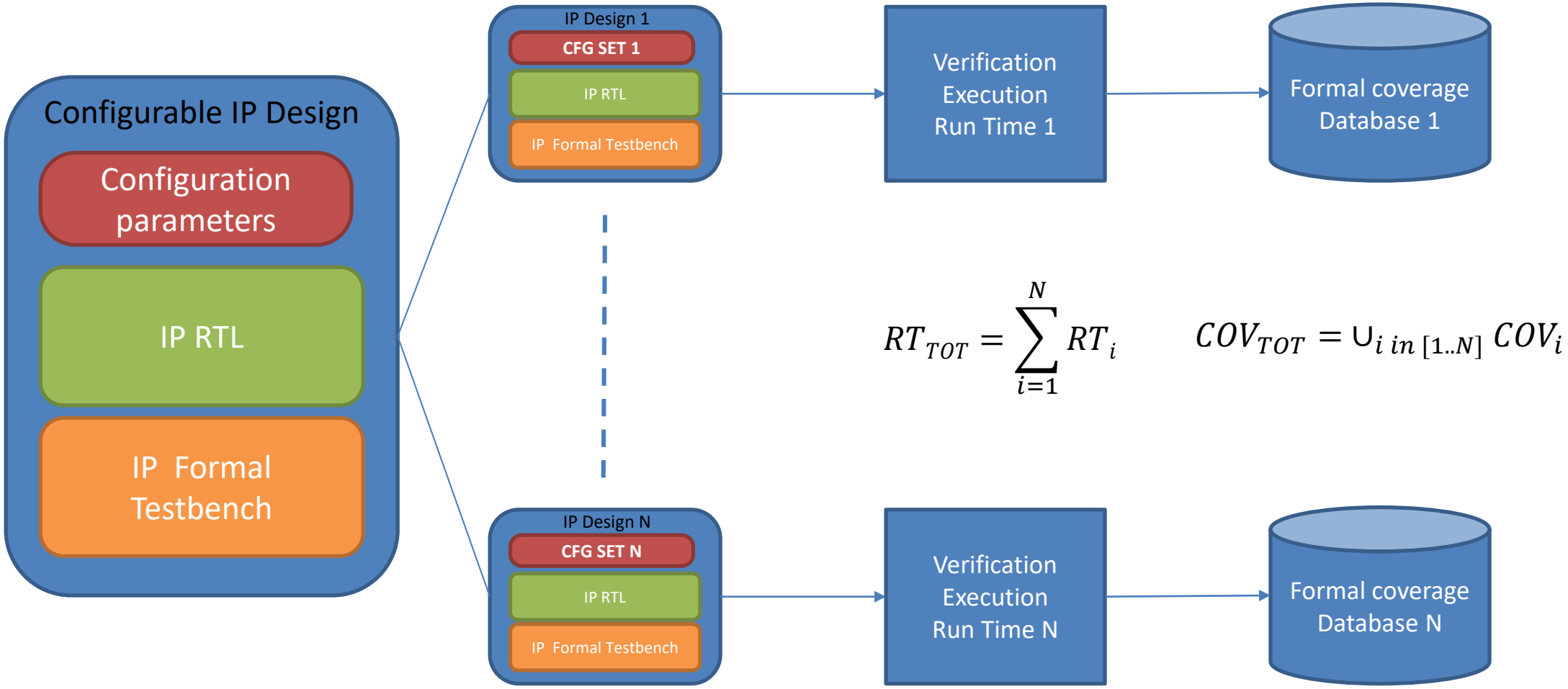
life.augmented


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



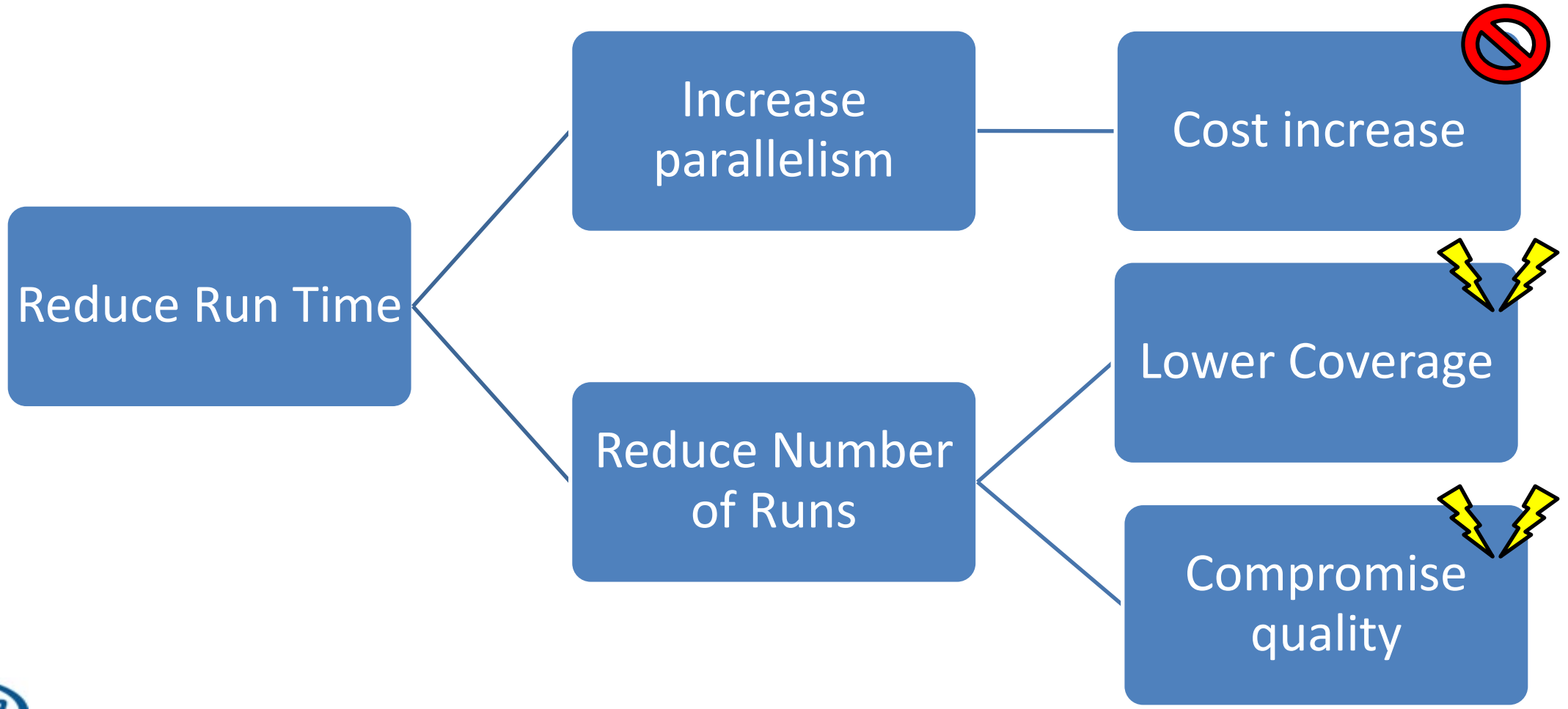
The execution challenge

Assuming $RT_{AVERAGE} = 6\ hours$

- N = 4 => 1 day
- N= 256 => 64 days => Increase parallelism by 8 =>~ 1 week
- N= $96 * 2 * 16 * 2^{96} * 2^{96} * 2^{96} * 2^{96}$ => Simply … NO WAY!!!

Need a compromise

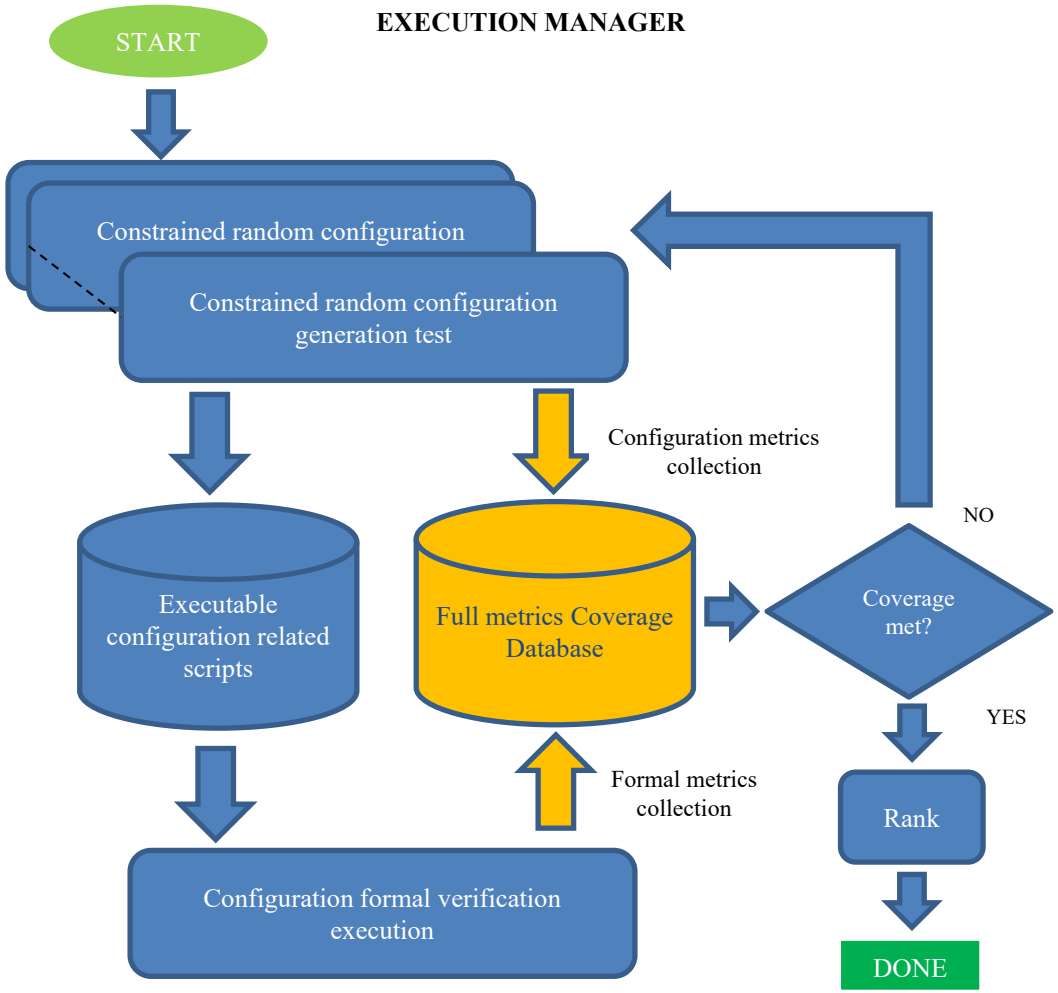
The execution challenge



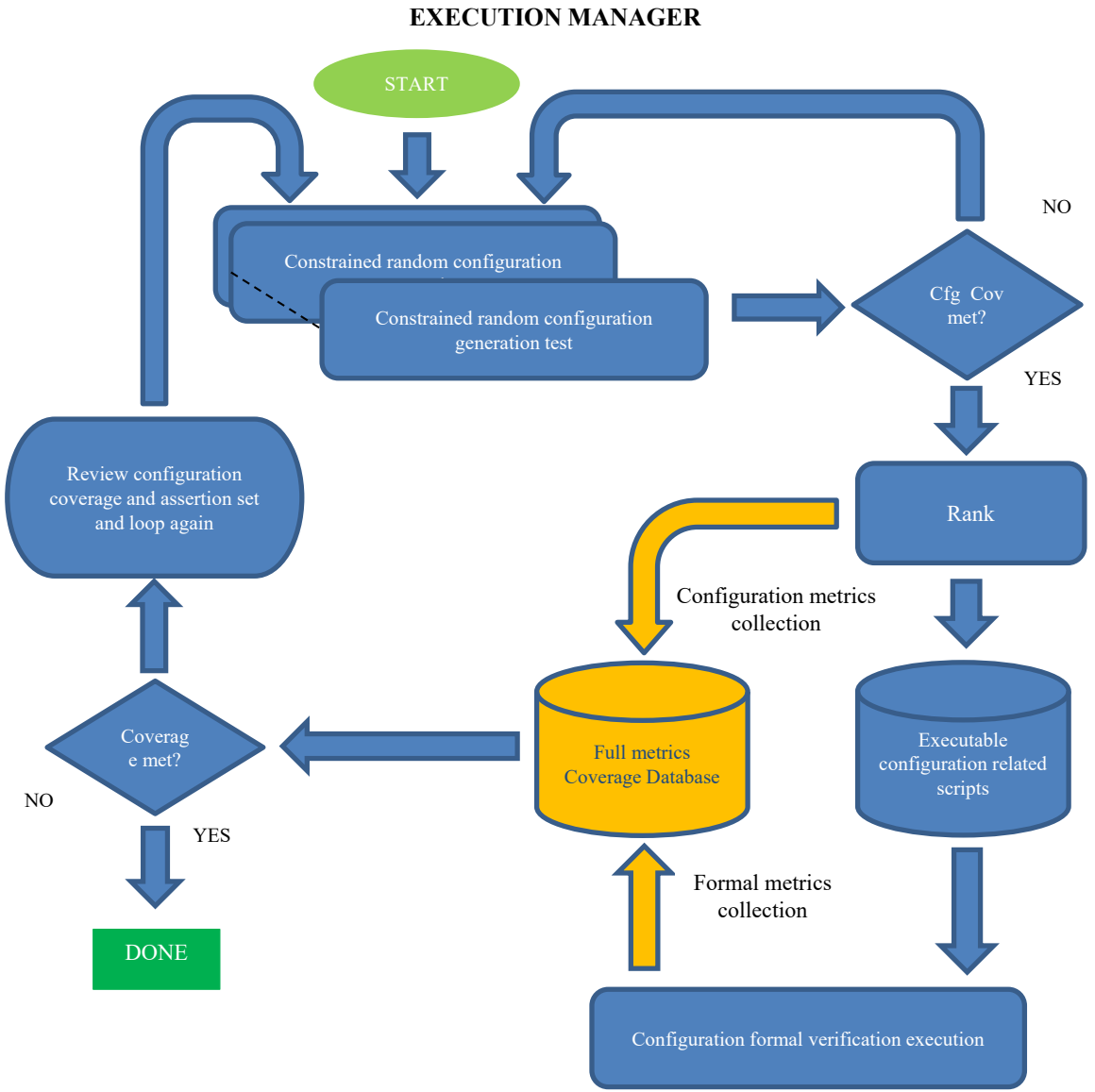
Coverage driven approach

- 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



Configuration metrics driven approach



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

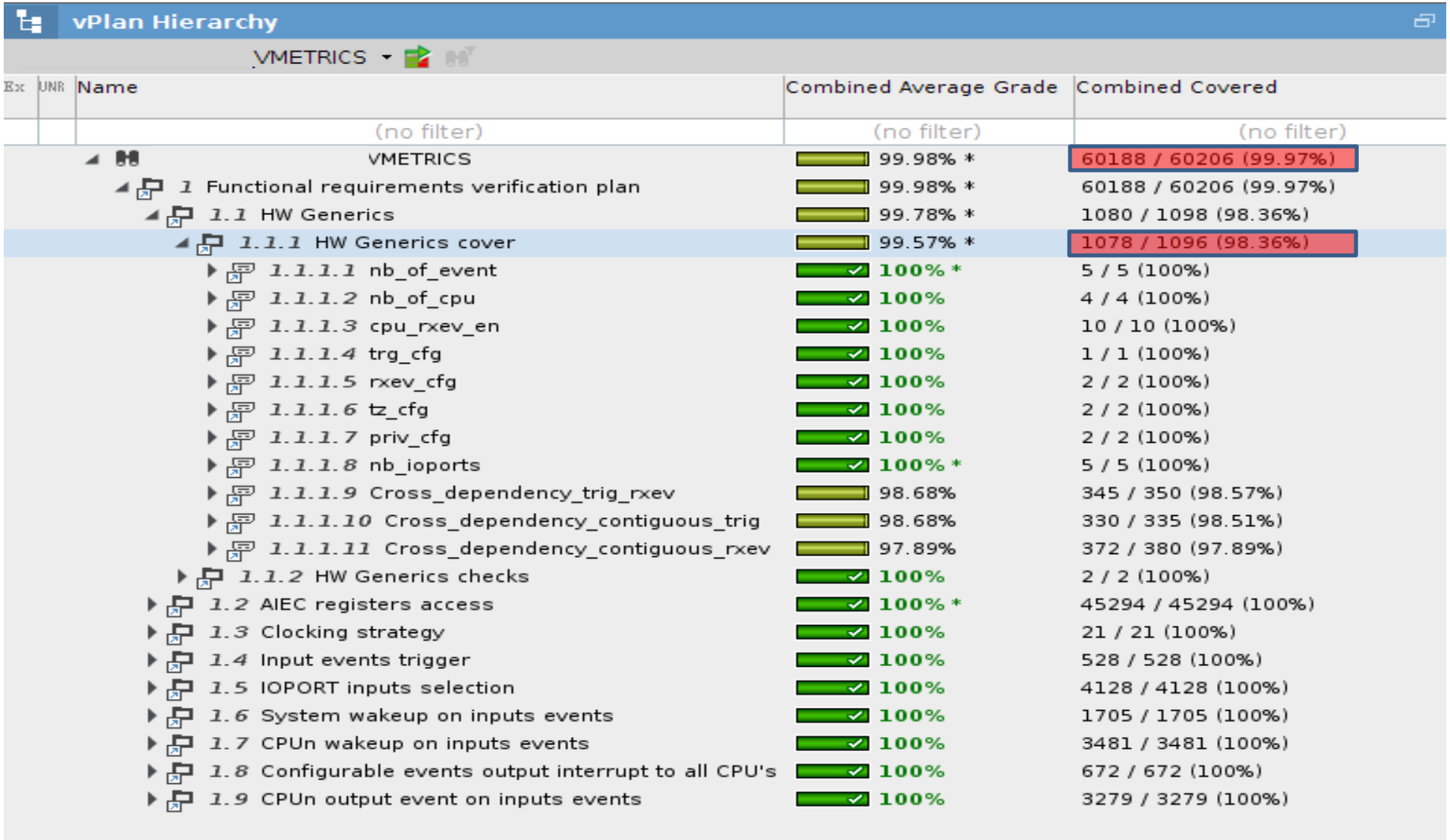
Parameter Name	Parameter Description	Parameter Range
nb_of_events	Number of events	16 to 96
nb_of_cpu	Number of CPU interrupt controller	1 to 4
trig_cfg	Type of event, a selection among two different kind of events	96 bits mask (only the lower Number of Events bits are valid)
cpu_rxev_en	Optional propagation of enabled events to dedicated CPU outputs	4 bits mask (only the lower Number of CPU bits are valid)
rxev_cfg	Mask bit vector for events enabled to CPU propagation	96 bits mask (only the lower Number of Events bits are valid)
tz_cfg	Mask bit vector for events which implement AHB5 TrustZone security protection	96 bits mask (only the lower Number of Events bits are valid)

Real case study: Event Controller IP

```
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;


  event cover_event_config_e;
};

  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



vPlan Hierarchy

VMETRICS

Ex	UNR	Name	Combined Average Grade	Combined Covered
		(no filter)	(no filter)	(no filter)
		VMETRICS	99.98% *	60188 / 60206 (99.97%)
		1 Functional requirements verification plan	99.98% *	60188 / 60206 (99.97%)
		1.1 HW Generics	99.78% *	1080 / 1098 (98.36%)
		1.1.1 HW Generics cover	99.57% *	1078 / 1096 (98.36%)
		1.1.1.1 nb_of_event	100% *	5 / 5 (100%)
		1.1.1.2 nb_of_cpu	100%	4 / 4 (100%)
		1.1.1.3 cpu_rxev_en	100%	10 / 10 (100%)
		1.1.1.4 trg_cfg	100%	1 / 1 (100%)
		1.1.1.5 rxev_cfg	100%	2 / 2 (100%)
		1.1.1.6 tz_cfg	100%	2 / 2 (100%)
		1.1.1.7 priv_cfg	100%	2 / 2 (100%)
		1.1.1.8 nb_ioports	100% *	5 / 5 (100%)
		1.1.1.9 Cross_dependency_trig_rxev	98.68%	345 / 350 (98.57%)
		1.1.1.10 Cross_dependency_contiguous_trig	98.68%	330 / 335 (98.51%)
		1.1.1.11 Cross_dependency_contiguous_rxev	97.89%	372 / 380 (97.89%)
		1.1.2 HW Generics checks	100%	2 / 2 (100%)
		1.2 AIEC registers access	100% *	45294 / 45294 (100%)
		1.3 Clocking strategy	100%	21 / 21 (100%)
		1.4 Input events trigger	100%	528 / 528 (100%)
		1.5 IOPORT inputs selection	100%	4128 / 4128 (100%)
		1.6 System wakeup on inputs events	100%	1705 / 1705 (100%)
		1.7 CPUn wakeup on inputs events	100%	3481 / 3481 (100%)
		1.8 Configurable events output interrupt to all CPU's	100%	672 / 672 (100%)
		1.9 CPUn output event on inputs events	100%	3279 / 3279 (100%)

Configuration metrics approach results

vPlan Hierarchy

C7AMBA_AIEC_V2_2_VMETRICS

Ex	UNR	Name	Combined Average Grade	Combined Covered
		(no filter)	(no filter)	(no filter)
		C7AMBA_AIEC_V2_2_VMETRICS	98.66% *	60199 / 60206 (99.99%)
		1 Functional requirements verification plan	98.66% *	60199 / 60206 (99.99%)
		1.1 HW Generics	99.92% *	1091 / 1098 (99.36%)
		1.1.1 HW Generics cover	99.83% *	1089 / 1096 (99.36%)
		1.1.1.1 nb_of_event	100% *	5 / 5 (100%)
		1.1.1.2 nb_of_cpu	100%	4 / 4 (100%)
		1.1.1.3 cpu_rxev_en	100%	10 / 10 (100%)
		1.1.1.4 trg_cfg	100%	1 / 1 (100%)
		1.1.1.5 rxev_cfg	100%	2 / 2 (100%)
		1.1.1.6 tz_cfg	100%	2 / 2 (100%)
		1.1.1.7 priv_cfg	100%	2 / 2 (100%)
		1.1.1.8 nb_ioports	100% *	5 / 5 (100%)
		1.1.1.9 Cross_dependency_trig_rxev	99.47%	348 / 350 (99.43%)
		1.1.1.10 Cross_dependency_contiguous_trig	99.21%	332 / 335 (99.1%)
		1.1.1.11 Cross_dependency_contiguous_rxev	99.47%	378 / 380 (99.47%)
		1.1.2 HW Generics checks	100%	2 / 2 (100%)
		1.2 AIEC registers access	88% *	45294 / 45294 (100%)
		1.3 Clocking strategy	100%	21 / 21 (100%)
		1.4 Input events trigger	100%	528 / 528 (100%)
		1.5 IOPORT inputs selection	100%	4128 / 4128 (100%)
		1.6 System wakeup on inputs events	100%	1705 / 1705 (100%)
		1.7 CPUn wakeup on inputs events	100%	3481 / 3481 (100%)
		1.8 Configurable events output interrupt to all CPU's	100%	672 / 672 (100%)
		1.9 CPUn output event on inputs events	100%	3279 / 3279 (100%)

Real case study results

Approach kind	Number of generated configurations	Number of executed configurations	Total run time	Configuration coverage metrics	Full Coverage metrics
Full metrics driven	30	30	~ 180h	98.36%	99.97%
Configuration metrics driven	440	22	~ 125h	99.36%	99.99%

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.

Is this really the end…?

Weakness

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

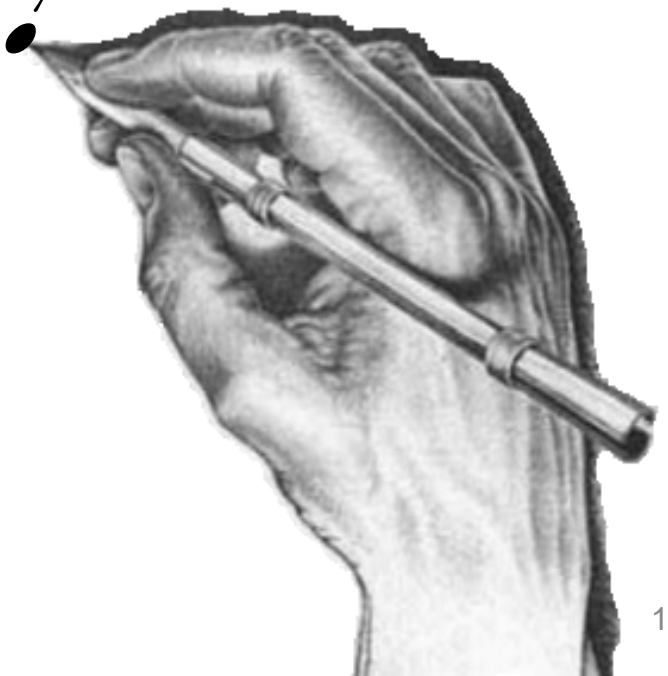
Future developments

- 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!

2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Thank you !



accellera SYSTEMS INITIATIVE