

Infineon

Munich - November 18, 2013

Compatible Qualification Metrics for Formal Property Checking

Holger Busch
Senior Staff Engineer Verification
Infineon Technologies



Never stop thinking

Overview

- Motivation
- Goals
- Qualification Approaches
- Onespin's Coverage Feature
- Certitude
 - General set-up
 - Coupling with Onespin
- Experience and Comparison
- Conclusions

Why Qualify Formal Property Sets ?

■ *“Formal properties are exhaustively checked!”*

➤ 100 % coverage ?

■ Yes:

– All input combinations implicitly checked by formal provers

■ No:

– Property assumptions constrain inputs

(better than constrained randomized simulation: not just seed)

– Property commitments cannot check all outputs

– Single property cannot cover all input-/output behaviour

➤ Properties are developed according to partitioned DUT-function

➤ Task: Guarantee completeness of partitioning

Goals

- Quality control
 - Assessment of formal property sets
- Formal verification management
 - Progress indication
 - Sign-Off Criteria
- Handling of mixed verification tool landscape
 - Directed & constraint driven randomized simulation
 - Formal property checking
- ISO26262 compliance of automotive μ C products
 - Traceability
 - Documentation of design and verification process

Qualification Approaches for Formal

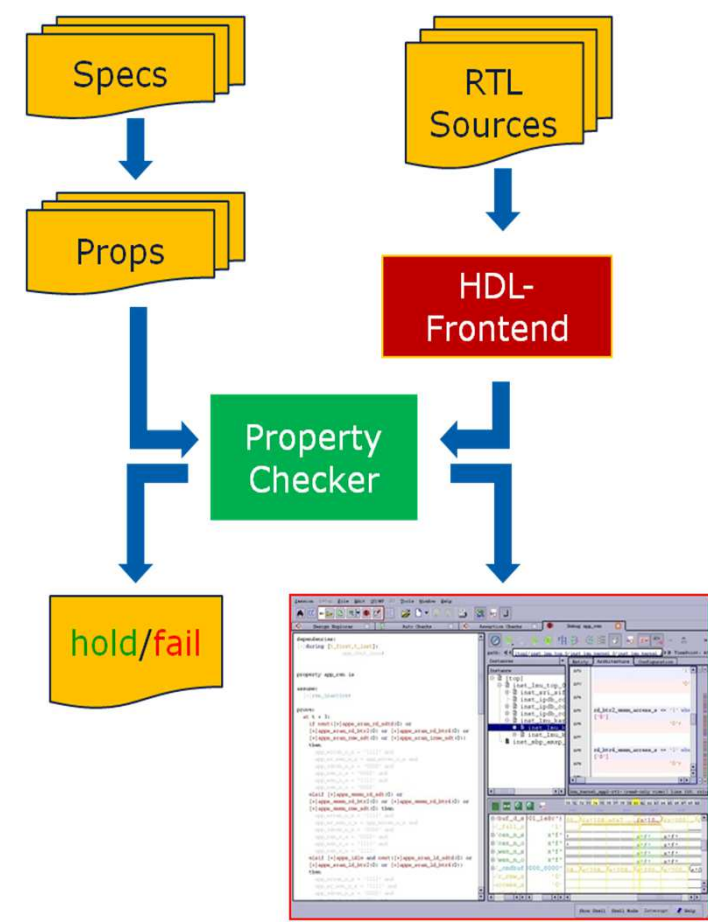
- Manual
 - Review of formal properties
- Formal completeness checks
 - Onespin's gap-free verification methodology
 - Strongest criterion, not related to simulation coverage metrics
- Formal witness generation
 - Simulation coverage for witness trace: line, branch
 - Quality of witness ?
- Design mutation
 - Onespin's built-in coverage feature Quantify
 - Link to test-bench qualification tool Certitude

Overview

- Motivation
- Goals
- Qualification Approaches
- **Onespin's Coverage Feature**
- Certitude
 - General set-up
 - Coupling with Onespin
- Experience and Comparison
- Conclusions

Onespin 360°™ MV

- Bounded model-checker
 - Various proof engines
- Property languages:
 - ITL (Interval Language) , SVA, PSL
- Linting
- Consistency checker
 - Dead-code, Stuck@signals, ...
- Property debugger
- Coverage
 - Formal completeness checker
 - Line & branch coverage: “Quantify“



stop thinking never

Onespin's Quantify Feature

■ Pre-analyses

- Formal-proof-based identification of dead, constrained, redundant code regions
- Code reachability by given property set

■ Observation coverage:

- Formal proof that code location (assignment) checked
- Code location observed when proof fails

■ User-guidance easy

- Push-button, focussing to code regions possible

■ Result presentation

- XML ~> UCDB-compatible
- HTML-visualisation



Onespin's Quantify Feature

stop thinking
never

Quantify MDV Overview - Konqueror (on vhlc690)

Location: /home/holger/ff_html/index.html

Quantify MDV Overview

Overview | Structural Coverage Overview | Structural Coverage by File | Assertion Coverage | File Status | Additional Information

Structural Coverage Overview

Status	Statements	Branches
1 covered	87 30.96%	29 39.73%
R reached	0 0.00%	0 0.00%
U unknown	0 0.00%	0 0.00%
OR unobserved	0 0.00%	0 0.00%
0 uncovered	182 64.77%	40 54.79%
OC constrained	0 0.00%	0 0.00%
OD dead	12 4.27%	4 5.48%
Sum quantify targets	281	73

Excluded Code Overview

Code Status	Statements	Branches
Xu excluded by user	0 0.00%	0 0.00%
Xr excluded redundant code	59 17.35%	15 17.05%
Xv excluded verification code	0 0.00%	0 0.00%
0/1/U quantify targets	281 82.65%	73 82.95%
Sum total code	340	88

Structural Coverage by File

File	Statements	Branches
ipdb_common_ff_rv0-rtl-a.vhd	3	3
ipdb_common_ff_rv0-rtl-a.vhd	3	3



Onespin's Quantify Feature

stop thinking
never

Quantify MDV File Result - Konqueror (on vihlc690)

Location Edit View Go Bookmarks Tools Settings Window Help

Location: /home/holger/ff_html/ipdb_common_ff_rv0-rtl-a.vhd.html

0C	constrained	0	0.00%	0	0.00%
0D	dead	0	0.00%	0	0.00%
Sum	quantify targets	3		3	

Annotated Source for ipdb_common_ff_rv0-rtl-a.vhd

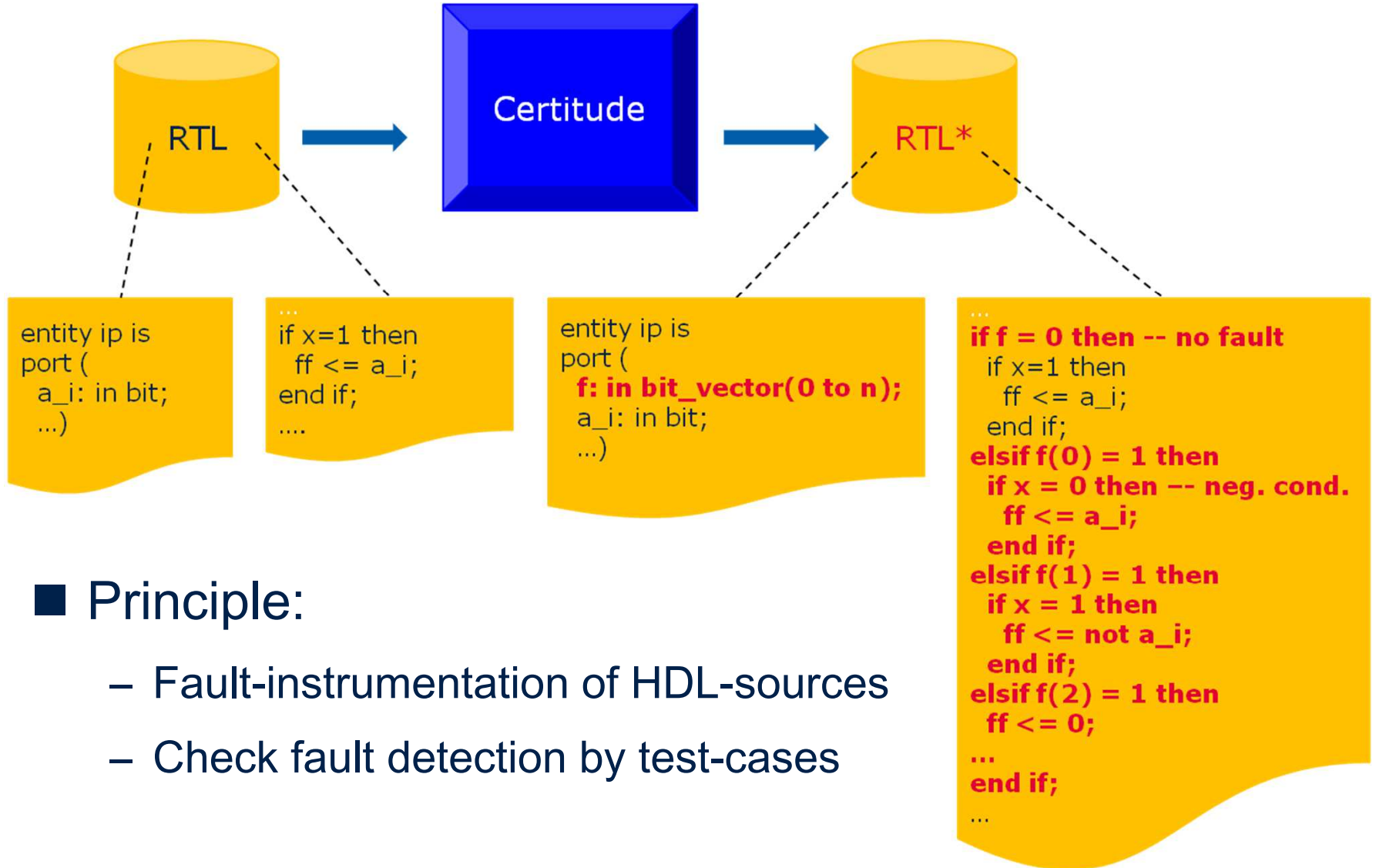
```
1  --#-----
2  --# File name:   ipdb_common_ff_rv0-rtl-a.vhd
3  --# Date:       November 2013
4  --# Author:     Holger Busch, IFAG ATV MC D
5  --# Contents:   Single FF with reset value 0 and write trigger
6  --# Copyright(c) Infineon Technologies AG 2013 all rights reserved
7  --#-----
8
9
10 ARCHITECTURE rtl of ipdb_common_ff_rv0 is
11   SIGNAL q_rs: std_ulogic;
12
13 BEGIN -- rtl
14
15   wr_p: PROCESS(clk_i, reset_n_i)
16   BEGIN
17     IF reset_n_i = '0' THEN
18       q_rs <= '0';
19     ELSIF clk_i'EVENT and clk_i = '1' THEN
20       IF wr_i = '1' THEN
21         q_rs <= d_i;
22       END IF;
23     END IF;
24   END PROCESS wr_p;
25
26   d_o <= q_rs;
27
28 END rtl; -- ipdb_common_ff_rv0
```

17	IF reset_n_i = '0' THEN	0
18	q_rs <= '0';	0
19	ELSIF clk_i'EVENT and clk_i = '1' THEN	1
20	IF wr_i = '1' THEN	1
21	q_rs <= d_i;	1
26	d_o <= q_rs;	1

Overview

- Motivation
- Goals
- Qualification Approaches
- Onespin's Coverage Feature
- **Certitude**
 - **General set-up**
 - Coupling with Onespin
- Experience and Comparison
- Conclusions

Certitude: Fault Instrumentation



■ Principle:

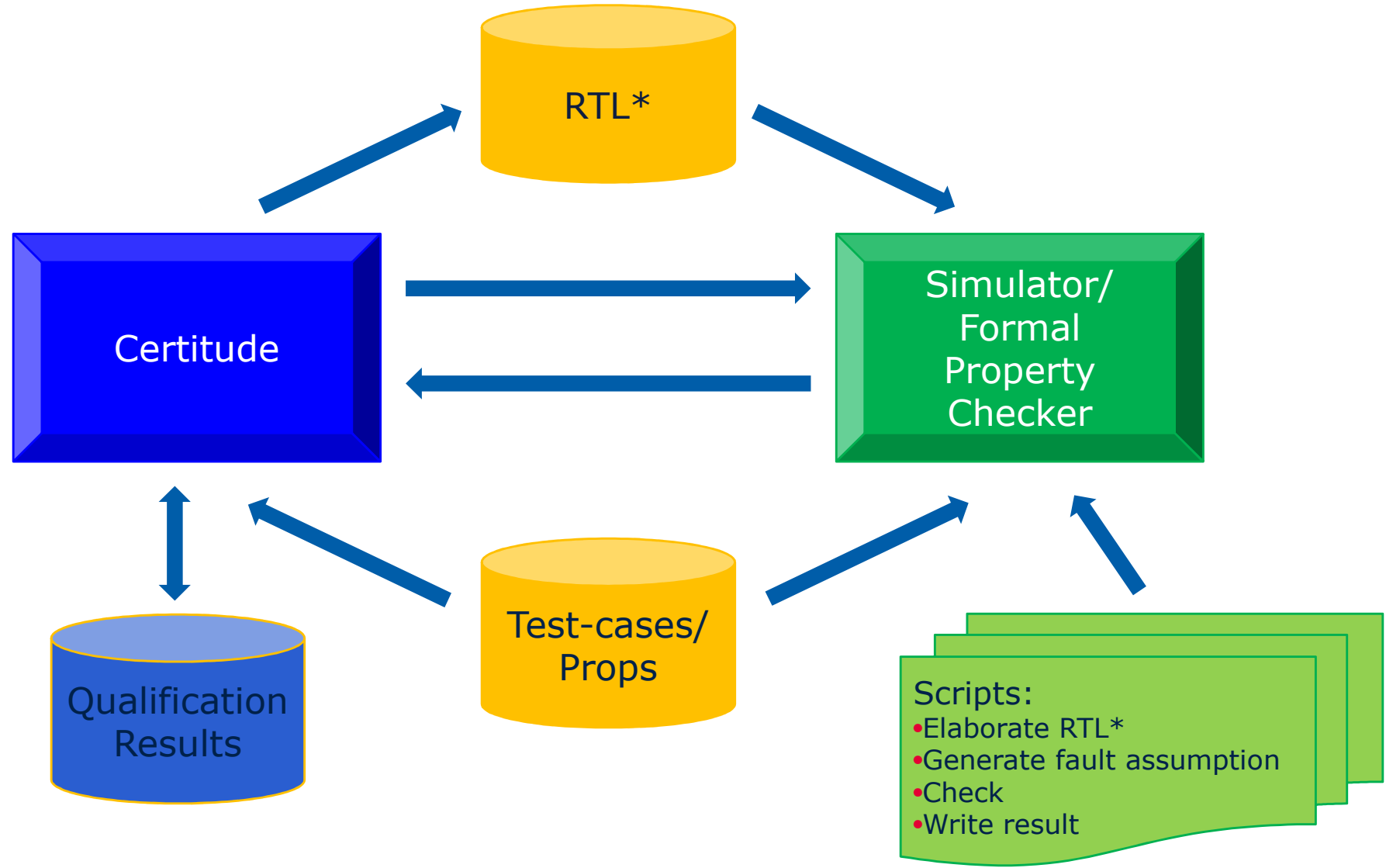
- Fault-instrumentation of HDL-sources
- Check fault detection by test-cases

stop thinking never

Certitude: Qualification Phases

- Modelling phase: RTL-code instrumentation by Certitude
 - Different fault models injected into RTL code
 - Top-level entity with additional input vector for individual activation
- Activation phase: Each test-case run once:
 - Activation: test-case stimulus activates fault condition
 - Propagation: fault visible at observation points (DUT interface)
 - Detection phase: Analyses for pairs of {fault test-case}:
 - Detection: fail of test-case instead of pass
 - Fault-sets: $F_{\text{injected}} \supseteq F_{\text{activated}} \supseteq F_{\text{propagated}} \supseteq F_{\text{detected}}$
 - Iterative detection controlled by Certitude:
- Statistical Approach by Certitude:
 - Metrics computation for statistical samples
- Application to Formal Properties
 - Iterative invocation of property checker for {fault / formal-property} pairs instead of simulator for {fault / test-case} pairs

Certitude Qualification Flow



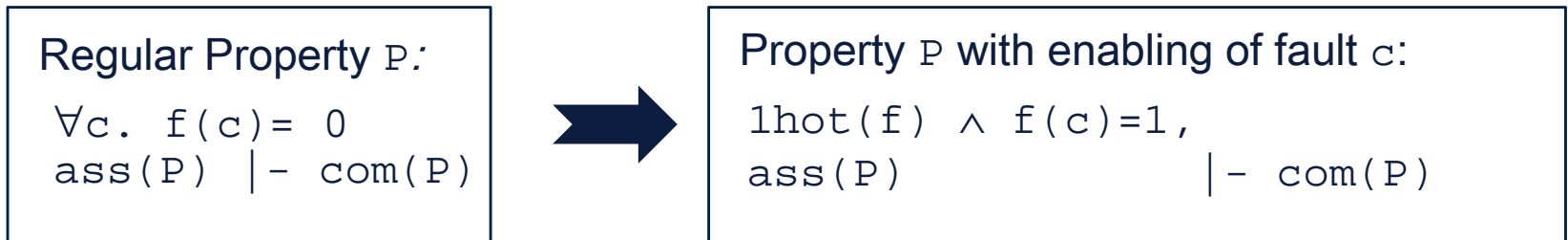
Overview

- Motivation
- Goals
- Qualification Approaches
- Onespin's Coverage Feature
- **Certitude**
 - General set-up
 - **Coupling with Onespin**
- Experience and Comparison
- Conclusions

Certitude \leftrightarrow Onespin

■ Iterative procedure:

- Let Certitude select:
 - Property P from set of qualification properties
 - Fault c from current set of non-detected faults
- Add fault assumption to regular property



- Check fault- c -enabled Property P in property checker
- Return proof result + run-time to Certitude
 - Fail: fault c detected by Property P
- Repeat until Certitude is finished:
 - All faults detected
 - or
 - All {fault / property}-pairs exercised for non-detected faults

Overview

- Motivation
- Goals
- Qualification Approaches
- Onespin's Coverage Feature
- Certitude
 - General set-up
 - Coupling with Onespin
- Experience and Comparison
- Conclusions

Experience and Comparison

Quantify	Certitude <-> Onespin Qualification
Added value: Coverage results useful for productive verification projects	
Design size: Modules with several 10 k locs manageable	
Usage: easy	Usage: less easy at the beginning: Set-up for Certitude and FPC required
Fault injection: elaborated model	Fault injection: RTL design
User control: Code regions (focus, skip, exclude) Property set	User control: rich configurability Code inclusion, fault types, density, properties; instance- or module-based
Restartability: yes (longer setup time)	Restartability: yes (short set-up time)
Compatibility with simulation: mergeable	Compatibility with simulation: 1:1
Maturity: Product feature, potentials for improvements	Maturity: Certitude available for many years; recently: scripting for IFX-internal usage for performance optimization
Licences: regular Onespin prover licenses	Licences: Onespin prover licenses + Certitude license

stop thinking never

Overview

- Motivation
- Goals
- Qualification Approaches
- Onespin's Coverage Feature
- Certitude
 - General set-up
 - Coupling with Onespin
- Comparison
- **Conclusions**

Conclusions

- Two simulation-compatible FPC qualification methods for productive usage
- Both handle big IPs and property sets
- Usage strongly recommended!
- Two different paradigms:
 - Integrated in property checking environment: Quantify
 - Efficient, but closed and vendor-specific
 - Metrics similar to simulation coverage
 - Coupling of separate tools: Certitude-Onespin/Simulators...
 - Open for customization
 - Exactly same metrics for FPC and simulation
- Potential synergies

Questions ?