

Using PSS: An Automotive Case Study

Ben Sutton (IFX)

Leo Matturi (IFX)

Kumar Shankar Dass Dilip (TVS)

Gopalakrishnan Balasubramanian (TVS)

Infiniteon Technologies UK Ltd.

Test and Verification Solutions Ltd



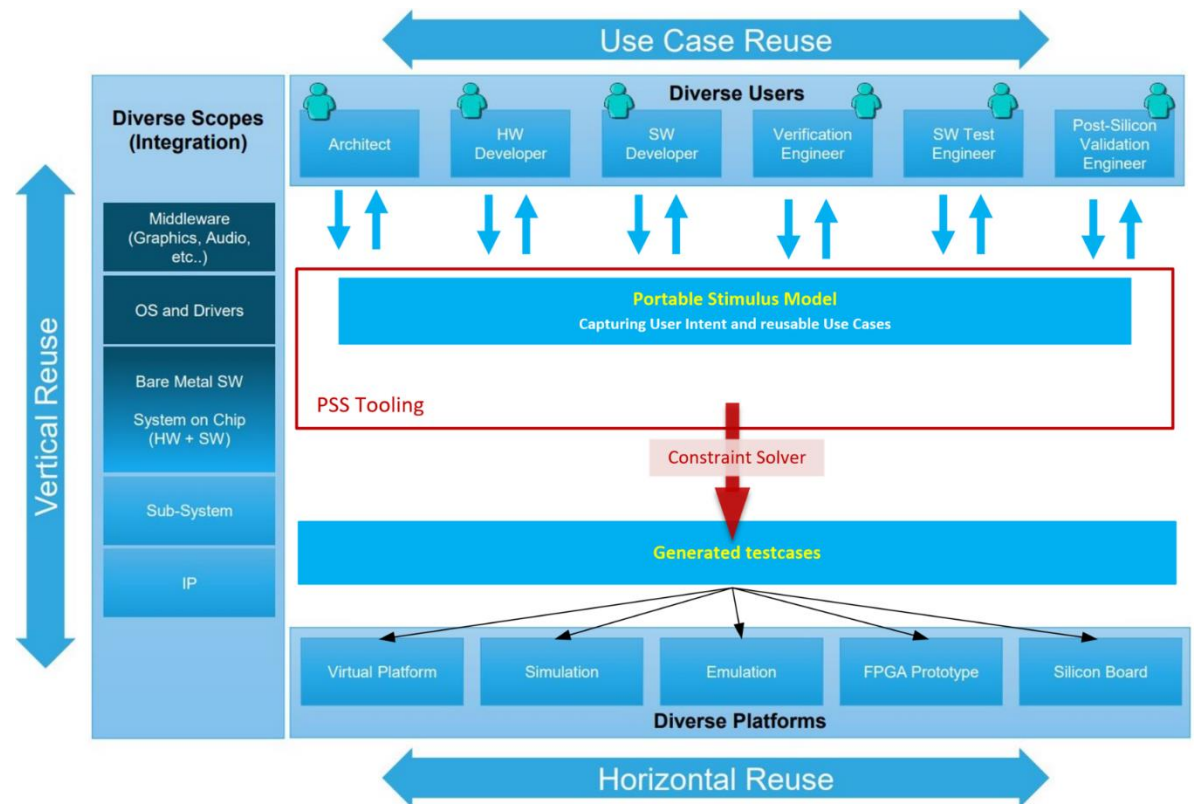
Motivation

- The AURIX™ microcontroller family is targeted at Automotive applications, both for Powertrain and Safety. It contains a comprehensive on-chip trace solution to aid debug.
- The TRACE IP supports concurrent trace of many different on-chip sources – via a complex multiplexing infrastructure.
- Functional verification of correct integration at subsystem and SOC level is a big challenge

File	Device	Time	Trace	Origin	Data	Operation	Address	Symbol	Label	SO	Comment	%s	Trace	CPU/TEMP
0	786542340	27	1	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
1	786542340	28	2	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
2	786542340	29	3	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
3	786542340	30	4	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
4	786542340	31	5	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
5	786542340	32	6	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
6	786542340	33	7	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
7	786542340	34	8	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
8	786542340	35	9	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
9	786542340	36	10	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
10	786542340	37	11	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
11	786542340	38	12	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
12	786542340	39	13	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
13	786542340	40	14	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
14	786542340	41	15	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
15	786542340	42	16	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
16	786542340	43	17	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
17	786542340	44	18	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
18	786542340	45	19	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
19	786542340	46	20	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
20	786542340	47	21	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
21	786542340	48	22	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
22	786542340	49	23	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
23	786542340	50	24	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
24	786542340	51	25	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
25	786542340	52	26	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
26	786542340	53	27	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
27	786542340	54	28	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
28	786542340	55	29	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
29	786542340	56	30	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
30	786542340	57	31	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
31	786542340	58	32	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
32	786542340	59	33	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
33	786542340	60	34	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
34	786542340	61	35	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
35	786542340	62	36	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
36	786542340	63	37	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
37	786542340	64	38	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
38	786542340	65	39	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
39	786542340	66	40	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
40	786542340	67	41	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
41	786542340	68	42	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
42	786542340	69	43	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
43	786542340	70	44	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
44	786542340	71	45	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
45	786542340	72	46	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
46	786542340	73	47	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
47	786542340	74	48	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
48	786542340	75	49	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
49	786542340	76	50	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
50	786542340	77	51	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
51	786542340	78	52	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
52	786542340	79	53	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
53	786542340	80	54	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
54	786542340	81	55	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
55	786542340	82	56	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
56	786542340	83	57	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
57	786542340	84	58	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
58	786542340	85	59	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
59	786542340	86	60	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
60	786542340	87	61	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
61	786542340	88	62	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
62	786542340	89	63	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
63	786542340	90	64	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
64	786542340	91	65	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
65	786542340	92	66	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
66	786542340	93	67	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
67	786542340	94	68	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
68	786542340	95	69	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
69	786542340	96	70	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
70	786542340	97	71	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
71	786542340	98	72	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
72	786542340	99	73	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
73	786542340	100	74	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
74	786542340	101	75	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
75	786542340	102	76	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
76	786542340	103	77	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
77	786542340	104	78	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
78	786542340	105	79	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
79	786542340	106	80	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
80	786542340	107	81	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
81	786542340	108	82	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
82	786542340	109	83	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
83	786542340	110	84	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
84	786542340	111	85	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
85	786542340	112	86	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
86	786542340	113	87	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
87	786542340	114	88	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
88	786542340	115	89	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				
89	786542340	116	90	CPU0	CPU0	300 STATE	78018C10	CPU0	ISR1-EDU1	1				

Challenge

- Existing methodology comprises platform tests manually coded in C, coupled with scripted checking to verify trace output.
- Disadvantages of this approach:
 - Maintenance and reuse of these test suites across multiple chip derivatives is both time consuming and expensive.
 - Test programmer has to manage system resources – avoiding conflicts between code running on multiple CPUs in parallel
 - Legacy scripting checking flow is hard to maintain and not scalable.
- Need to improve efficiency for test generation and reuse – investigated testcase automation solutions.
- The Accellera Portable Test & Stimulus Standard (PSS) gives a framework to define your high level verification intent in a model (stimulus and test scenarios).
- Verification engine generates testcases targeting different integration levels, platforms and chip derivatives.
- Variety of PSS tools available
- Chose Cadence Perspec System Verifier PSS tool – already in use at Infineon.
- Intention was to automate our testcase generation for reuse in:
 - Different derivatives.
 - Different integration levels (Subsystem, SOC).
 - Different platforms (simulation, emulation, validation).



Solution

- Created a model of the TRACE functionality.
- Written in Perspec SLN (System-Level Notation), one of the bases for Accellera PSS.
- Existing models for other key system IP, such as the CPUs and DMA already exist at Infineon. Can be used in conjunction with the new TRACE model to provide trace stimuli.

- Example constraint code in the TRACE model:

```
#define MCDS_POBX_SRC_LIST CPU0,CPU1,CPU2,CPU3,CPU4,CPU5,LMU0,OLDA,OTGB,OTGBM,NONE;
```

```
#define MCDS_POBY_SRC_LIST CPU1,CPU2,CPU3,CPU4,CPU5,OTGBM,NONE;
```

```
#define MCDS_POBZ_SRC_LIST CPU0,NONE;
```

```
#define MCDS_CPUX_SRC_LIST CPU0,CPU1,CPU2,CPU3,CPU4,CPU5;
```

```
action pobx_a like ob_base {
```

```
    constraint ob_tpy == OB_POBX;
```

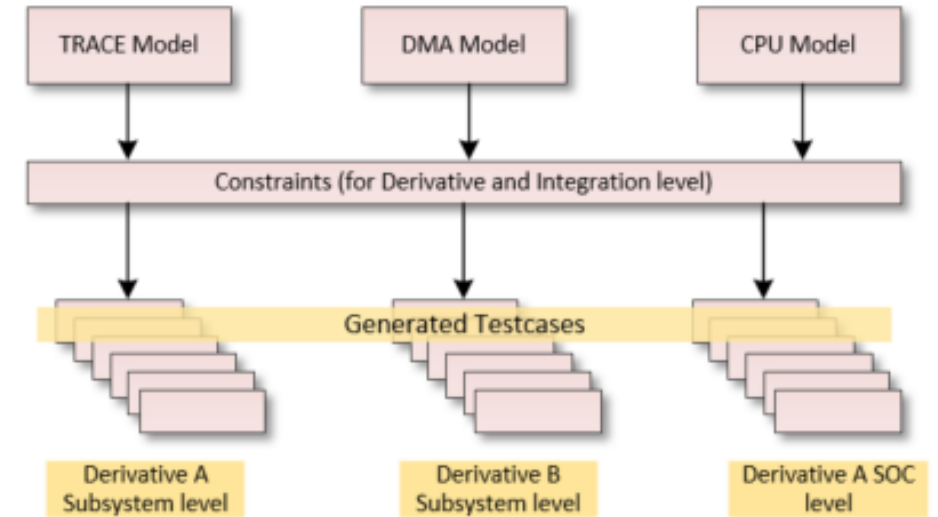
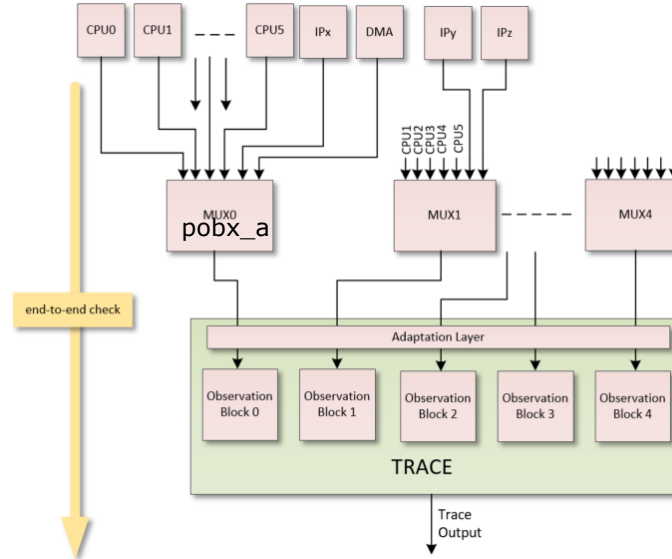
```
    constraint src != NONE => trc_in.idx == ob_tpy.as_a(uint);
```

```
    constraint tmux_out.ob_tpy == OB_POBX;
```

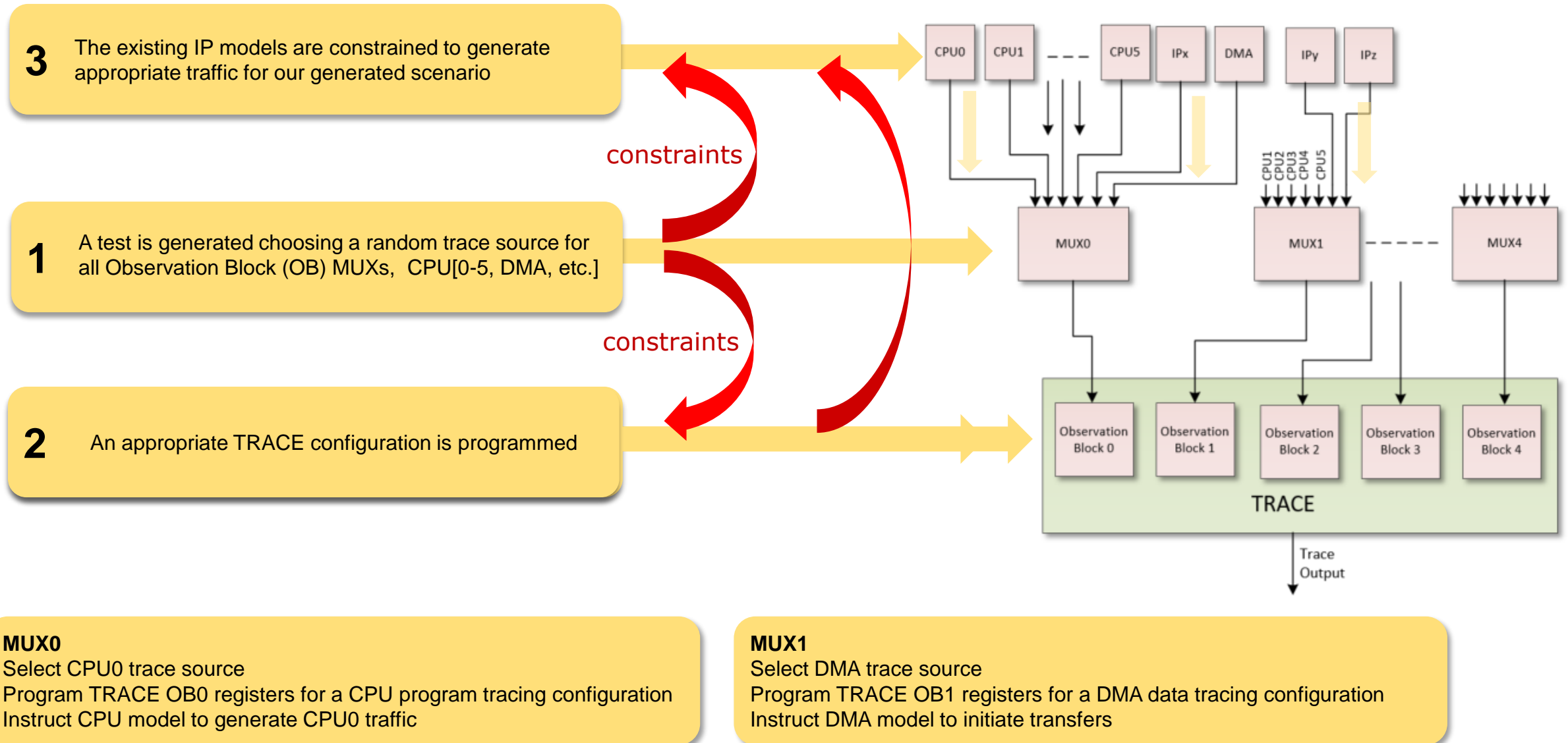
```
    constraint src in [MCDS_POBX_SRC_LIST];
```

```
};
```

```
constraint (src in [MCDS_CPUX_SRC_LIST]) => tmux_out.tcmux_tpy in [CPU_DTRACE, CPU_MEMSLAVE];
```

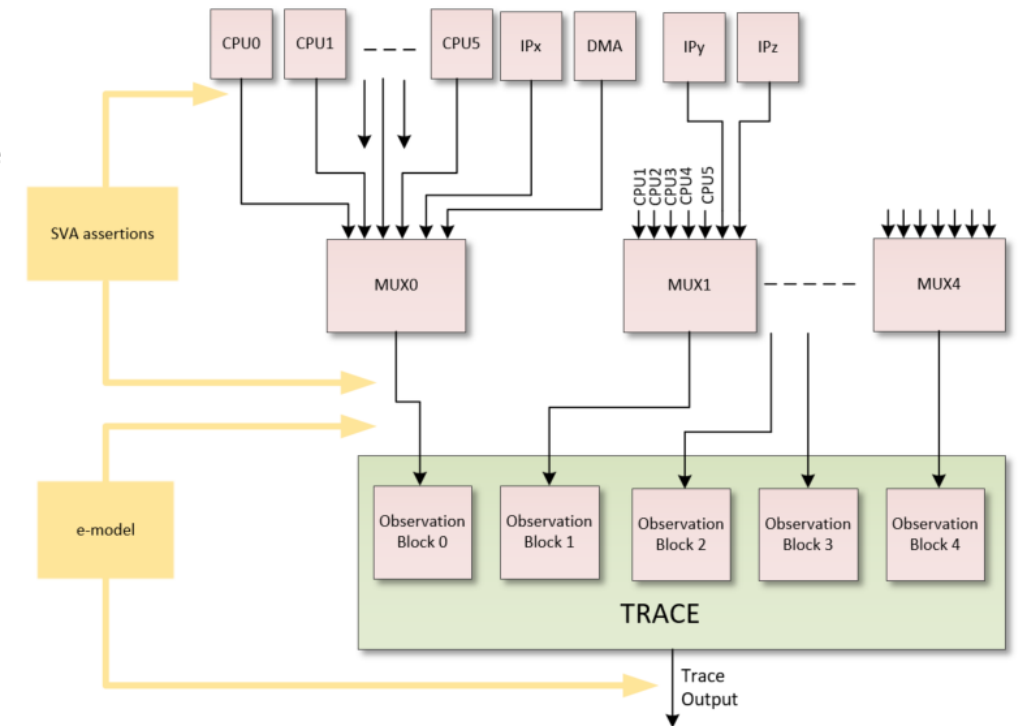


TRACE Model – Implementation example



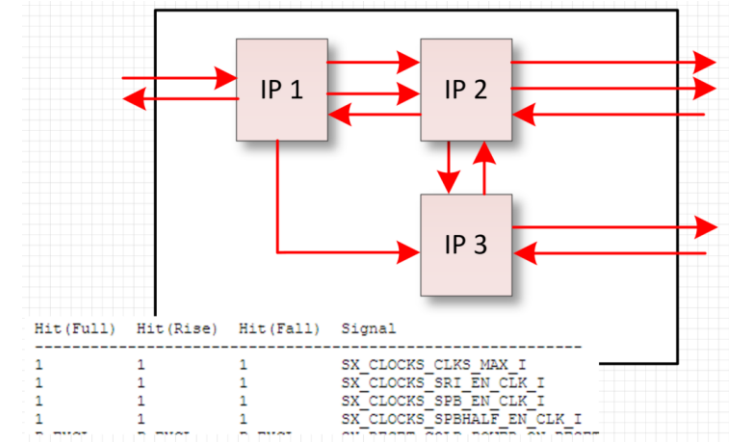
Checking Flow

- Easily generate hundreds of tests but legacy scripting based verification flow is not scalable.
- SVA Assertions are used to prove trace connectivity
- The TRACE IP testbench environment is a constrained random environment utilising Specman *e*, with a reference model enabling end-end verification of the TRACE flow.
- The IP reference model is instantiated at the integration level in passive mode to verify the trace flow through the TRACE IP.
- Other PSS models in our integration verification include checking, however the TRACE checking is independent.
- Existing Verification IP model was available for reuse – saving effort.
- Verification IP model is simulated at integration level, potentially highlighting wrong assumptions made in IP level modelling.



Coverage

- Basic metric for integration verification is Design Implementation Toggle Coverage, ensuring all IP interfaces have been exercised.
- This only provides a measure of activation of the *implemented* features/connectivity and does not identify those missing.
- On top we specify Functional Coverage, ensuring system features are exercised and interesting scenarios covered.
- The PSS tooling provides Generation Functional Coverage, allowing us to know before regressing what will be stimulated and hence if all required scenarios will be covered by a generated test suite.
- This Generation Coverage is then converted into Runtime Coverage



Type (default scope): pobx_a

Overall Local Grade: 25.45% | Functional Local Grade: 25.45% | CoverGroup Local Grade: 25.45% | Assertion

Cover groups		
Name	Overall Average Grade	Overall Covered
(no filter)	(no filter)	(no filter)
generated	50.91%	20 / 97 (20.62%)
ended	0%	0 / 97 (0%)

Bins		
Name	Overall Average Grade	Overall Covered
(no filter)	(no filter)	(no filter)
CPU0	100%	1 / 1 (100%)
CPU1	100%	1 / 1 (100%)
CPU2	100%	1 / 1 (100%)
CPU3	100%	1 / 1 (100%)
CPU4	100%	1 / 1 (100%)
CPU5	100%	1 / 1 (100%)
LMU0	100%	1 / 1 (100%)
OLDA	100%	1 / 1 (100%)
OTGB	0%	0 / 1 (0%)
OTGBM	0%	0 / 1 (0%)
NONE	100%	1 / 1 (100%)

Items		
Name	Overall Average Grade	Overall Covered
(no filter)	(no filter)	(no filter)
src	81.82%	16 / 19 (84.21%)
cpu_memslave_src	16.67%	1 / 6 (16.67%)
cpu_memslave	100%	1 / 1 (100%)
A# cross_src_cpu_memslave	54.55%	3 / 5 (60%)
A# cross_src_cpu_memslave_src	1.52%	1 / 66 (1.52%)

Trace path Functional Coverage

Cover Gro.. Assertions

Cover groups			
Name	Overall Average Grade	Overall Covered	Score
(no filter)	(no filter)	(no filter)	(no filter)
generated	52.58%	13 / 25 (52%)	13
ended	0%	0 / 25 (0%)	0

Bins			
Name	Overall Average Grade	Overall Covered	Score
(no filter)	(no filter)	(no filter)	(no filter)
DCU	0%	0 / 1 (0%)	0
DTU	100%	1 / 1 (100%)	46
PTU	100%	1 / 1 (100%)	13
OTU	0%	0 / 1 (0%)	0
WTU	0%	0 / 1 (0%)	0
TC_ACT	0%	0 / 1 (0%)	0
TSU_REL	0%	0 / 1 (0%)	0

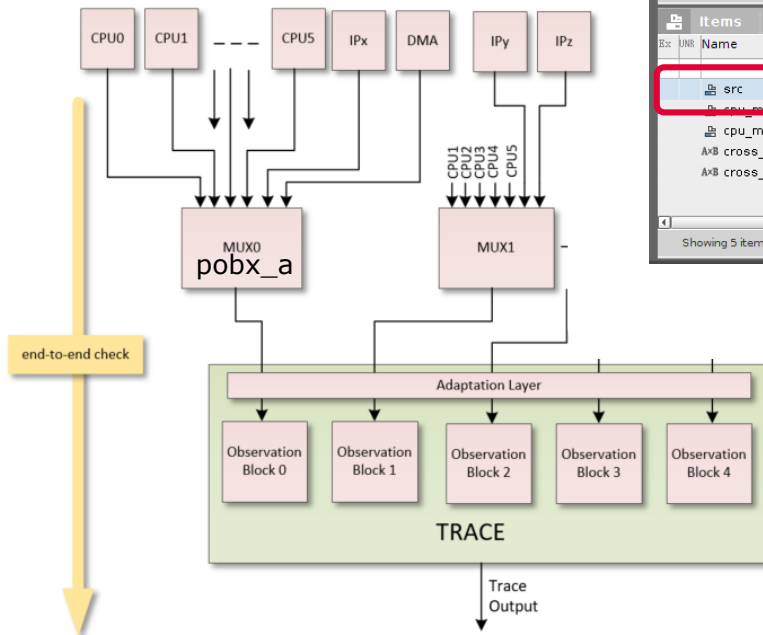
Items		
Name	Overall Average Grade	Overall Covered
(no filter)	(no filter)	(no filter)
ob_sel	100%	1 / 1 (100%)
pobx_trace_unit	28.57%	2 / 7 (28.57%)
pobx_idx_sel	29.17%	2 / 7 (29.17%)

Details		
Col #	Name	Value
	(no filter)	(no filter)
	Cover Group	generated
	CoverGroup Average Grade	28.5714%

TRACE Configuration Functional Coverage

Coverage

- Toggle coverage can only be measured in simulation but Generation Functional Coverage (at interface level) gives a good idea of expected toggle coverage.



Type (default scope): pobx_a

Overall Local Grade: 25.45%

Functional Local Grade: 25.45%

CoverGroup Local Grade: 25.45%

Assertion

Cover Gro...

Assertions

Cover groups

Ex	UNK	Name	Overall Average Grade	Overall Covered
		(no filter)	(no filter)	(no filter)
		generated	50.91%	20 / 97 (20.62%)
		ended	0%	0 / 97 (0%)

Showing 2 items

Items

generated

Ex	UNK	Name	Overall Average Grade
		src	81.82%
		cpu_memslave_src	16.67%
		cpu_memslave	100%
		AxB cross_src_cpu_memslave	54.55%
		AxB cross_src_cpu_memslave_sr...	1.52%

Showing 5 items

Bins

src

Abstract

Expand

Ex	UNK	Name	Overall Average Grade	Overall Covered
		(no filter)	(no filter)	(no filter)
		CPU0	100%	1 / 1 (100%)
		CPU1	100%	1 / 1 (100%)
		CPU2	100%	1 / 1 (100%)
		CPU3	100%	1 / 1 (100%)
		CPU4	100%	1 / 1 (100%)
		CPU5	100%	1 / 1 (100%)
		LMU0	100%	1 / 1 (100%)
		OLDA	100%	1 / 1 (100%)
		OTGB	0%	0 / 1 (0%)
		OTGBM	0%	0 / 1 (0%)
		NONE	100%	1 / 1 (100%)

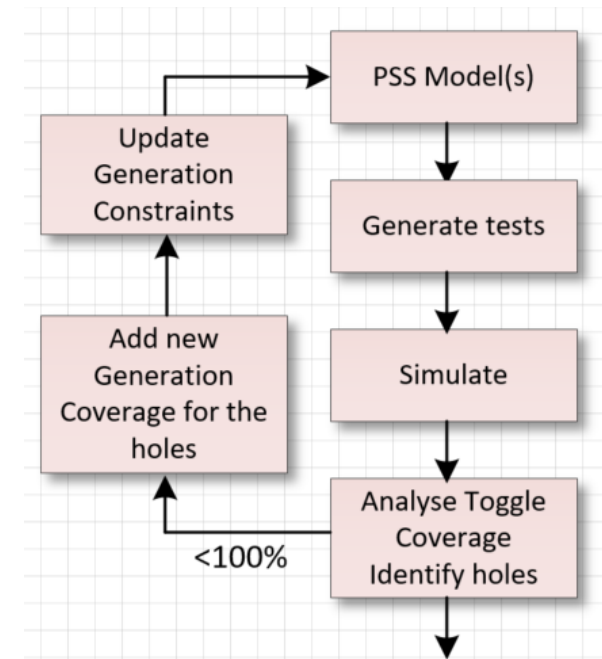
Showing 11 items

Details

src

Attributes

Source



- An iterative development loop allows stimuli generation to be updated for improved toggle coverage.
- e.g.
 - toggle coverage shows the error signals are not toggling
 - Add error generation to the model and add generation coverage of this
 - Then we will know in advance of regressing whether error signal toggle coverage will be 100%

Conclusions

- Development of the PSS model flow has shown real benefits, including quick setup of new product derivatives and reduced workload for test maintenance.
- Very quick to generate 100's of tests and associated functional coverage shows completeness.
- Easy to generate a lot of stimuli but we need robust checkers and coverage - Reuse of IP checkers enabled this
- Real benefit reusing existing PSS models of other IPs – can come from other subsystems or SOC level.
- Developers don't need to understand full functionality of the models being utilised.
- Future Work
 - As new trace sources are developed, their accompanying PSS models can be reused for stimuli in TRACE integration verification with minimal overhead.