Automated Specification Driven Verification by Generation of SystemVerilog Assertions

Ferdinando Pace, Senior ASIC Designer, SENSIRION AG, ferdinando.pace@sensirion.com

A1. ABSTRACT

The ATG tool is described that **automatically generates** both testbench and **SystemVerilog assertions** checking signal timing specifications in Excel format.

- The timing specifications can be **dependent on** programmable parameters.
- The generated SystemVerilog assertions **cover all** parameter **combinations.**
- **Multi-clocked** timing specifications are supported.
- A compact description for long timing sequences (e.g. periodic signals) is provided.
- ATG is **used** by SENSIRION **to verify** signal timings specified to drive an **A/D converter.**
- ATG can be used to **verify** signal timing specifications **of analog/mixed-signal or digital IPs.**

A2. PROBLEM: VERIFY COMPLEX SIGNAL TIMING

Analog/mixed-signal IPs often require **complex signal timing.**

Below it is described a timing sequence for an A/D converter using 4 phases (idle, reset, settling, Conversion) in which 3 signals (RESET, SETTLING, CHOP) are driven according to programmable parameters expressed in clock cycles.

The **verification effort increases** significantly depending on the number of parameter combinations allowed.

A3. VERIFICATION CHALLENGE

Typically, **the verification process** involves the following steps:

- The Specification Writer understands and writes the IP timing specification.
- The Design Engineer implements the RTL code generating the IP timing.
- The Verification Engineer verifies that the signal timing generated by the RTL code fulfills the specification for all possible parameter combinations.

This process is very critical because it relies on flawless communication and depends on error prone subjective generation and interpretation of the specification.

Several **verification methods** are available:

- **Visual comparison** of the timing diagram generated by the RTL design with the timing diagram produced by the Specification Writer: this visual method **is error prone by nature.**
- **Semi-automated comparison** (with specific EDA tools) of the simulation database generated by the RTL design with the simulation database generated by the IP level testbench: **the result analysis is difficult**.
- **Assertions development** (e.g. Verilog, SystemVerilog, PSL) to verify the timing diagram: the handcrafted development of assertions **is time consuming.**

A5. AUTOMATED VERIFICATION REQUIREMENTS

A rigorous flow from the objective specification to the verification is required.

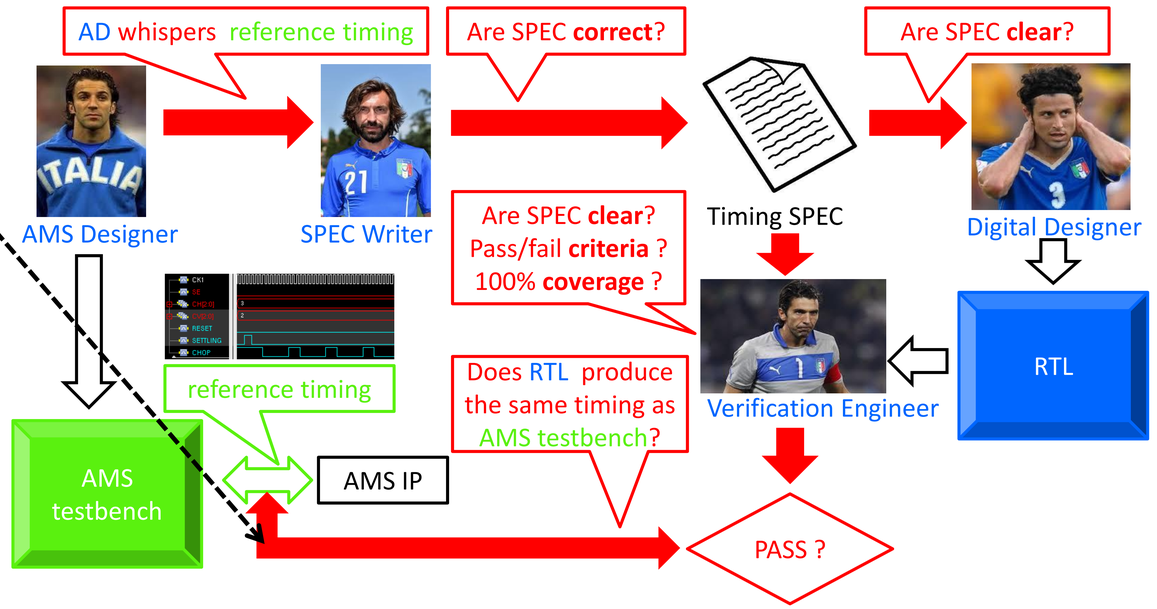
This flow is ideally implemented using an automated tool which satisfies the following requirements:

- **Unique**, objective, **unambiguous** description of specification
- **User friendly**, ideally an automated flow once the specification is available
- **Cheap**, open source tools
- **Easy to integrate** into existing project simulation environment
- **Flexible**, it accepts programmable on-chip parameters, multiple clock sources
- **Exhaustive**, all possible combinations for programmable on-chip parameters are covered
- **Self-checking**, automatically generated assertion statements can be tested stand-alone before being integrated into the simulation environment

A4. SPECIFICATION2VERIFICATION: HARD TEAMWORK!

- **Several players** involved editing and understanding the **specification**: **time consuming** & **error prone**
- **Several on-chip programmable parameters** affect the **specification**: **big effort** to reach **100% coverage**

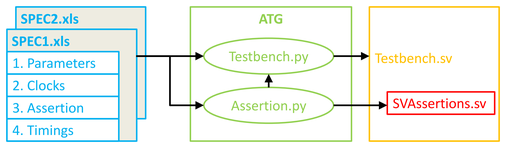
=> ATG is the solution !



B. SOLUTION

What is the Assertion&TestBenchGenerator tool ?

- INPUT: signal timing specification in XLS format (4 sheets per XLS file)
- OUTPUT1: **SVA**ssertions describing the signal timing specification
- OUTPUT2: Testbench verifying the **SVA**ssertions



Why to use it ?

- Written in Python: open source
- Push-button Specification2Verification tool:
  - Assertion.py -i DSP -x SPEC1.xls -o SVAssertions.sv
  - Testbench.py -i DSP -x SPEC1.xls -x SPEC2.xls -o Testbench.sv
- Easy integration into existing simulation environment
- Verify complex, parametric and multi-clocked signal timing specifications
- Suitable for AMS IPs (e.g. A/D converter, switched cap filters)

C. INPUT SHEETS REQUIRED BY ATG

1. Parameters

IP	`DSP	`DSP	`DSP
SIGNAL NAME	SE	CV	CH
SIGNAL WIDTH		[2:0]	[2:0]
values	0	0	3
	1	1	5
		2	

Parameter combinations:

1. SE=0, CV=0, CH=3
2. SE=0, CV=0, CH=5
3. SE=0, CV=1, CH=3
4. SE=0, CV=1, CH=5
5. **SE=0, CV=2, CH=3**
6. SE=0, CV=2, CH=5
7. SE=1, CV=0, CH=3
8. SE=1, CV=0, CH=5
9. SE=1, CV=1, CH=3
10. SE=1, CV=1, CH=5
11. SE=1, CV=2, CH=3
12. SE=1, CV=2, CH=5

2. Clocks

CLOCK NAME	CK1
Phase delay	0ns
High pulse	5ns
Low Pulse	5ns

3. Assertion

A_CHOP

4. Timings (template)

DISABLE	TRIGGER	GROUP NAME	GROUP VALUE	ROW NAME	ROW VALUE	SIGNAL A	...	SIGNAL N	EVENT
condition	condition						...		E
					Rvalue	Svalue	...	Svalue	E
		Gname	Gvalue	Rname	Rvalue	Svalue	...	Svalue	E
		Gname	Gvalue	Rname	Rvalue	Svalue	...	Svalue	E
		...	...	...	...	...	...	...	...
		Gname	Gvalue	Rname	Rvalue	Svalue	...	Svalue	E
									E

Gvalue, Rvalue:
a) Function of Parameter Sheet
b) Sequence Repetition Expression:
[* N]
[* N : M]
[* N : $]
[= N]
[-> N]

Svalue:
a) Verilog value
b) don't care "-"

E is the **assertion** capturing event for the previous row
E is the **sequence** launching event for the current row

D. EXAMPLE

Timings sheet

DISABLE	TRIGGER	GROUP NAME	GROUP VALUE	ROW NAME	ROW VALUE	CHOP	SETTLING	RESET	EVENT
`DSP.RESET==1	$fell(Assertion_Trigger)								@(posedge `DSP.CK1)
				reset	[* 2]	1'b1	1'b1	1'b0	@(posedge `DSP.CK1)
				settling	SE+3	1'b1	1'b0	1'b0	@(posedge `DSP.CK1)
		Conversion	2**CV	chop_low	10-CH	1'b0	1'b0	1'b0	@(posedge `DSP.CK1)
		Conversion	2**CV	chop_high	CH	1'b1	1'b0	1'b0	@(posedge `DSP.CK1)
									@(posedge `DSP.CK1)

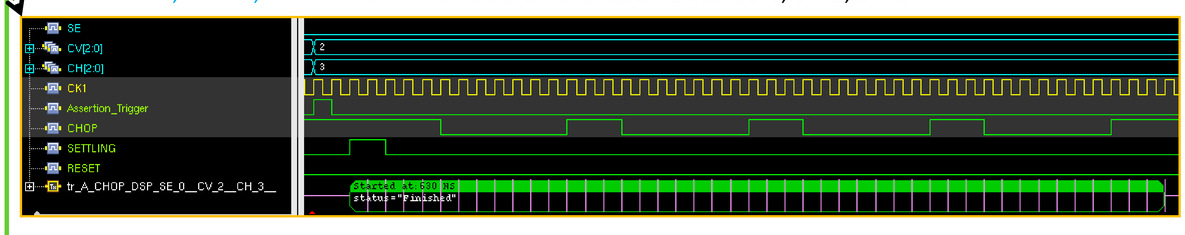
SystemVerilog assertion property generated by ATG from:

- **Timings** sheet
- **Parameters, Clocks , Assertion** sheets in Table C. Parameter values: SE=0, CV=2, CH=3

```
property A_CHOP_DSP_SE_0__CV_2__CH_3__;
disable iff (`DSP.RESET==1)
(@(posedge `DSP.CK1) ($fell(Assertion_Trigger) && `DSP.SE==0 && `DSP.CV==2 && `DSP.CH==3) |=>
   @(posedge `DSP.CK1)   ( CHOP==1'b1 && SETTLING==1'b1 && RESET==1'b0 ) [* 2] ##1
   @(posedge `DSP.CK1)   ( CHOP==1'b1 && SETTLING==1'b0 && RESET==1'b0 ) [* 3] ##1
   (@(posedge `DSP.CK1)   ( CHOP==1'b0 && SETTLING==1'b0 && RESET==1'b0 ) [* 7] ##1
    @(posedge `DSP.CK1)   ( CHOP==1'b1 && SETTLING==1'b0 && RESET==1'b0 ) [* 3])[*4]
); endproperty
```

Signal waveforms of the testbench generated by ATG from:

- **Timings** sheet above.
- **Parameters, Clocks , Assertion** sheets in Table C. Parameter values: SE=0, CV=2, CH=3



E. RESULTS

- ATG was used to verify the timing of **chopping** signals, dependent on **several programmable parameters** (reset time, settling time, oversampling ratio, chopping frequency) that were specified to drive an A/D converter in a SENSIRION chip.
- ATG demonstrated to **lower the time required to reach 100% verification coverage**: SystemVerilog properties from 3 XLS files were automatically generated to allow the verification of 256 parameter combinations