

CDT techday'e hoş geldiniz

Questa One

Transforming Verification from Complexity To Clarity

Faiçal Chtourou

Filed Application Engineer



SIEMENS

Agenda

Why, What **Questa One**

Questa One in a nutshell

- **Questa One** Sim
- **Questa One** SFV
- **Questa One** VIQ
- **Questa One** Avery VIP

Deep Dive **Questa One** Sim

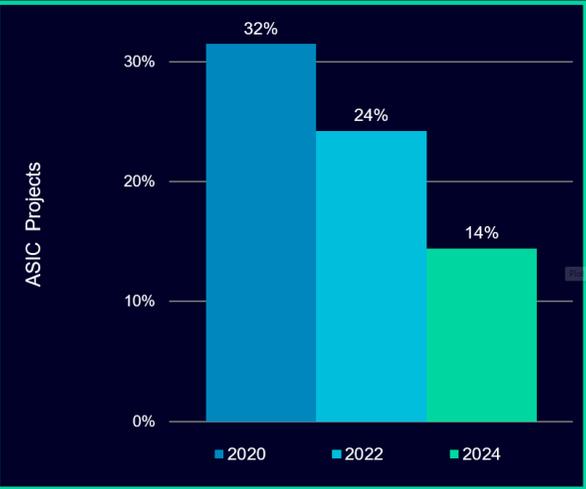
Why, What Questa One

Why Does It Matter?

First Silicon Success Declining

Decline in ASIC first silicon success and increase in FPGA no non-trivial bug escapes

ASIC First Silicon Success



FPGA No Bug Escapes

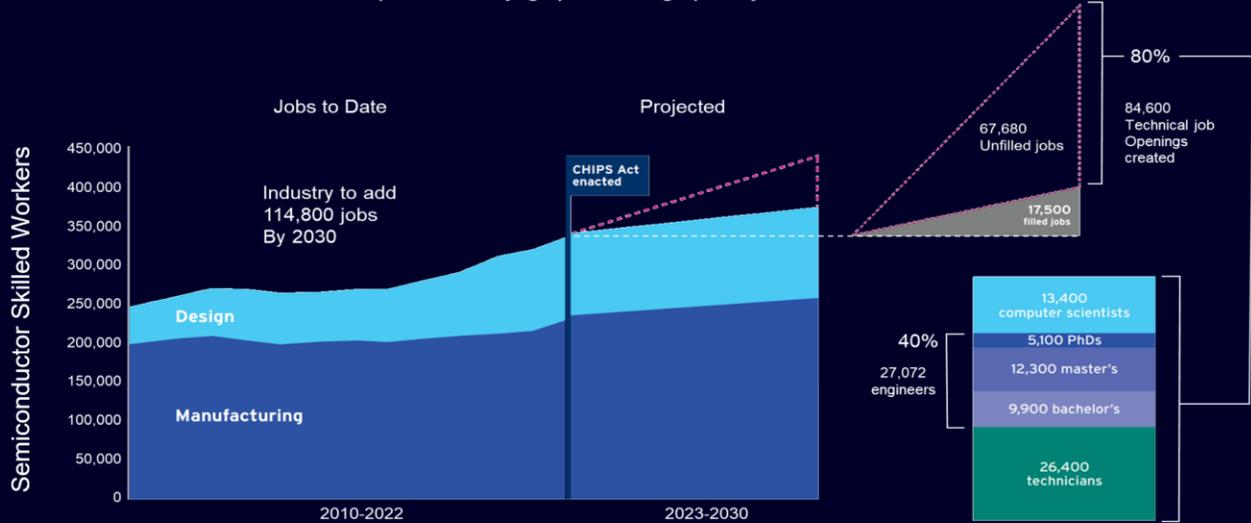


Source: Wilson Research Group and Siemens EDA, 2023 Functional Verification Study

Source: Siemens EDA and Wilson Research Group, 2024 Functional Verification Study

Insufficient Skilled Workforce

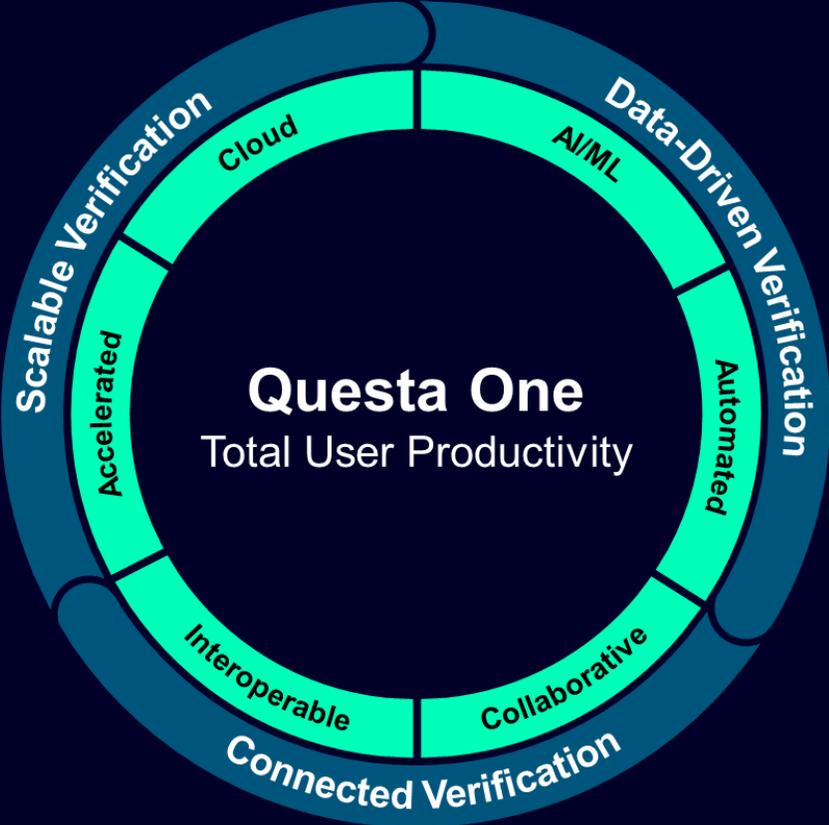
Insufficient workforce leads to productivity gap causing quality and schedule declines



Source: Semiconductor Industry Association (SIA), in partnership with Oxford Economics Study, July 2023

Questa One – Next Generation of Smart Verification Solution enabled by AI

Improving user productivity through the collective power of technology



Faster
Engines



Faster
Engineers



Fewer
Workloads



Delivering up to **5x** improved
Total User Productivity

Questa One Smart Verification Solution enabled by AI

Faster engines, faster engineers, fewer workloads

Questa One Sim

One engine with breakthrough performance for functional & fault sim for RTL, GLS, & DFT applications with parallel processing & profiling

Questa One SFV

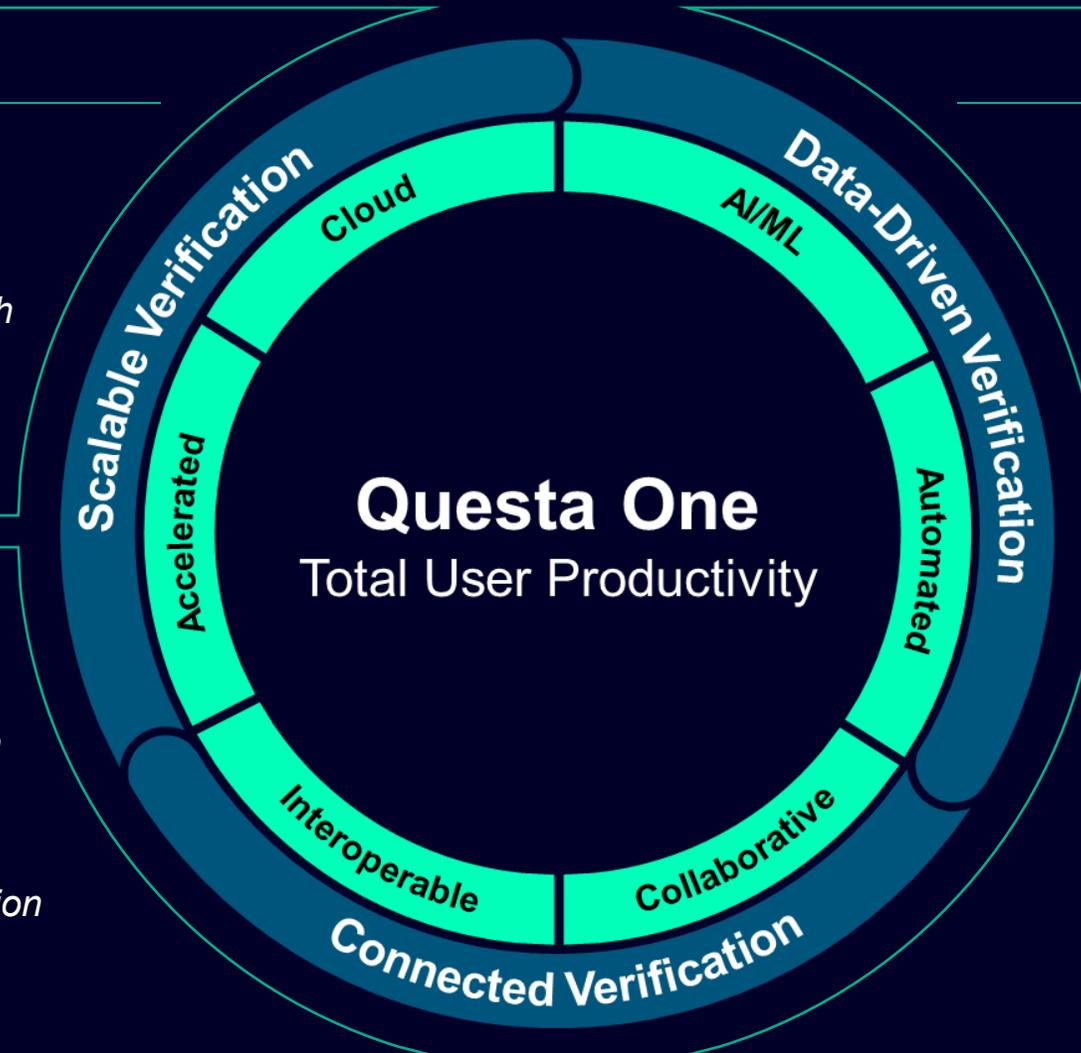
One stimulus free verification solution delivers total user productivity by combining static & formal analyses through AI, automation, & parallelization

Questa One Verification IQ

One smart common coverage solution that utilizes generative, analytic, & predictive AI to drive to verification closure faster, with fewer workloads

Questa One Avery VIP

One industry-leading VIP solution that supports 3DIC & chiplet verification from IP to SoC, seamlessly integrating simulation & emulation



Questa One

Verification IQ

Questa™ One Smart Verification

Faster engines, faster engineers and fewer workloads



Smart Debug

Intelligent Root-cause
VIQ Regression Navigator –
Signature, Bad-commit &
Root-cause Prediction,
Protocol Assist



Smart Verification

Smart Creation

Intelligent Assistance
*Property Assist, Docs Assist,
PSS Assist, Testplan Assist, Code Assist*



Smart Regression

Optimized Regression Cycles
VIQ Regression Navigator -
Failure, Smoke-test & Schedule
Prediction



Smart Engines

Intelligent Verification Cycles
QCX
Questasim Performance,
Coverage Optimization



Smart Analysis

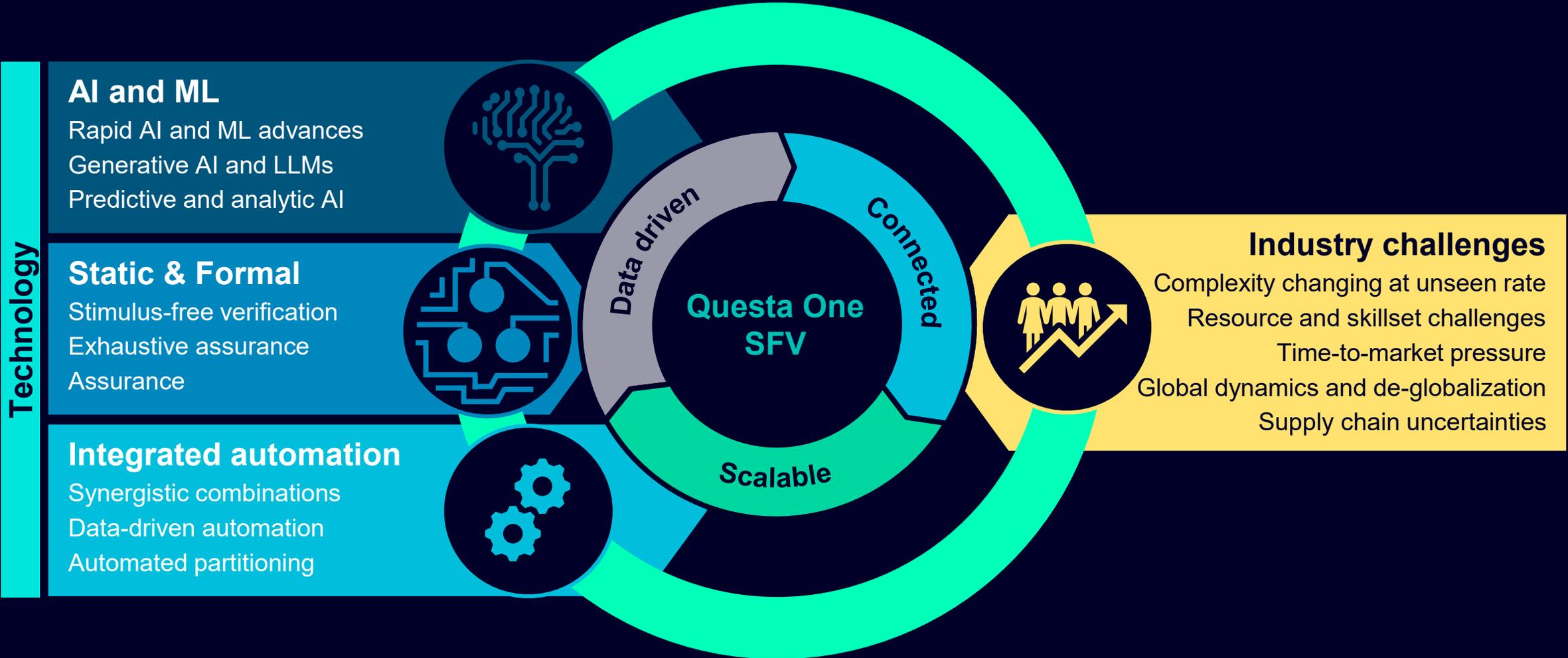
Intelligent Insights
*VIQ Coverage Analyzer,
CDC Assist & RDC Assist*



Questa One SFV

Introducing Questa One SFV: Purpose-built stimulus-free verification

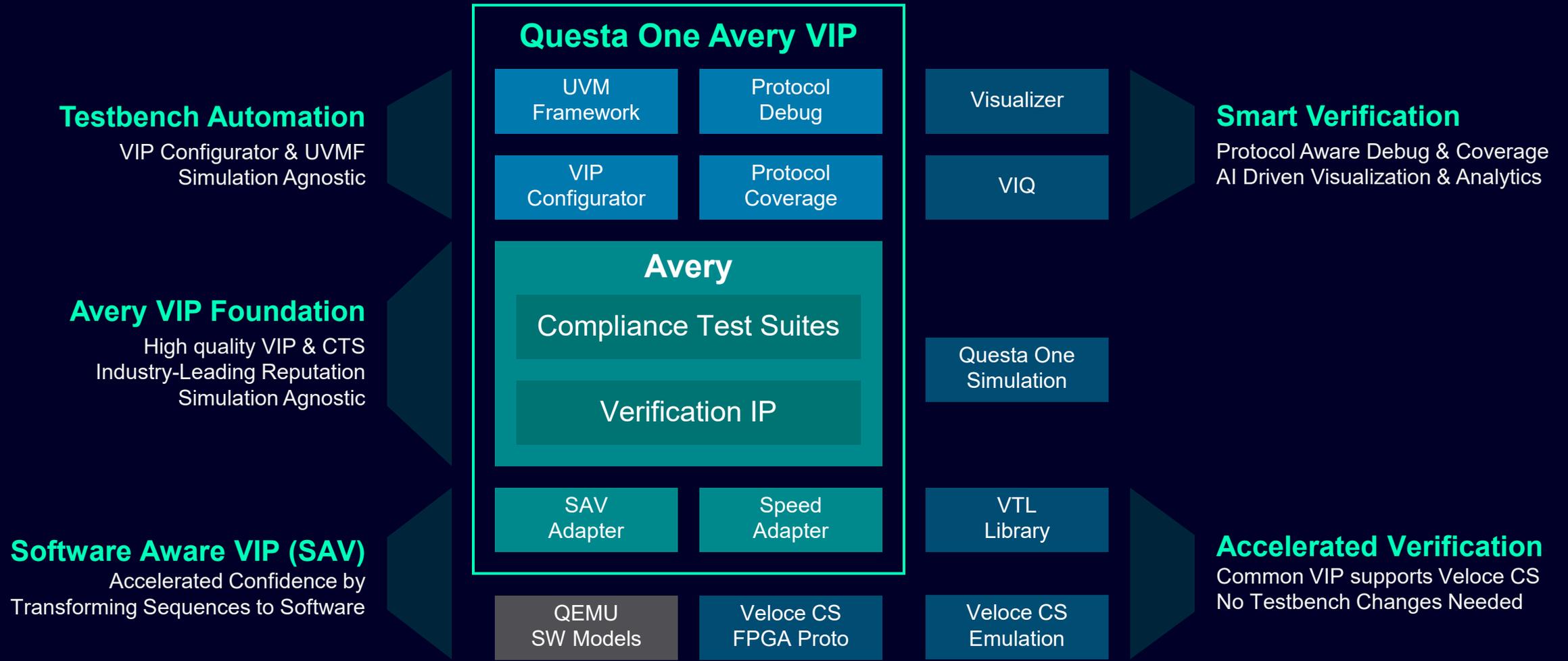
AI/ML, automation and integration enable new high-productivity flows and solutions



Questa One Avery VIP

Questa One Avery VIP

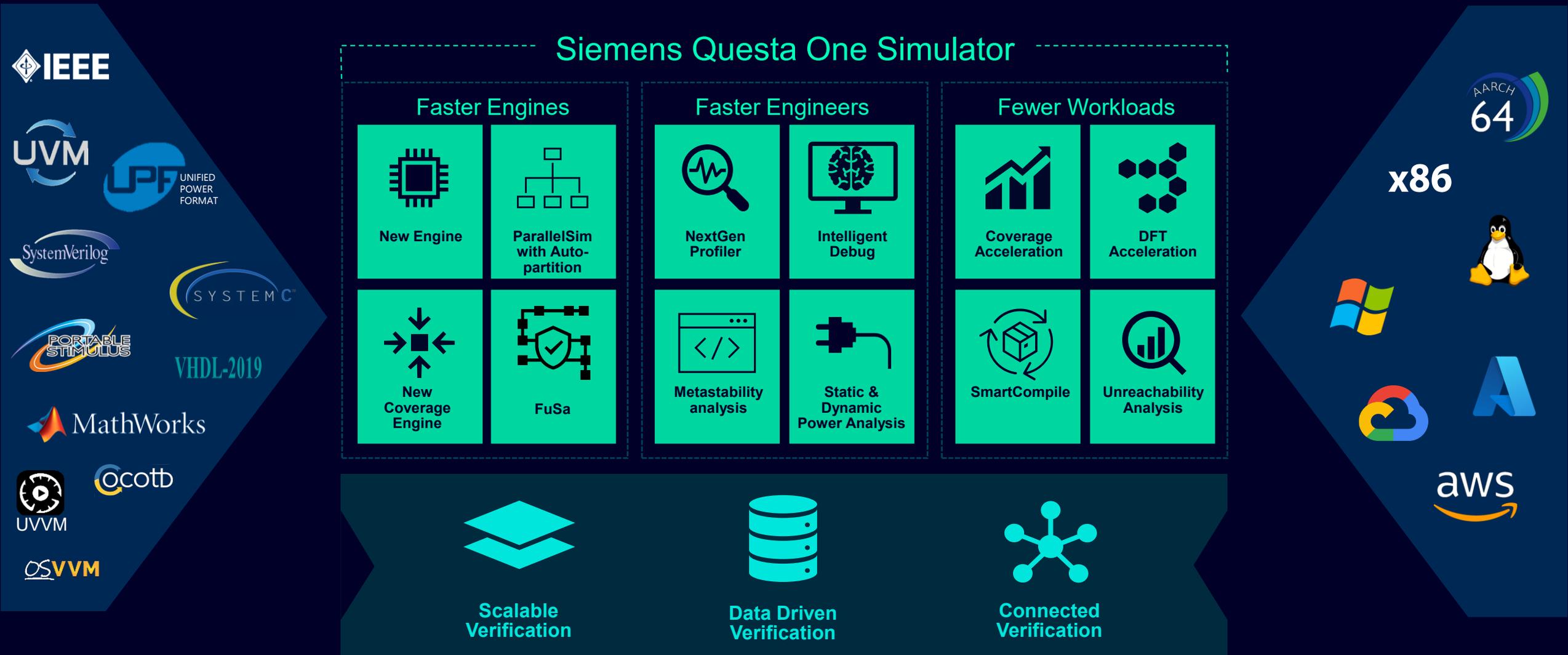
Expanding the power of Avery through solution ecosystem



Questa One Simulator

Questa One Simulator

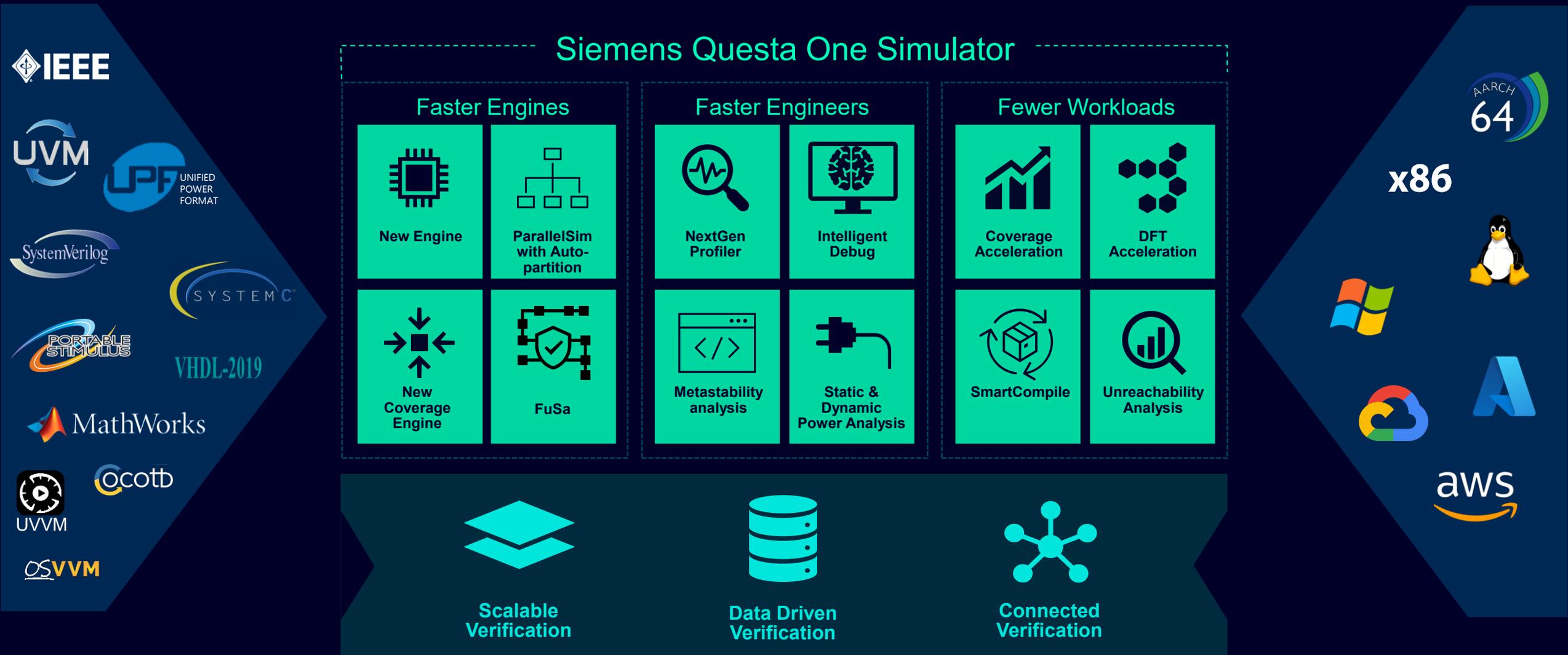
Purpose built to meet the productivity demands of the industry



Deep Dive into Questa One Sim

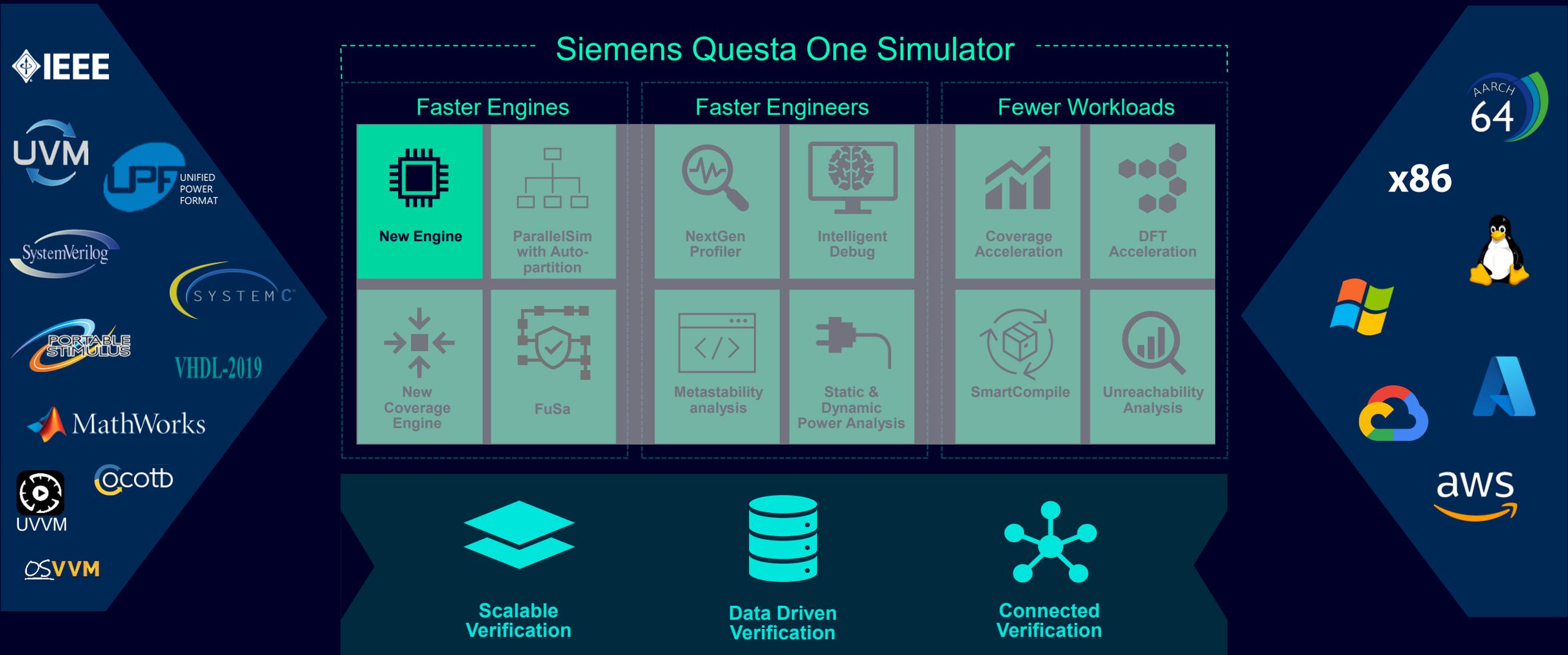
Questa One Simulator

Purpose built to meet the productivity demands of the industry



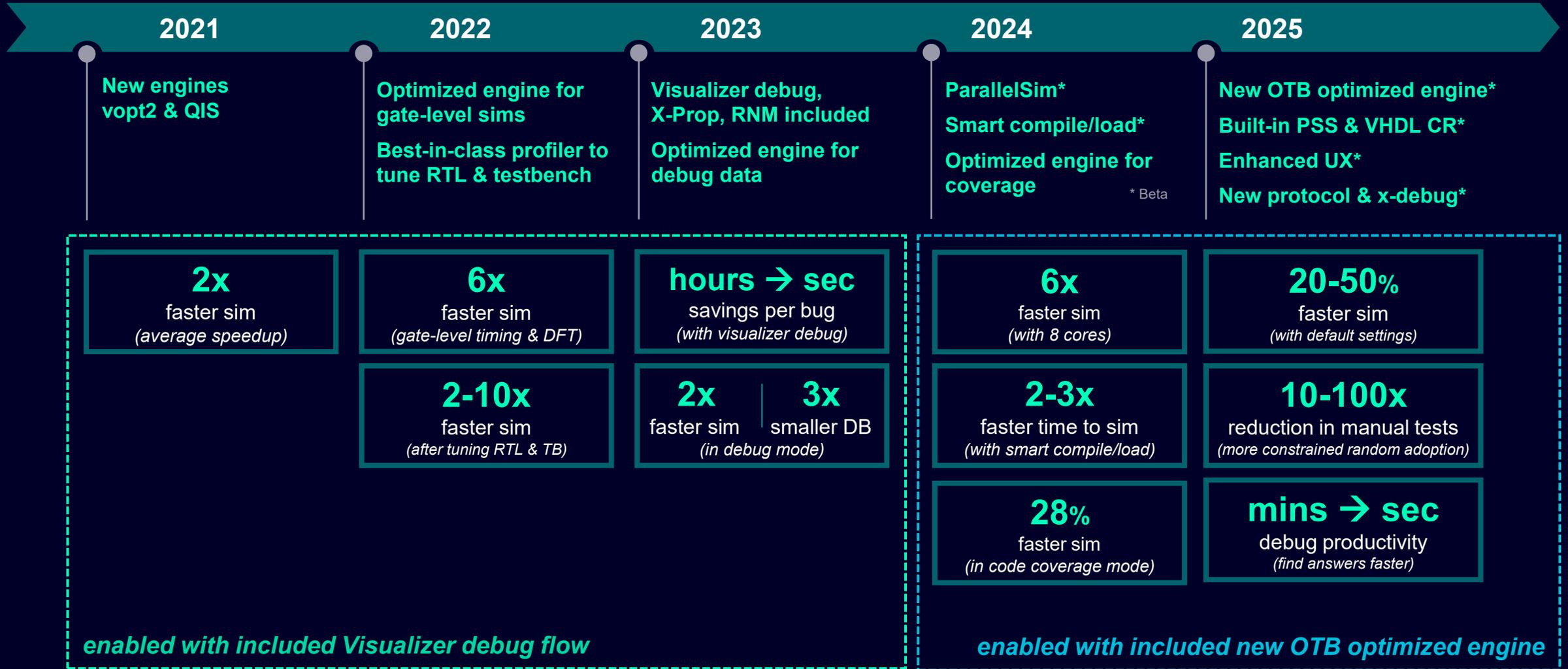
Questa One Simulator

Purpose built to meet the productivity demands of the industry



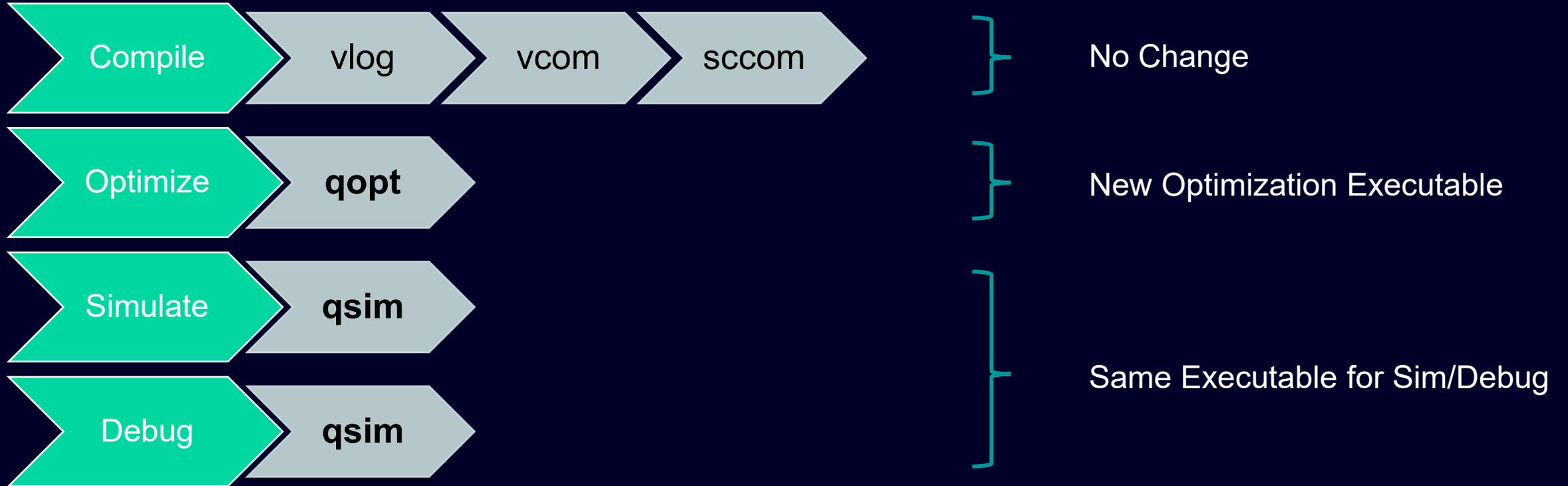
Questa One Sim

Investments that brought us here



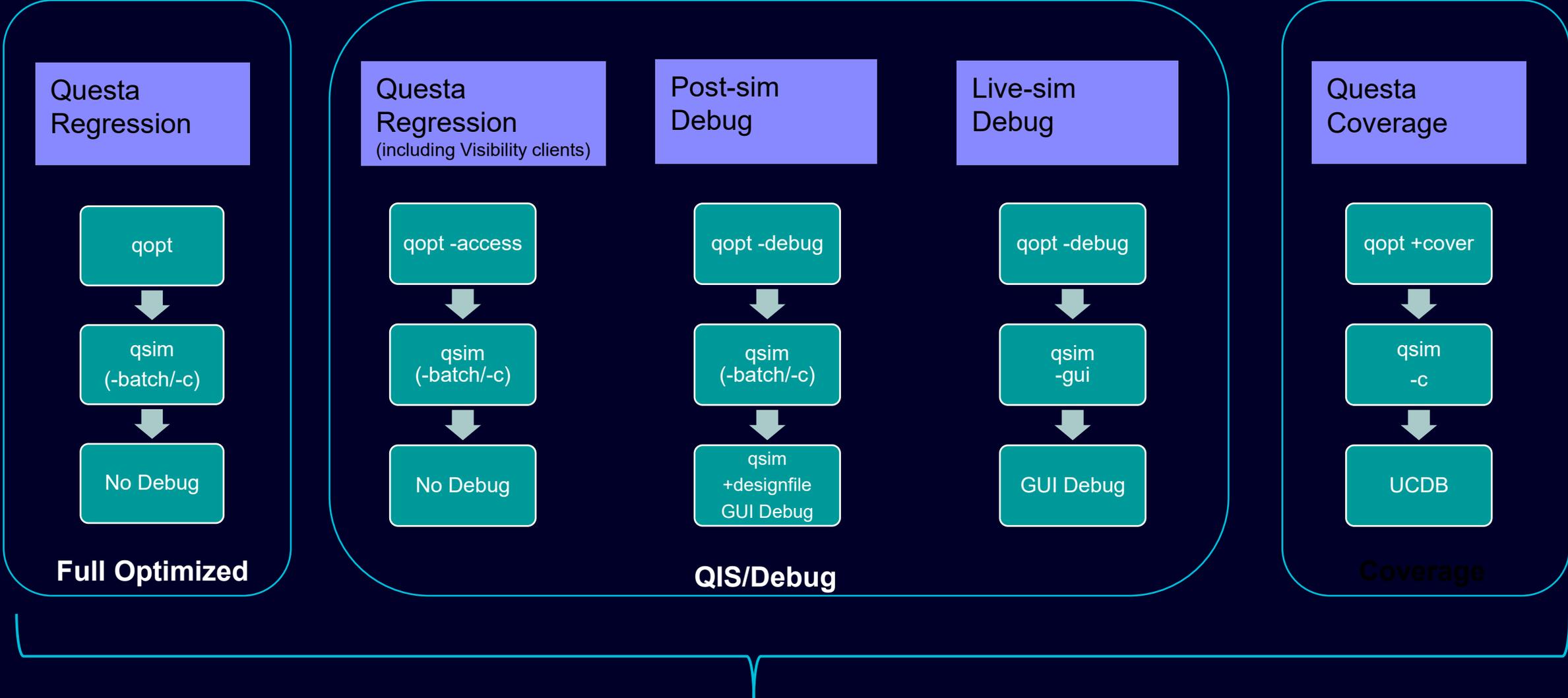
Definition

New Executables for Questa One Sim



Same package as Questa Sim Prime

Why? Simulation Behavior Consistency



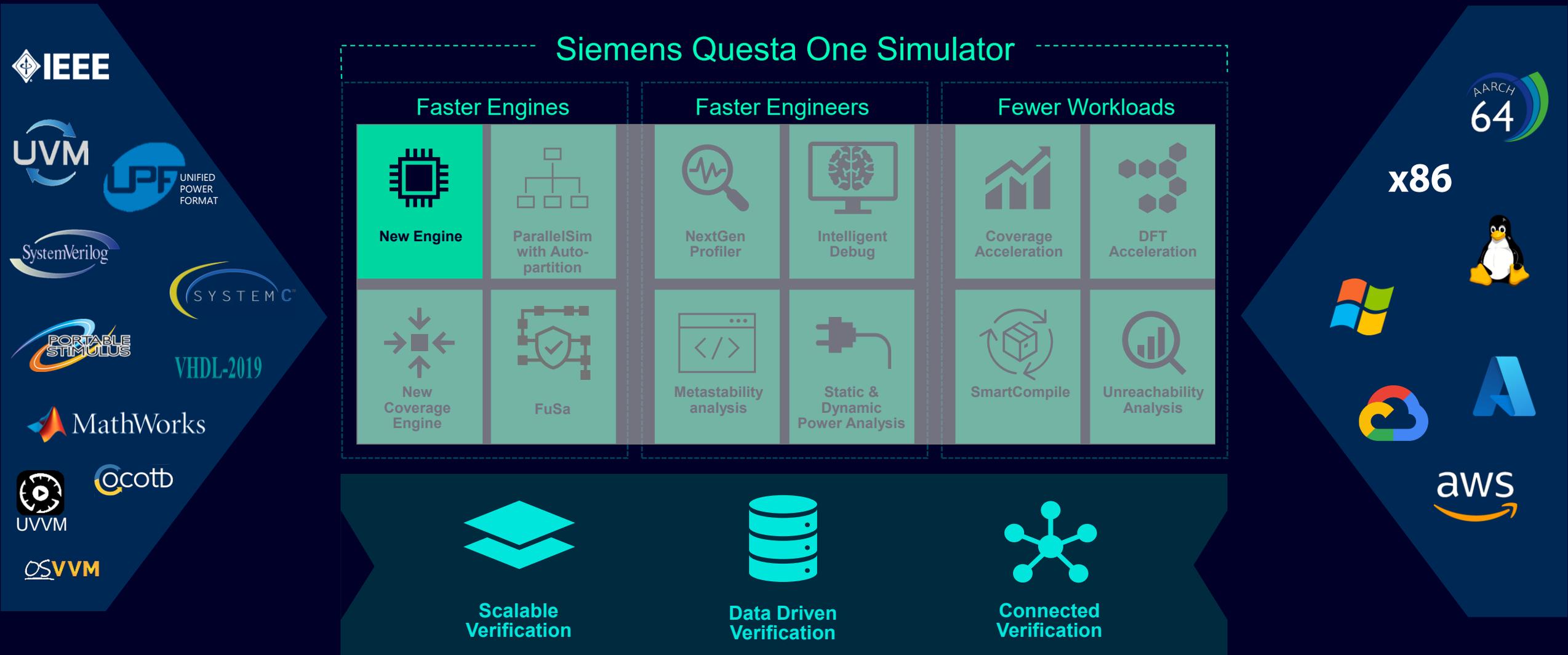
Similar Optimization Scheme

General Use Model Examples

Use Model	Legacy Executables	New Executables	Details
General Purpose run	vlog/vcom/sccom vopt vsim	vlog/vcom/sccom qopt qsim	<ul style="list-style-type: none"> User should change to qopt/qsim to open Questa One Sim features
How to open GUI “Live-sim mode”	vlog vopt +acc vsim -gui -do “add log...”	vlog qopt -debug -designfile... qsim -gui -do “add log ...”	<ul style="list-style-type: none"> +acc is hard error User should add -debug, -designfile -gui will invoke Questa One Sim GUI (Live-sim mode) design.bin file will be located and loaded Qwave dumping is enabled upon “add log” or through -qwavedb
How to open GUI “Post-sim mode”	vsim -view vsim.wlf visualizer -wavefile -designfile	qsim -wavefile -designfile	<ul style="list-style-type: none"> -view is not supported qsim follows Visualizer operation for backward compatibility
qsim Default Mode “without top”	vsim -do “...”	qsim -do “...”	<ul style="list-style-type: none"> Invokes Questa One Sim GUI in (Frame-work (IDE) mode)
qsim Default Mode “with top”	vsim top_opt	qsim top_opt	<ul style="list-style-type: none"> Open Questa One Sim GUI (Live-sim mode)
Visibility Clients	vopt +acc	qopt -access=...	<ul style="list-style-type: none"> Same use-model as QIS
Regressions Mode	vsim -c	qsim -c	<ul style="list-style-type: none"> New -console technology is loaded automatically

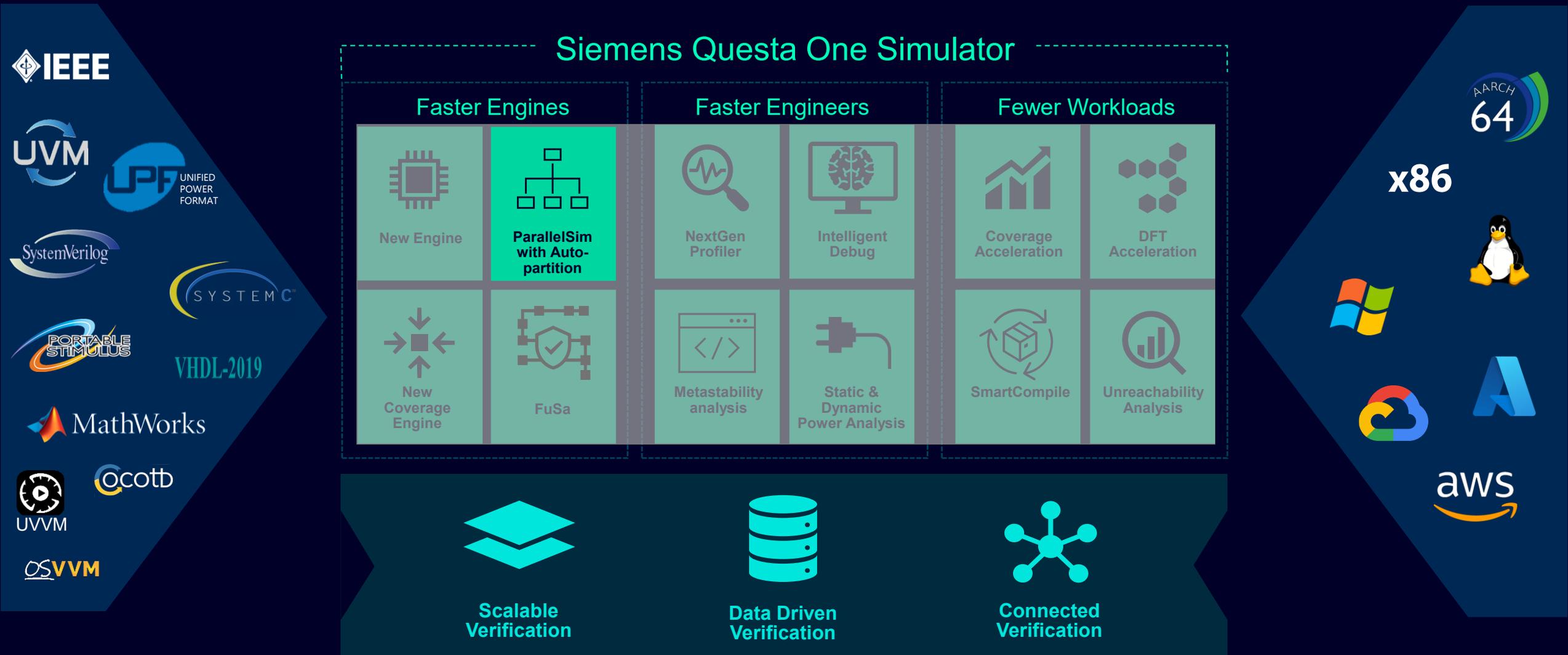
Questa One Simulator

Purpose built to meet the productivity demands of the industry



Questa One Simulator

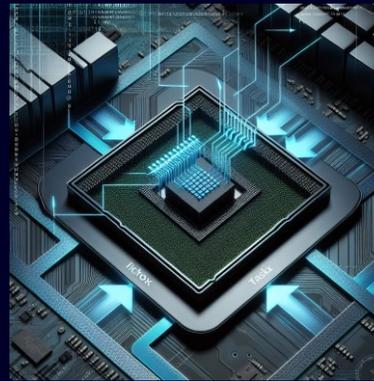
Purpose built to meet the productivity demands of the industry



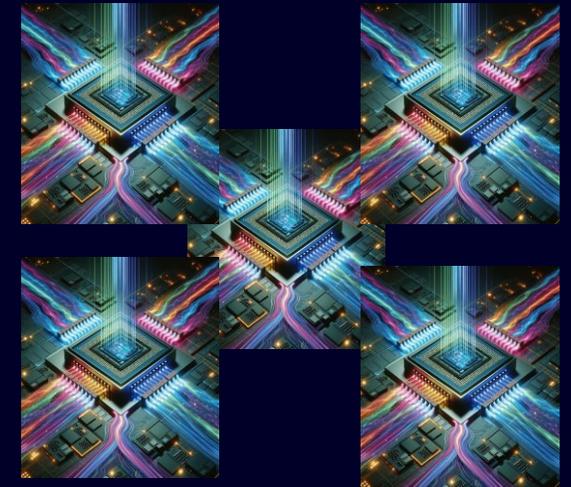
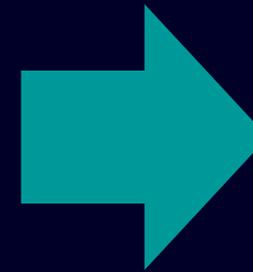
ParallelSim

What is ParallelSim?

- ParallelSim is a multi-core simulation to achieve performance acceleration
- ParallelSim automatically partitions the design at fine-grained level
- Runs each partition in parallel to achieve better performance
- Leverage existing Questa licenses to run on multiple cores
- Preview design fit with Questa Profiler



Questa Single-core simulator



Questa ParallelSim using multiple cores

ParallelSim

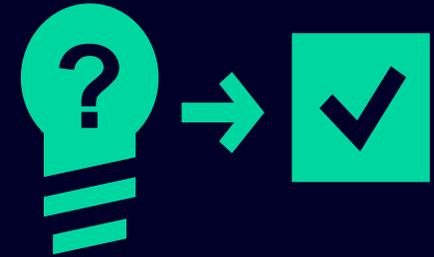
How does the user know?

- That the design is suitable for ParallelSim and can achieve dramatic performance results?
- What is the number of cores that will get the best performance results?



So, it is important to **qualify** the design first and decide if it fits well with ParallelSim or not.

Use **ParallelSim Qualifier**



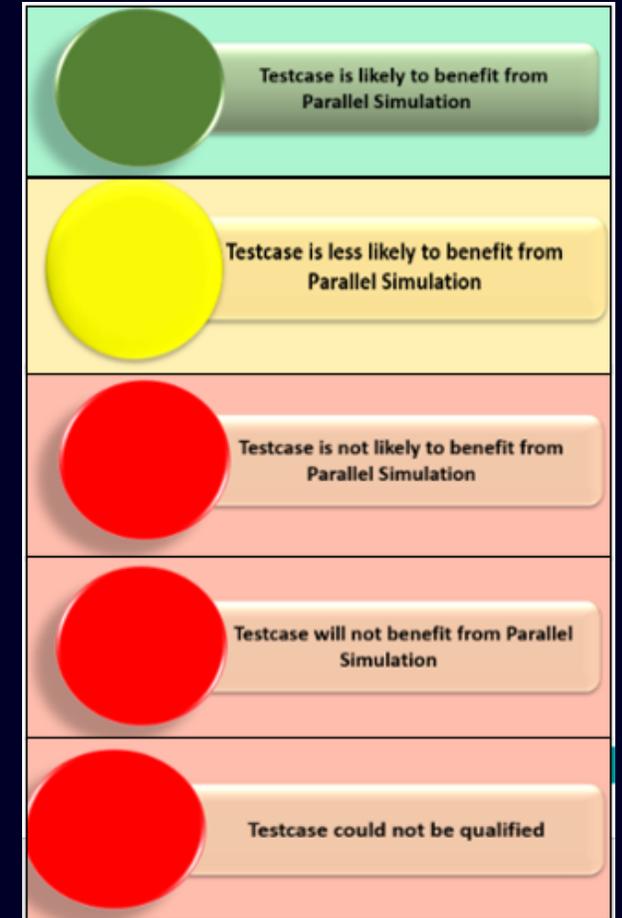
ParallelSim Qualifier

What is it?

- ParallelSim Qualifier is a profiler utility that helps with the following :
 - Determines if a given design will be applicable for parallelism or not.
 - Recommends the best number of partitions to run on our design.
 - `-flps=<the number of partitions> <qualifier recommended options>`



Qualifying the design →



ParallelSim Qualifier conclusions

ParallelSim Qualifier

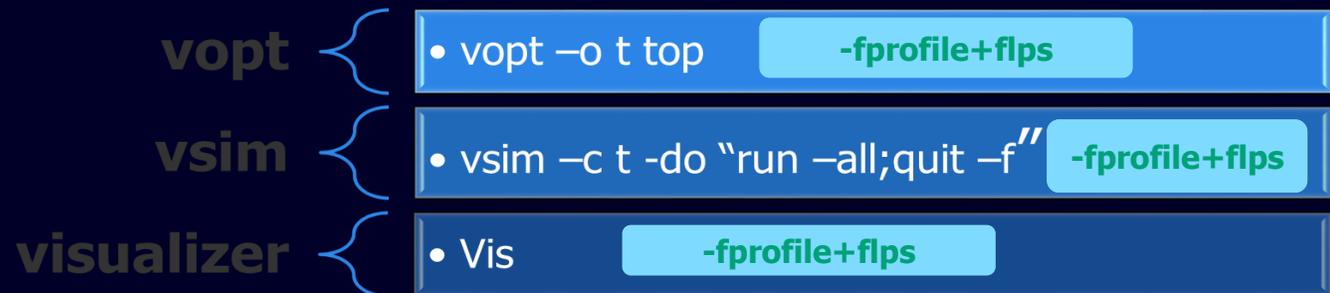
How to Run?

- Add `-fprofile+flps` option to `qopt` & `qsim` command line
- Use `-fprofile+flps` with visualizer to invoke qualifier analysis client
- Same options with all the flows :

Flows

- Qrun flow
`qrun -fprofile+flps`
- 3-step Flow
`qopt top -o opt -fprofile+flps`
`qsim -c opt -do "run -all; quit -f" -fprofile+flps`
- 2-step flow
`qsim -c top -do "run -all; quit -f" -fprofile+flps`
- ELAB flow
`qopt top -o opt -fprofile+flps`
`qsim -c opt -fprofile+flps -elab`
`qsim -c opt -fprofile+flps -load_elab ...`

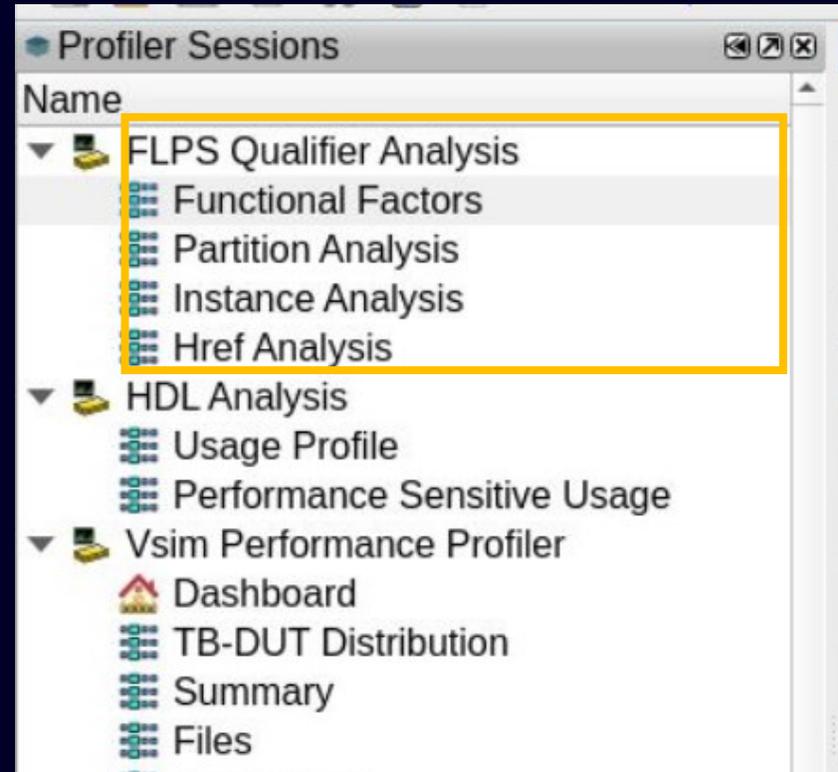
Qualifier also works with `vopt/vsim`



ParallelSim Qualifier

What data does it show?

- **Main QUALIFEIR ANALYSIS Windows:**
 - Functional Factor window
 - Partition analysis window
 - Instance analysis window
 - Href Analysis Window



Functional Factors Window

- The Functional Factors window lists the factors that contributed to the Negative or Moderately positive/negative conclusion. This is also main window to review Qualifier conclusion details.

The screenshot shows the 'Functional Factors' window with a table of factors. A red callout bubble labeled 'Conclusion' points to the top right of the window. A red callout bubble labeled 'Factors Leading to Functional Conclusion' points to the table. A red callout bubble labeled 'Expand "See more" to see conclusion details' points to the 'See More' button.

Name	Details
Moderately Positive Factors	
Design : UVM - HDL Check Path access detected not within the design top hierarchy	yes
Design : UVM - HDL Read access detected not within the design top hierarchy	yes
Design : UVM - HDL Deposit access detected not within the design top hierarchy	yes

Factors Leading to Functional Conclusion

Expand "See more" to see conclusion details

The expanded view shows the conclusion text: 'Testcase is less likely to benefit from Parallel Simulation' followed by 'Testcase is within Field of Use (FoU) with some attributes outside FoU, and with fewer design characteristics that could benefit from Parallel Simulation. You can try Parallel Simulation using "-flps=3".'

Partition Analysis Window

- It gives an insight on the trials that the qualifier did on partitioning the design.
- The best number of partitions (if it exists) appears in green.
- The following information is provided for every partition:
 - Profiler Weight
 - RTL Weight

✓ Testcase is likely to benefit from Parallel Simulation. You can try Parallel Simulation using "-flps=7".

Testcase is within Field of Use (FOU) for Parallel Simulation. You can try Parallel Simulation using "-flps=7".

▲ See Less

Best Partition option

Name	Profiler Weight	Rtl Weight	Details	File and Line
▶ Partitions : 3				
▶ Partitions : 4				
▶ Partitions : 5				
▶ Partitions : 6				
▼ Partitions : 7				
▶ partition master	13.81%	1.61%		
▶ top			Type: VL	tb_top.v:34
▶ partition p1	17.50%	13.72%		
▶ partition p2	18.14%	13.72%		
▼ partition p3	20.71%	15.58%		
▶ top/nvdla_top/u_partition_ma/u_NV_NVDLA_cmac/u_cor...	4.70%	3.43%	Type: VL	NV_NVDLA_CMAC_core.v:2245
▶ top/nvdla_top/u_partition_ma/u_NV_NVDLA_cmac/u_cor...	4.70%	3.43%	Type: VL	NV_NVDLA_CMAC_core.v:2183
▶ top/nvdla_top/u_partition_ma/u_NV_NVDLA_cmac/u_cor...	4.89%	3.43%	Type: VL	NV_NVDLA_CMAC_core.v:2276
▶ top/nvdla_top/u_partition_ma/u_NV_NVDLA_cmac/u_cor...	4.27%	3.43%	Type: VL	NV_NVDLA_CMAC_core.v:2090
▶ top/nvdla_top/u_partition_ma/u_NV_NVDLA_cmac/u_cor...	2.16%	1.86%	Type: VL	NV_NVDLA_CMAC_core.v:1655
▶ partition p4	21.11%	15.58%		
▶ partition p5	4.42%	19.78%		
▶ partition p6	4.32%	20.02%		
▶ Partitions : 8				

Instance Analysis Window

- The Instance Analysis window provides a tree view of the design structure.
- The following information is provided for every instance in the design tree:
 - Design Unit
 - Scope/Local percentage
 - RTL scope/local percentage
 - File and line of instantiation
 - DU File and line
 - Unsupp Summary

Testcase is less likely to benefit from Parallel Simulation

Name	Design Unit	Scope Percentage	Local Percentage	RTL scope	RTL local	Unsupp Summa	File and Line	DU File and Line
tb_top	tb_top	66.6	0.0	100.00%	0.87%		tb_top.sv:18	tb_top.sv:18
cv32e40p_tb_wrapper_i	cv32e40p_tb_wr...	61.8	0.0	99.13%	0.00%		tb_top.sv:154	cv32e40p_tb_wra...
ram_i	mm_ram	18.6	0.0	12.23%	7.48%		cv32e40p_tb_wr...	mm_ram.sv:21
dp_ram_i	dp_ram	3.0	0.9				mm_ram.sv:672	dp_ram.sv:15
random_interrupt_generator_i					1.39%			
data_gnt_stall_i	riscv_gnt_stal...						mm_ram.sv:798	riscv_gnt_stall.sv:...
instr_rvalid_stall_i	riscv_nv...	4.7					mm_ram.sv:690	riscv_rvalid_stall...
instr_gnt_stall_i	riscv...	1.9						
cv32e40p_core			1.0	86.90%	0.13%			
if_stage_i			0.0	10.24%	1.13%			
prefetch...			0.0	2.76%	0.03%			
prefetch_controler...	cv32e40p_prefet...	0.7	0.0	1.10%	1.10%		cv32e40p_prefet...	cv32e40p_prefet...
instruction_obi_i	cv32e40p_obi_in...	1.6	0.0				cv32e40p_prefet...	cv32e40p_obi_int...
gen_no_trans_stable	cv32e40p_obi_in...	1.6	0.0				cv32e40p_obi_in...	cv32e40p_obi_int...
fifo_i	cv32e40p_fifo	1.8	0.0	1.00%	1.00%		cv32e40p_prefet...	cv32e40p_fifo.sv:15
aligner_i	cv32e40p_aligner	2.6	0.0	1.32%	1.32%		cv32e40p_if_sta...	cv32e40p_aligner...
compressed_decoder_i	cv32e40p_comp...	0.9	0.0	5.04%	5.04%		cv32e40p_if_sta...	cv32e40p_compr...
ex_stage_i	cv32e40p_ex_st...	7.2	0.0	21.20%	0.86%		cv32e40p_core....	cv32e40p_ex_sta...
mult_i				2.69%	2.69%			

Can't partition

Dynamic activity (vsim)

Static activity

Partitioning limitations

-dpi

ParallelSim

How to Run?

- Add `-flps=<n>` option to `qvopt` & the other qualifier recommended options.
- Add `-flps` option to `qsim`.
- For debugging, visualizer can be opened as usual.

qopt

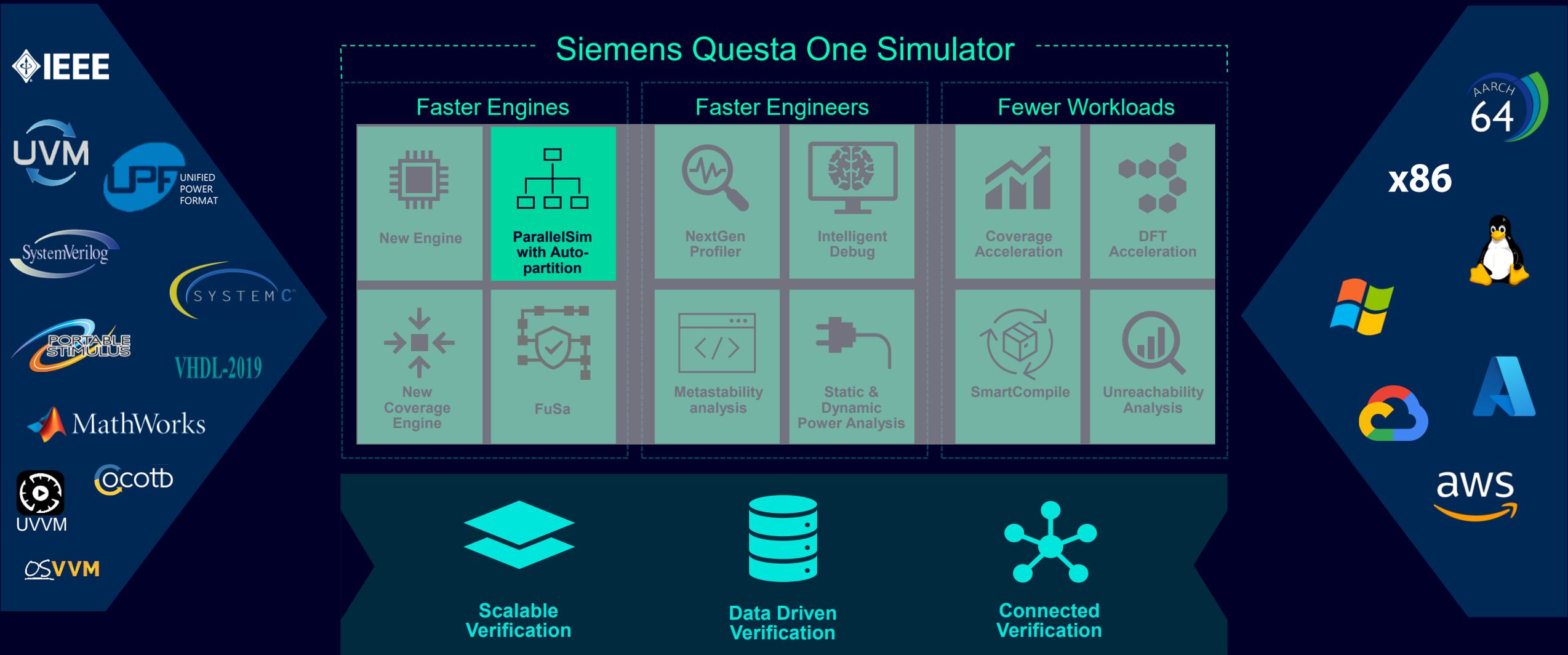
• `qopt -o t top -flps = <n> <recommended options>`

qsim

• `qsim -c t -do -flps "run -all";quit -f"`

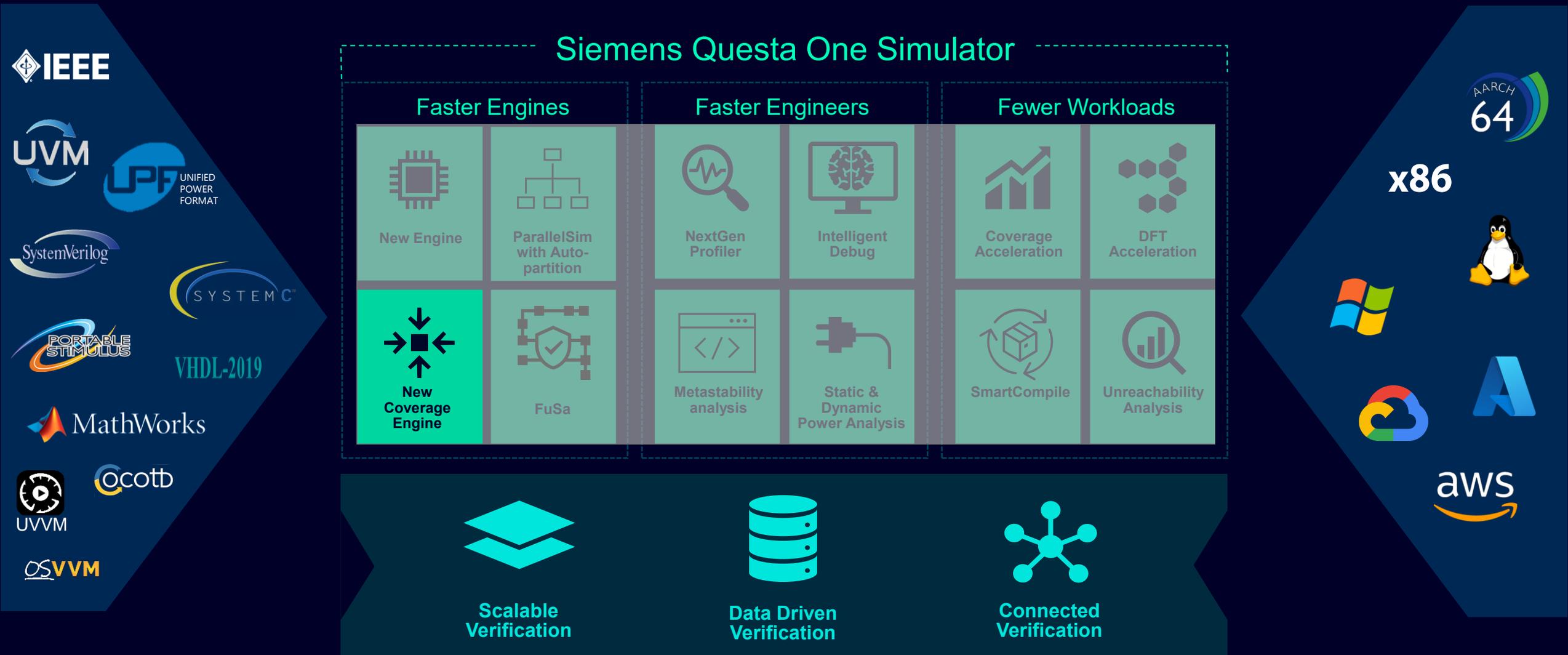
Questa One Simulator

Purpose built to meet the productivity demands of the industry



Questa One Simulator

Purpose built to meet the productivity demands of the industry



What is Ngcoverage

Next Generation Code Coverage Engine

- **Faster Performance**
 - More optimizations
 - Performance oriented defaults
 - Turn on the extra coverage you need
- **Consistency**
 - Always create bins based on user code, not optimizations
 - Better Match fully optimized behavior
 - Less race condition differences
- **Smarter Architecture**
 - Enable new enhancements and tools

28%
faster sim
(in code coverage mode)

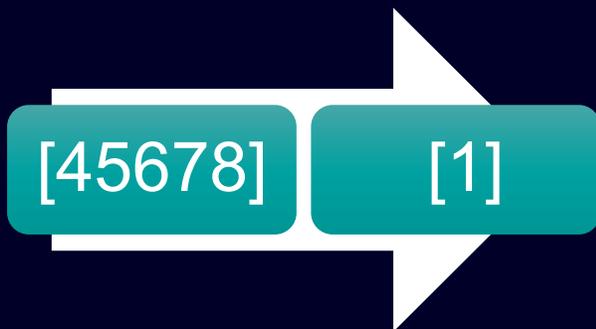


Next Generation Coverage

-ngcoverage efficiency & consistency

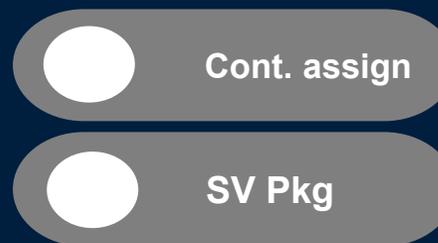
Reduced Data Size

Less storage space.
Faster merge and analysis



Better Defaults

Many coverage constructs are now
off by default



Consistent Bins

Always create bins based on
user code, not optimizations



Block Coverage

Block Coverage

- Block coverage replaces statement coverage
- A block is a group of statements with a single entry and exit point such as:
 - begin-end, if-else, case, wait, while loop, for loop etc.
- Block coverage measures each group of statements executed
- Same exclusion syntax. Exclude any line in the block, exclude the whole block.

```
40
41 always @(posedge clk)
42 begin
43     if(rstn == 0) begin
44         count <= 0;
45         ip_count <= 0;
46         op_count <= 0;
47         foreach(data_fifo[i]) begin
48             data_fifo[i] = 0;
49         end
50     end
51     else begin
52         case({push, pop})
53             2'b01: begin
54                 if(count > 0) begin
55                     op_count <= op_count + 1;
56                     count <= count - 1;
57                 end
58             end
59             2'b10: begin
60                 if(count <= 0) begin
61                     ip_count <= ip_count + 1;
62                 end
63             end
64         endcase
65     end
66 end
67 end
```

Block 1 (lines 44-46)

Block 2 (line 48)

Block 3 (lines 55-56)

Line#	Coverage Type	Hit	Coverage%	Details
44 - 46	Block	✓	100.00%	count <= 0;...op_count <= 0;
48	Block	✓	100.00%	data_fifo[i] = 0;
55 - 56	Block	✓	100.00%	op_count <= op_count + 1;...
61 - 63	Block	✓	100.00%	ip_count <= ip_count + 1;...
67 - 69	Block	✗	0.00%	... <= op_count + 1;...data_fifo

Questa One Sim - New Code Coverage Defaults

- Questa One Sim
 - Code Coverage engine is `ngcoverage`
 - `qopt +cover`
 - `qsim -coverage -do "coverage save onexit name.ucdb"`
- Post-sim Flow changes
 - `vsim -c -viewcov -> qsim -c -viewcov`
 - `vis -viewcov -> qsim -viewcov`
- Questa Core/Prime 2025.2
 - No Change to defaults

Questa One Sim (ngcoverage) Coverage Migration Guide

qopt +cover

qsim -coverage -do "coverage save onexit name.ucdb

- Post-sim Flow changes
 - vsim -c -viewcov -> qsim -c -viewcov
 - vis -viewcov -> qsim -viewcov
- UCDB compatibility
 - No format change.
 - Questa One gui/Visualizer/MIQ/"vsim -c" readers all work.
 - TK GUI does **NOT** support ngcoverage
- Merge – code coverage must match engine
 - “code coverage” from legacy **NOT** merge compatible with ngcoverage
 - “vopt +cover -ngcoverage” ↔ “qopt +cover” **OK**



Questa One Sim (ngcoverage) Coverage Migration Guide

optimization for best performance

- Change **vopt** → **qopt**
- **Remove** +acc switches.
- **Remove** -coveropt
- **Remove** -inlineFactor=0
- **Remove** -covermode
- **Add** -coverconstruct only if you need them
- The best coverage performance is now out of the box
- And just like before: Limit +cover to the hierarchy/coverage types necessary, and -toggleportsonly if you only need toggle ports.

Questa One Sim (ngcoverage) Coverage Migration Guide

Exclusion Commands Migration

Toggle Exclusions

- **No change** is required to the exclusion commands.
- Note: Toggle nodes may have additional alias nodes

```
coverage exclude -du <du_name> -togglenode <node_name>
```

Block Exclusions

- **No change** is required to statement exclusion commands
- Note that when applying an exclusion on a line/statement that's a part of a block, the whole block gets excluded.



Line#	Coverage Type	Hit	Coverage%	Details
281	Block	✓	100.00%	WCtrlDataStart_q <= WCtrlDataStart;
291	Block	✓	100.00%	Nvalid <= 1'b0;
296	Block	✗	0.00%	Nvalid <= 1'b0;
301	Block	✗	0.00%	Nvalid <= 1'b1;
311 - 324	Block	E		...CtrlDataStart_q1 <= 1'b0;...LatchByte <= 2'b00;
330 - 344	Block	✓	100.00%	...CtrlDataStart;...InProgress_q3 <= InProgress_q2;
366 - 367	Block	✓	100.00%	InProgress <= 1'b0;...WriteOp <= 1'b0;
376	Block	✗	0.00%	WriteOp <= WriteDataOp;

```
coverage exclude -line 314 -code s -scope /ethmac_tb/dut/dut/miim1
```

Questa One Sim (ngcoverage) Coverage Migration Guide

Exclusion Commands Migration (Continued)

Branch Exclusions

- **No change** is required to the exclusion commands. (2025.2)

FSM Exclusions

- **No change** is required to the exclusion commands.

Expression/Conditions Exclusions

- **In some cases**, rows order in Expr/Condition tables may differ between next-gen and legacy coverage

Recommendation

Please check the exclusion commands before applying them to make sure that they map to the correct coverage item

Questa One Sim (ngcoverage) Coverage Migration Guide

Testplans

Testplans Changes

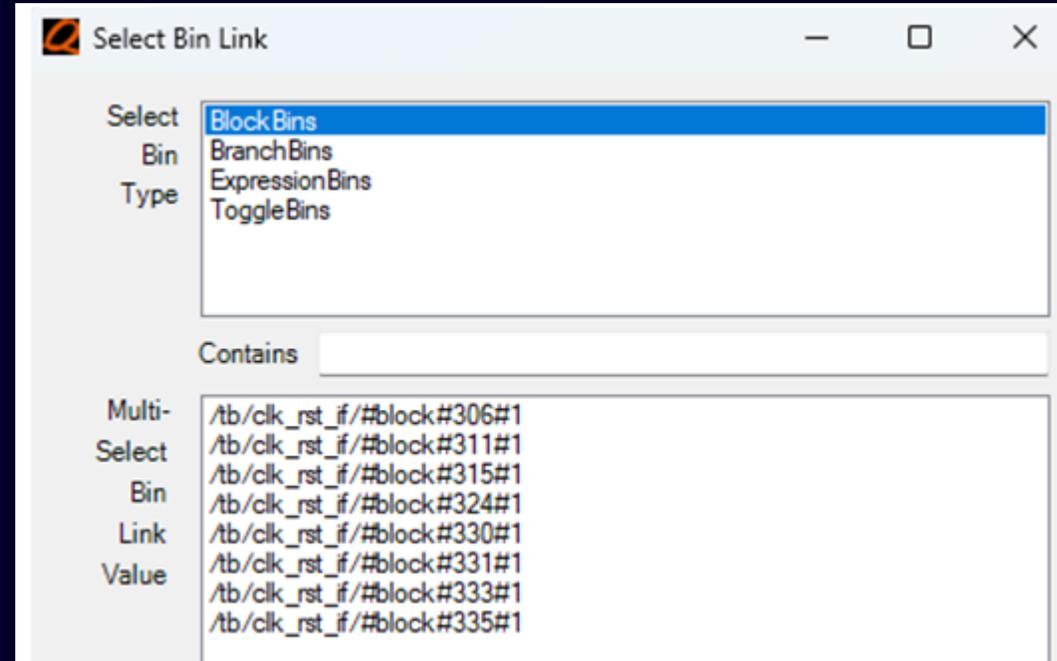
Block Coverage bin links

#stmt# → **#block#**

Select Bin Link in Questa Excel Add-in will show "BlockBins"

Tag links are unchanged

"-code s"



Questa One (ngcoverage) Coverage FAQ

Q?: How will this change my coverage %?

Answer: It can be different. The new percentage number better reflects remaining effort

Comparing coverage % between engines isn't useful information. View this like a different code coverage engine.

Q?: Will I have more bins or less?

Answer: Hard to predict. Blocks are usually less than statement. Toggle are often more. Constructs off by default remove bins. Likely more bins than coveropt3

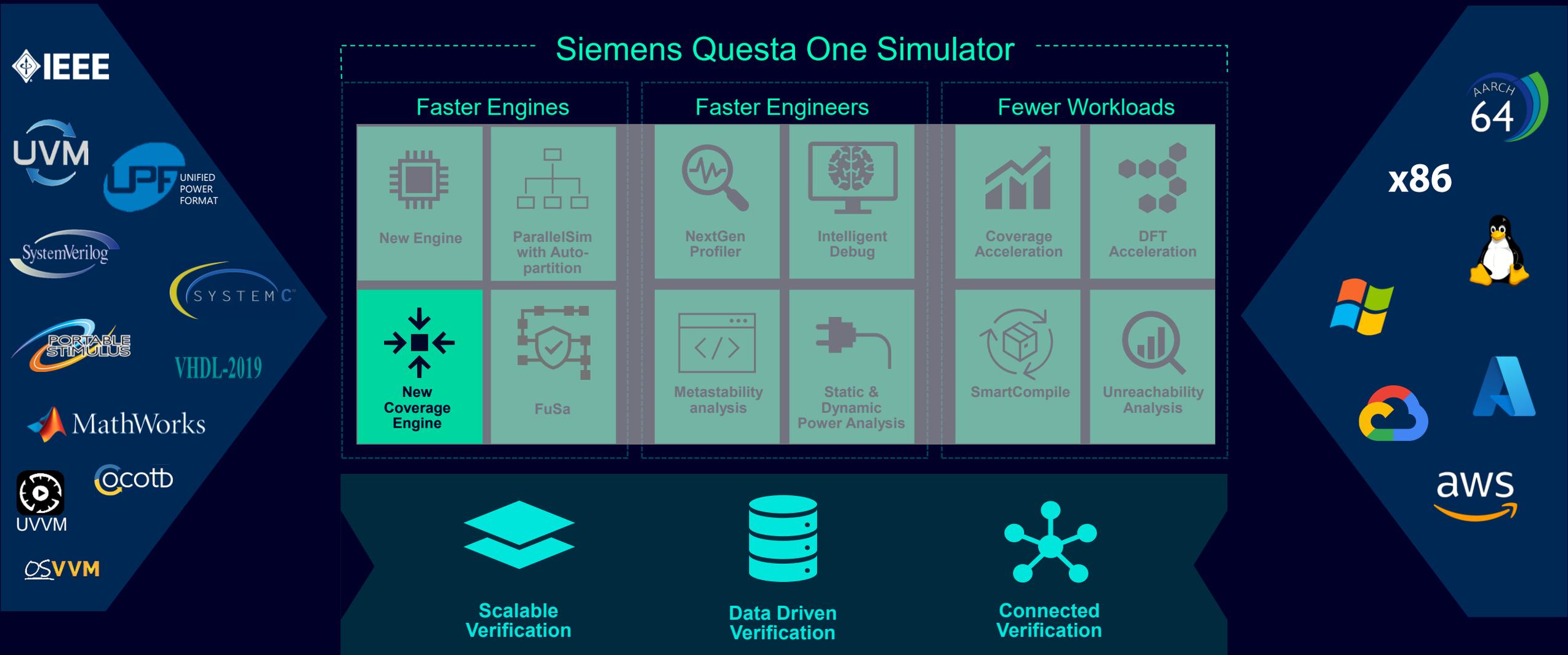
Q?: How much faster will my coverage be?

Answer: 28% on avg

Internal answer: The more optimized full regression, and larger overhead for legacy, the more room for improvement.

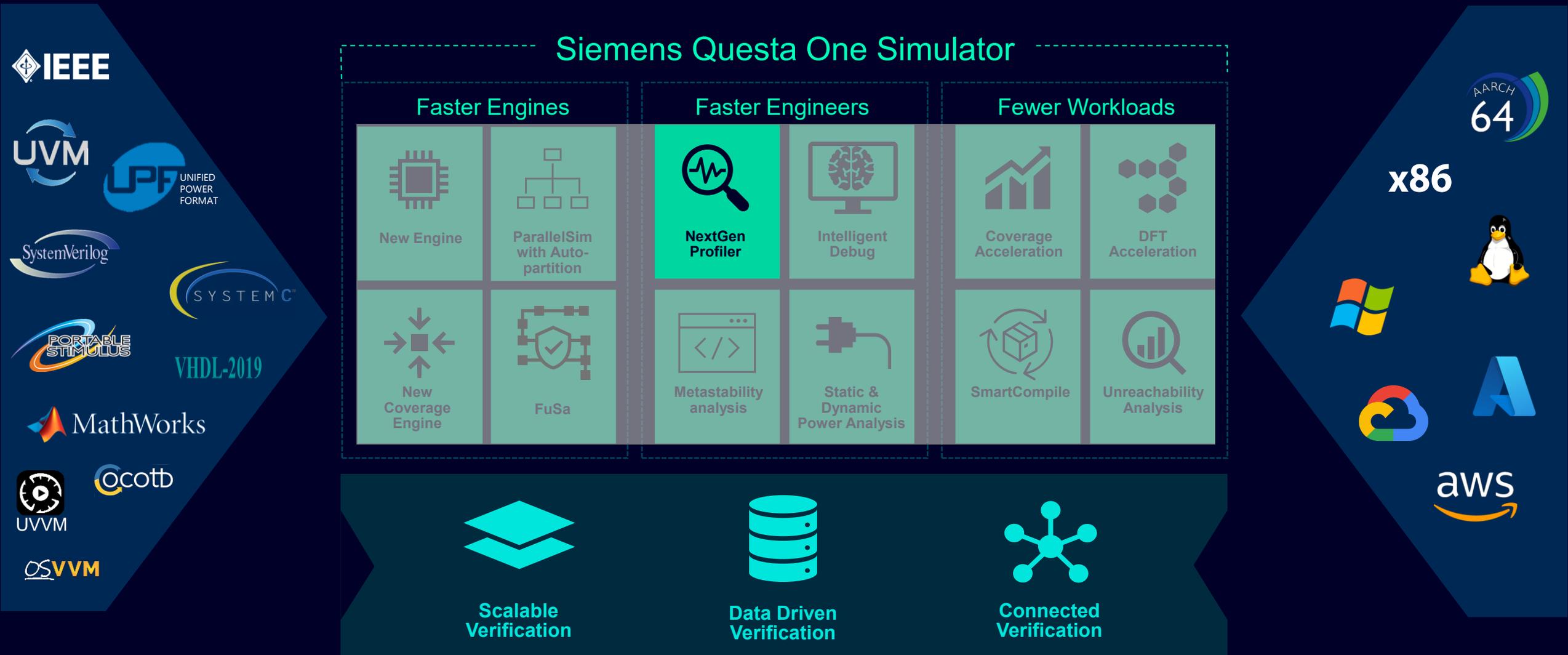
Questa One Simulator

Purpose built to meet the productivity demands of the industry



Questa One Simulator

Purpose built to meet the productivity demands of the industry



Use Model Refresh - Data Generation

- qopt/vopt -fprofile[+options]
- qsim/vsim -fprofile[+options][+engine][+testname=<test_name>]

options:

- +file=<file_name> :Write data to the specified fdb
- +dir=<dir_name> :Create fdb files inside <dir_name>
- +tag=<string> :Tag FDB file names with <tag>

engine:

- +perf : performance profiler
- +solver : solver profiler
- +mem : memory profiler
- Default : Usage profiler

testname: **NEW**

<test_name> identified by the customer, used by multi-test profiler

Use Model Refresh - Data Analysis

- `qsim/visualizer [-fbatch] -fprofile [+engine] [[+options] | [+compare_options]][+report_options]`

options:

+file=<file_name>

+dir=<dir_name>

+tag=<string>

engine

+perf

+solver

+mem

+mass

report options

+report[=<file>] :Report on all available data trees in a formatted ASCII text report

+data=<search_string> :Filter the generated report by <search_string>

+csv :Print report in csv format

compare_options (Currently supported for usage and performance profiler)

+compare+file1=<fdb1>+file2=<fdb2>

-fbatch

UI is not loaded

Multi-test Profiler Use Model

Multi-test Profiler

Use Model

- **Data Generation :**

Same as regular profiler generation

Performance : vsim/qsim -fprofile+perf

Solver : vsim/qsim -fprofile+solver [-solverfprof=<options>]

- **Data Analysis :**

- **qsim** -fprofile+**mass**[+engine][+options]

- engine : +perf , +solver

- options

- default → loads all fdb's in fprofile directory

- **+dir=<>** → provide multiple +dir to load multiple dirs of fdb's

- **+file=<>** → provide multiple +file to load all specified fdb's

- **NEW** **+filelist=<>** → loads all fdb's specified in filelist

- **+rdir=<>** → searches and loads all fdb's in provided directory

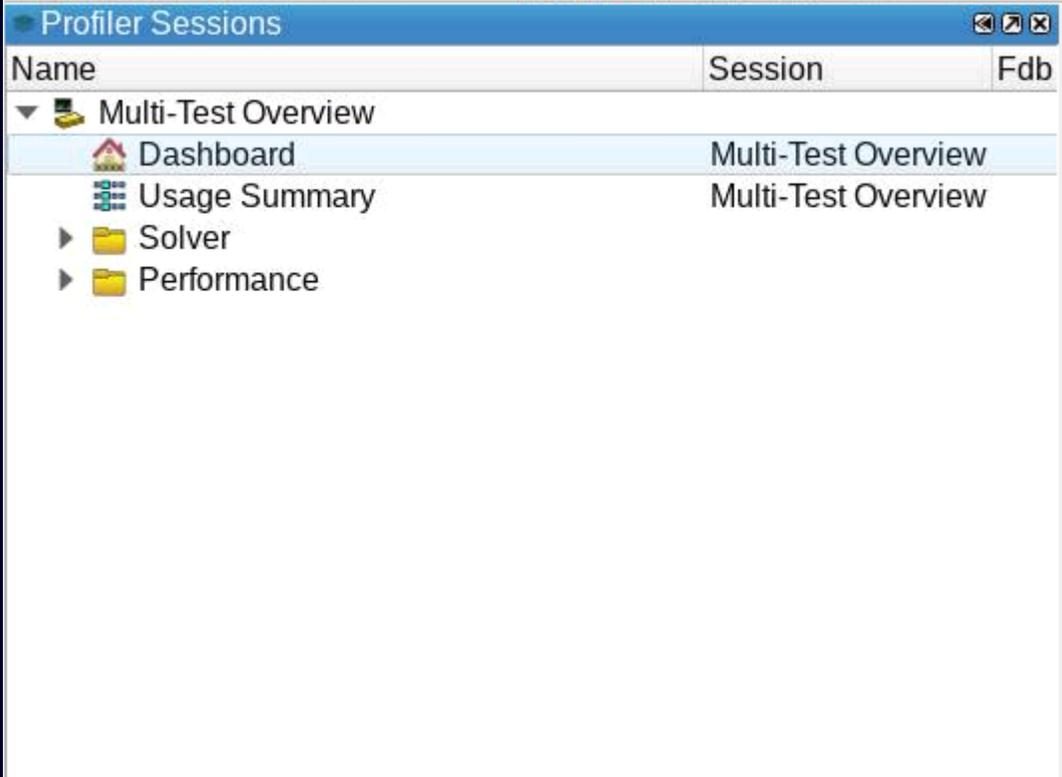
- Note:

- **Fdb's of different version than the 1st fdb will be ignored**

Multi-test Profiler Supported Clients

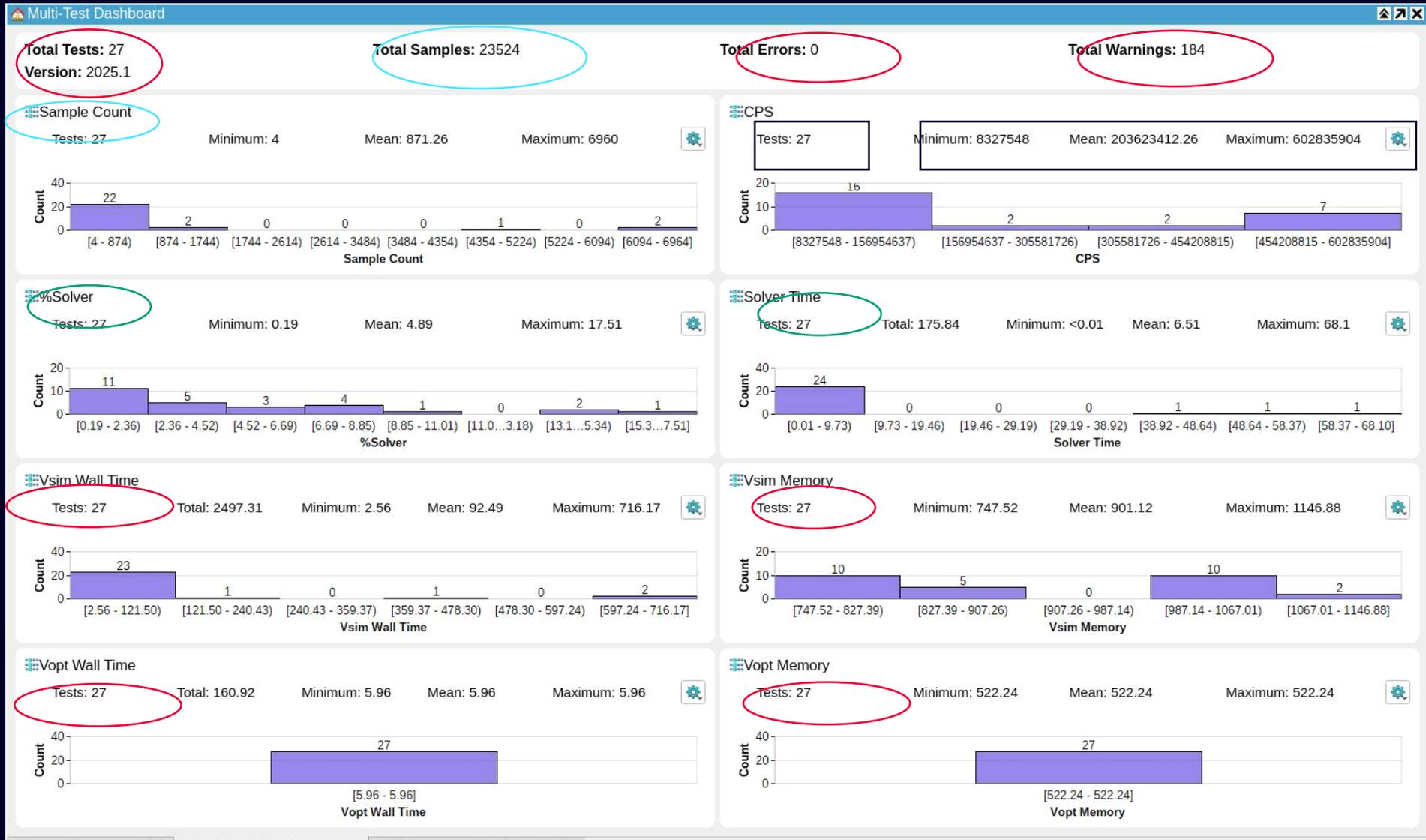
- Usage
- Performance
- Solver
- ParallelSim Qualifier

All clients are merged under a single Session



Name	Session	Fdb
Multi-Test Overview		
Dashboard	Multi-Test Overview	
Usage Summary	Multi-Test Overview	
Solver		
Performance		

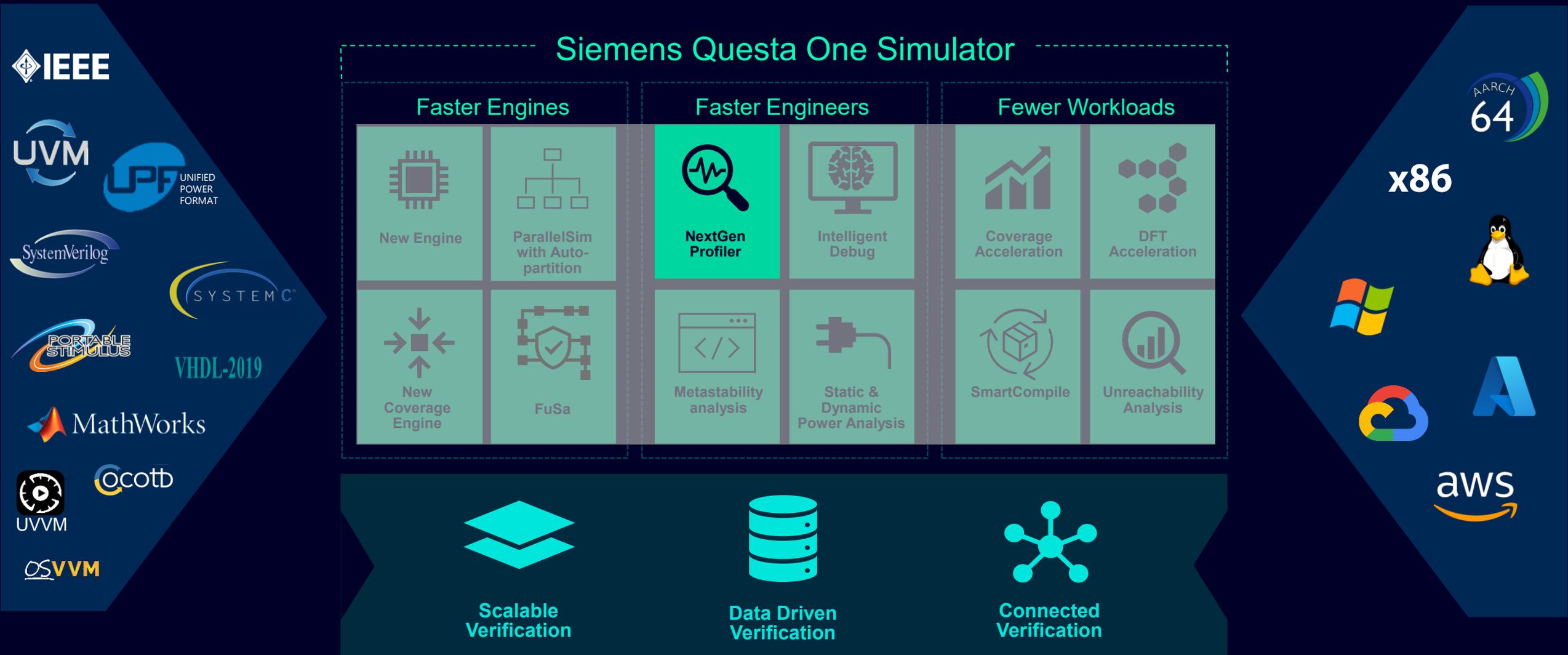
Multi-Test Profiler Windows Dashboard



Usage (default)
 Performance
 Solver

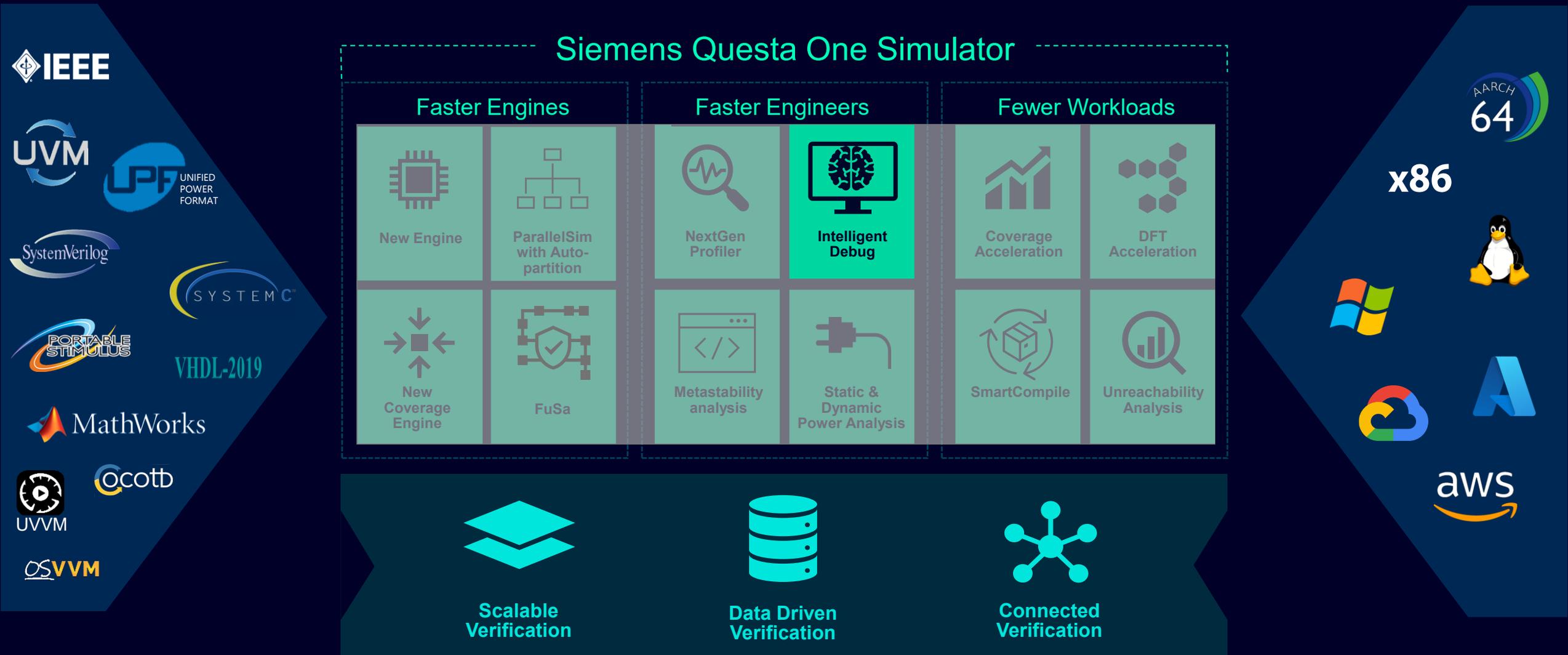
Questa One Simulator

Purpose built to meet the productivity demands of the industry



Questa One Simulator

Purpose built to meet the productivity demands of the industry



The most advanced debugger included with every Questa Sim

“Visualizer inside” Siemens Enterprise Verification Platform

On-demand loading

Compact DB

- waves, design, coverage

Unique features

- Multi-test profiler
- Heat-map x-debug
- Interactive coverage analysis

Cool features

- What-if constraint analysis
- Omni-search
- Time Cone
- X tracing
- “Any transition” tracing

Best in class UVM Debug

- Schematic
- Class instances
- Classes in the wave window
- Transaction debug
- Interactive debug
 - Sequence Window
 - Breakpoints
 - File/Line, Break-on-Change

Seamless platform integration

- Avery VIP protocol-aware
- Veloce HW/SW debug
- Symphony Pro AMS

FSM Debug

Schematics

- Smart flattening
- Smart pruning

Memory Viewer

Real number modeling

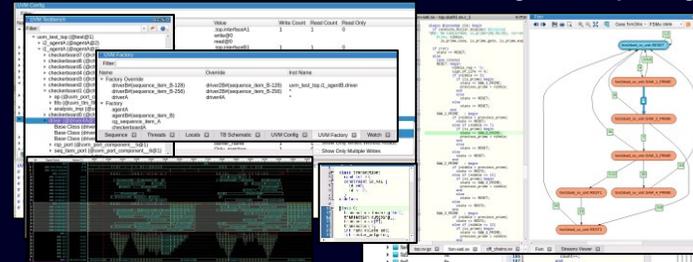
Power Aware Debug

Waves

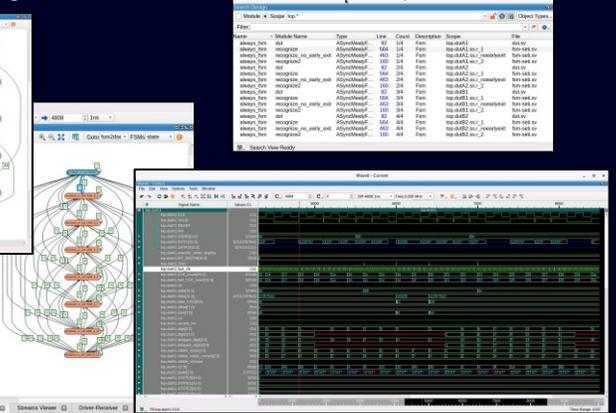
- Custom Radix
- Biometric Search
- Wave Expressions
- Count Events
- Grid Events
- Markers

More ...

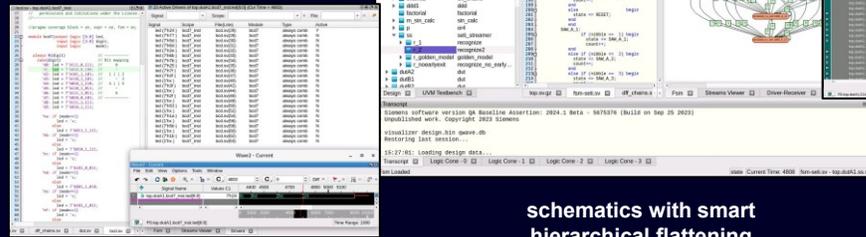
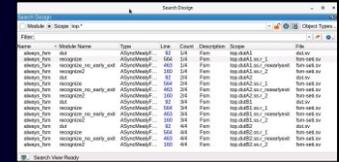
UVM classes and transactions native everywhere



FSM debug & cross probing



design search by type

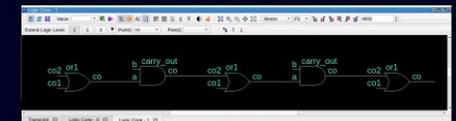
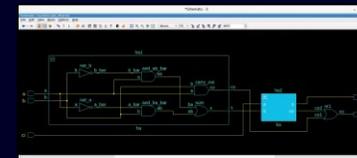


driver/receiver tracing from wave and source code



'x' tracing through sequential logic and time

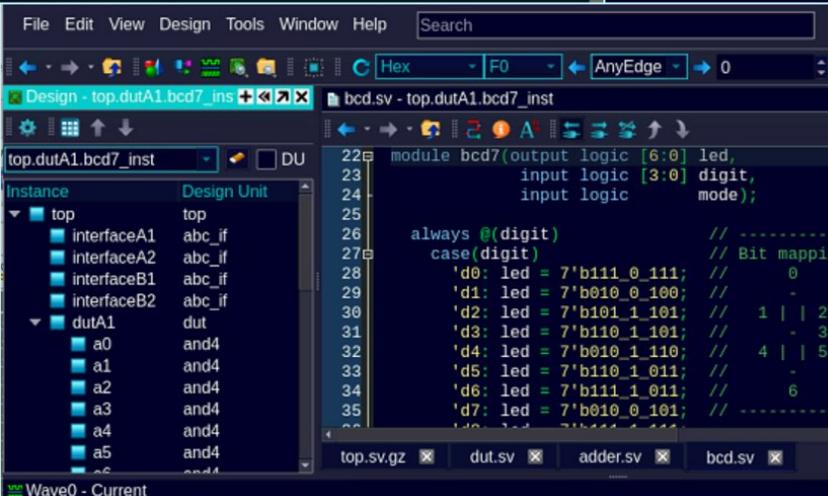
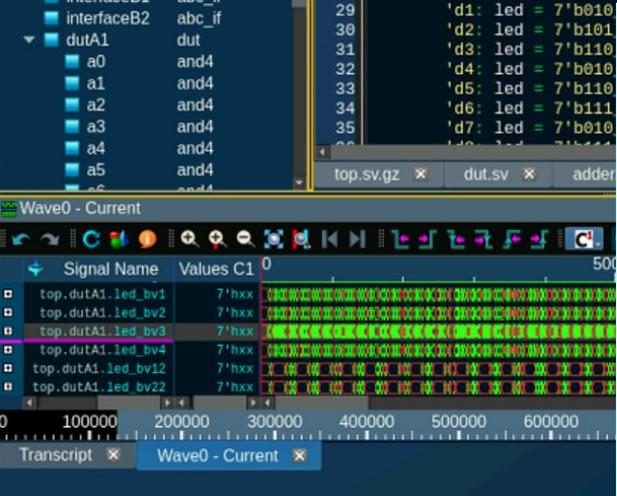
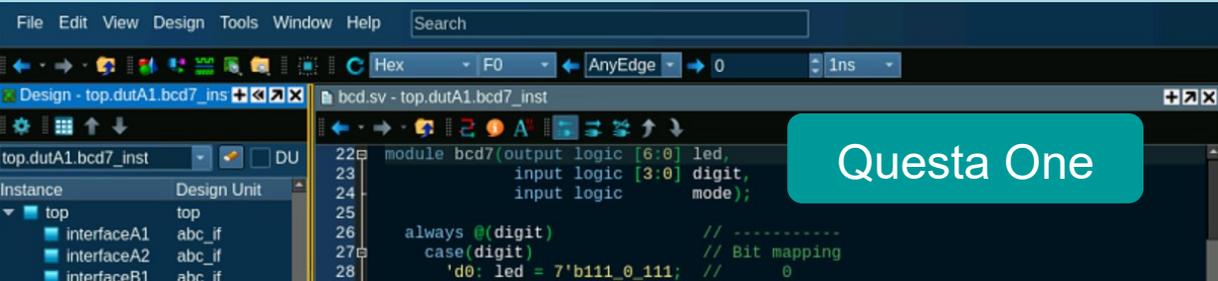
schematics with smart hierarchical flattening



schematic exploration with smart pruning

And even more features with Questa One Sim!

Questa One Theme & Dark Theme



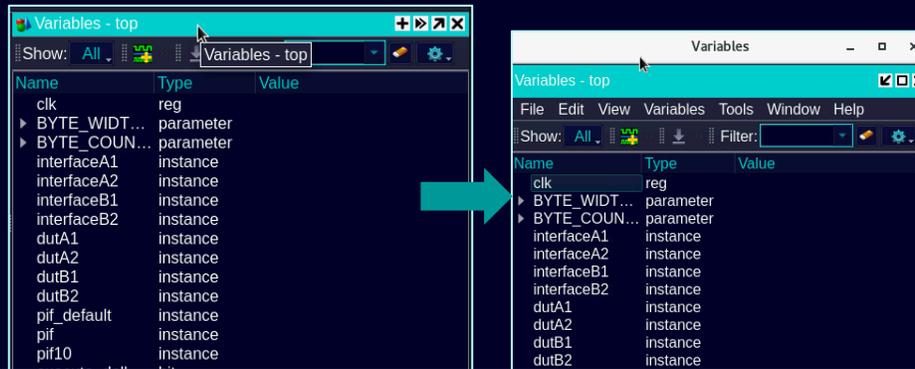
- Ayu Dark
- Ayu Light
- Ayu Mirage
- Dark
- Dracula
- GitHub Dark
- GitHub Light
- Light
- Pitaya Smoothie
- Questa Classic
- Questa One
- Rose Pine
- Rose Pine Dawn
- Rose Pine Moon
- Solarized Dark
- Solarized Light
- Tokyo Night
- Tokyo Night Day
- Visualizer
- Rich

Themes

- Some favorites
- Make your own



Dragging Windows



Infinite drop locations

Drag the Tab “header” → Turns into a “managed window”

Drag the managed window frame... It stays “not dropped in”

Drag the tab header in the managed window to “drop the tab in somewhere”

The screenshot displays the Siemens EDA software interface with several windows open:

- Variables - top.dutA2.add_offset**: A table showing signal values.

Name	Type	Value
s[7:0]	output logic	8'hxx
co	output logic	1'hx
ci	input wire	1'h0
a[7:0]	input wire	8'hd
b[7:0]	input wire	8'hxx
- Schematic**: A logic diagram showing a chain of adders.
- Search Design**: A table showing search results.

Name	Module Name	Type	Line	Count	Descri
#ublk#31584#88	top	NamedBegin	88	1	Scope
BYTE_COUNT[31:0]	top	Parameter	38	1	Variabl
BYTE_WIDTH[31:0]	top	Parameter	27	1	Variabl
- Search Files**: A search results table with columns for Match and Line.
- Waveform**: A timing diagram showing signals for top.dutA1 (CLK, VALID, READY, RW) over time.
- Transcript**: A text window showing simulation output, including "Qsim> |" and "Routing done. Routing option set to 100%."

Buttons, stdio reading, tcl commands, scripting...

VHDL stdin reading

gets stdin

Tcl command support - ~389 with many switches

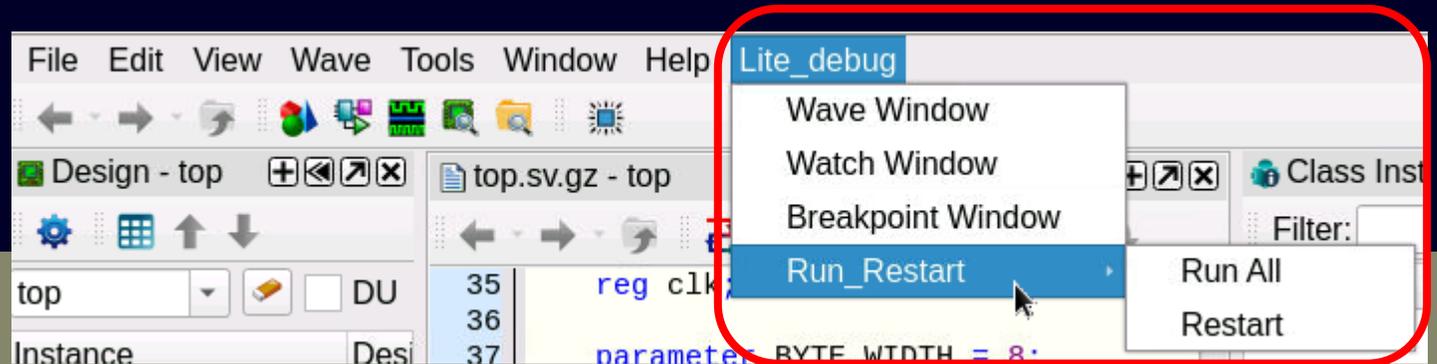
Users can use existing scripts

Buttons / Menus / Toolbars

```
add button Compile {do compile.do} NoDisable {-fg red -bg pink}
add button Optimize {do optimize.do} NoDisable {-fg red -bg pink}
add button Simulate {do simulate.do} NoDisable {-fg red -bg pink}
add button Rerun {do change.do} NoDisable {-fg red -bg pink}
```



Buttons and Menus



```
set myglobalvar 0

proc AddMyBtn {wname} {

    global myglobalvar
    set cmd1 "echo Wave!; view wave $wname"
    set cmd2 "echo Watch!; view watch $wname"
    set cmd3 "echo BP!; view breakpoints $wname"
    set cmd4 "echo Run All; run -all $wname"
    set cmd5 "echo Restart; restart $wname"

    add_menu      $wname Lite_debug
    add_menuitem  $wname Lite_debug      "Wave Window"      $cmd1
    add_menuitem  $wname Lite_debug      "Watch Window"     $cmd2
    add_menuitem  $wname Lite_debug      "Breakpoint Window" $cmd3
    add_submenu   $wname Lite_debug      Run_Restart
    add_menuitem  $wname Lite_debug.Run_Restart "Run All"          $cmd4
    add_menuitem  $wname Lite_debug.Run_Restart "Restart"          $cmd5 -variable myglobalvar \
                                                    -onvalue 1 -offvalue 0 -indicatoron 1
}

AddMyBtn $wname
```

Omnisearch Demo

The screenshot displays the OneSim software interface for a Verilog testbench simulation. The top window shows the Verilog code for `interleaver_tester.v`, which includes logic for counting packets and asserting properties. The bottom window shows a waveform viewer with various signals plotted against time.

```
62 1'b0;
63 up_hs_less10_cnt <= (9(1'---));
64 1'b1;
65 if (up_hs_less10_cnt == 9'b10000000)
66   up_hs_less10_cnt <= (9(1'b0));
67 else
68   up_hs_less10_cnt <= up_hs_less10_cnt + 1;
69 endcase
70 cover property (s_hs_less10(downstream_rdy, downstream_acpt));
71 cover property (s_hs_more10(downstream_rdy, downstream_acpt));
72
73 initial
74 begin
75   clk = 1'b0;
76   reset = 1'b0;
```

The waveform viewer shows signals such as `interleaver_tester.interleaver1.fifo_write_block_r`, `interleaver_tester.interleaver1.assert_pkt_length_check`, `interleaver_tester.interleaver1.assert_pkt_start_check`, and `interleaver_tester.interleaver1.assert_pkt_byepass_check`. The time axis ranges from 10000 to 140000.

X-Debug

Sources of Xs

❖ How are X-values generated in simulation?

Assigned X-values

- Assigned by designers under conditions which should not occur
- Default case statements

Uninitialized State Elements

- Uninitialized state elements/registers are Xs
- Incomplete reset operation
- Power Aware Simulation produces X values

Secondary Sources

- Bus conflicts/multiple drivers
- Out-of-range index
- Incorrect arithmetic (divide by zero)

Siemens Solution

Static Solutions

- Exhaustively identifies all X-state issues
- Enables exhaustive evaluation of circuit start-up and post-reset “X” issues
- Static Products includes Questa Lint, Questa Inspect, Questa Xcheck, Questa CDC/RDC/Simulate FX

Dynamic Solution for RTL

- Ensures simulation closely matches the silicon behavior
- Traps the source of X during simulation
- Ensures signals resolve to a known value if possible
- Dynamic Solution for RTL - QuestaSim

Dynamic Solution for Netlist

- For Gate level Simulation X-pessimism
- Eliminates false Xs automatically
- Eliminates zero-delay race conditions
- Eliminates library modelling error
- Finds and corrects any X-pessimism found on a path
- Dynamic Solution for Netlist-SimXACT

Questa X verification solutions eliminate X-related issues, and more

Eliminating issues and improving design quality from creation through completion

X Solution	Benefits	Limitations
Questa Lint	<ul style="list-style-type: none"> ➤ Run early and often ➤ Minimize X sources as early as possible ➤ Minimize bugs 	No support for sequential related X sources
Questa Inspect	<ul style="list-style-type: none"> ➤ Minimize X sources ➤ Correct initialization issues ➤ Find complex corner case bugs 	No support for corruption due to X-propagation
Questa Check X	<ul style="list-style-type: none"> ➤ Remove random design behavior due to unexpected X-propagation ➤ Minimize flops needing resets 	No support for verifying design behavior
CDC/RDC Simulate FX	<ul style="list-style-type: none"> ➤ Unmask problematic structures early. ➤ Comprehensive coverage of domain crossings and clock-reset tree 	Targets domain crossing paths and clock-reset tree only

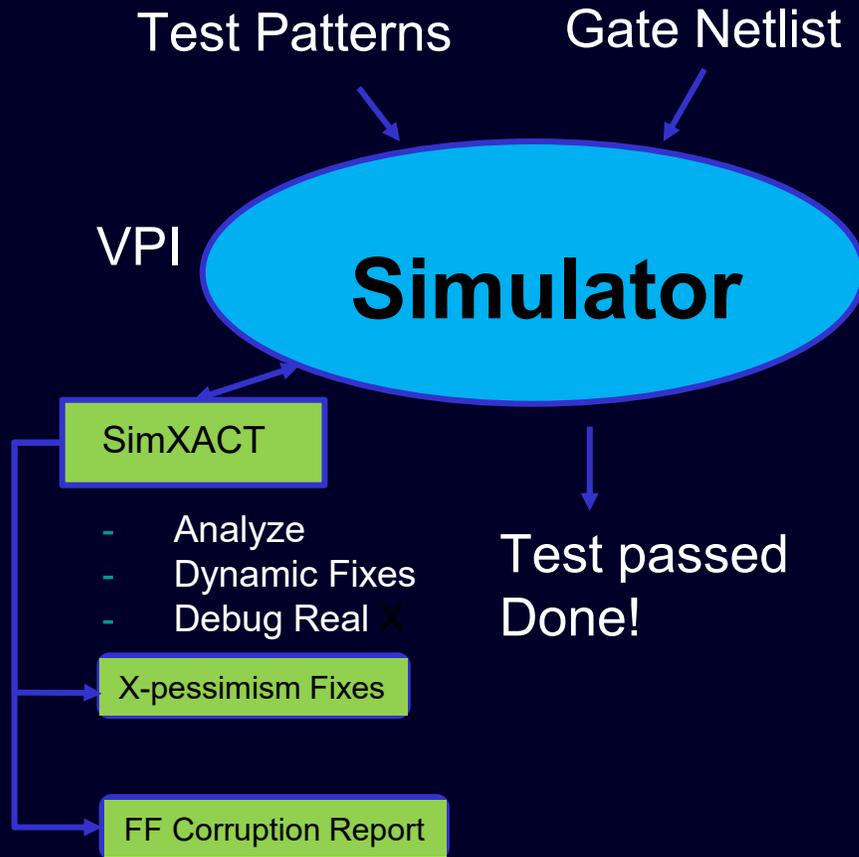
Questa X verification solutions eliminate X-related issues, and more

Eliminating issues and improving design quality from creation through completion

X Solution	Benefits	Limitations
Questa Simulation	<ul style="list-style-type: none">➤ Unmask initialization related bugs	May increase simulation time
XProp	<ul style="list-style-type: none">➤ Better correlation between RTL and gate sims	
SimXACT	<ul style="list-style-type: none">➤ Remove FalseX (X-Pessimism) by analyzing and providing fix for FalseX.➤ Work only on Gate level Netlist	Analysis time during Gate level Simulation is significant but it reduces overall debug time, eliminating all FalseX

Part Of Questa One

SimXACT Flow for Gate Simulation



SimXACT is a 3rd party VPI program to be run with simulator. All analysis is formal and performed under the hood.

- ❑ SimXACT eliminates all false Xs and real Xs still propagate out
 - ❑ If a test passed, the original failure was due to false X and is not a real issue!
 - ❑ If a test failed, because all false Xs are cleaned, debugging Xs becomes much easier
- ❑ SimXACT also writes out X-pessimism fixes for further simulation as well as FF corruption report for X debugging.

SimXACT: Use Model Change – New Flow (contd..)

➤ New Flow

❑ Find false Xs

✓ qopt **-simxact_findx**

✓ qsim **-simxact_findx[=simxact_dut=dut_location][,simxact_delay=start_time_in_ns]
[,simxact_config_file=fileName][,incremental]**

❑ Apply fixes

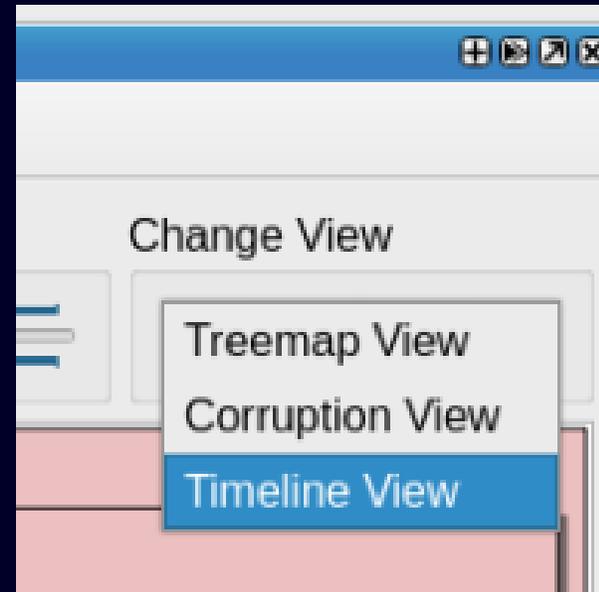
✓ qsim **-simxact_applyfix**

❑ Merging fixes

✓ qsim **-simxact_findx=incremental**

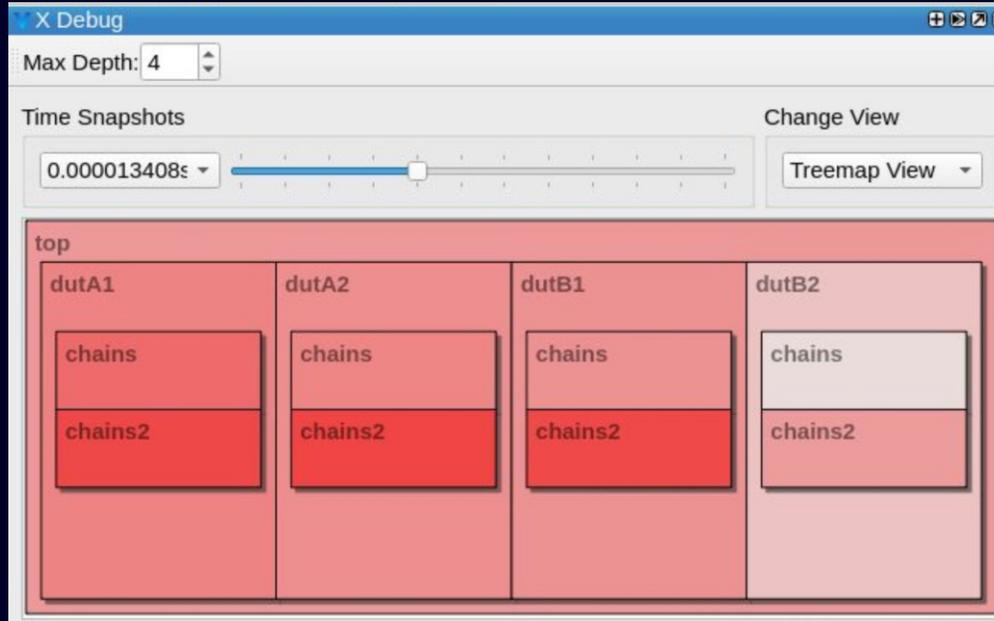
SimXACT: Integration with Visualizer

- It's a visualizer window that mainly displays the distribution of X values within the module and submodules of the design.
- Completely independent feature for debugging Xs. It can be used with or without SimXACT (findx/applyfix).
- XVISDB_DUMP can be used in both RTL and gate level simulation
- X-Debug has 4 main views:
 - Treemap view
 - Corruption view
 - X Timeline view
 - X Drivers view

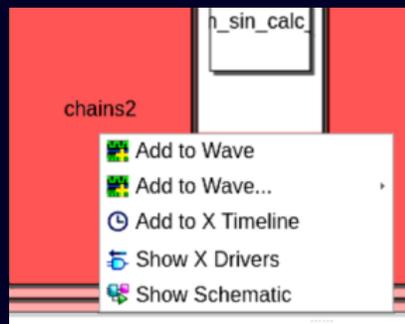
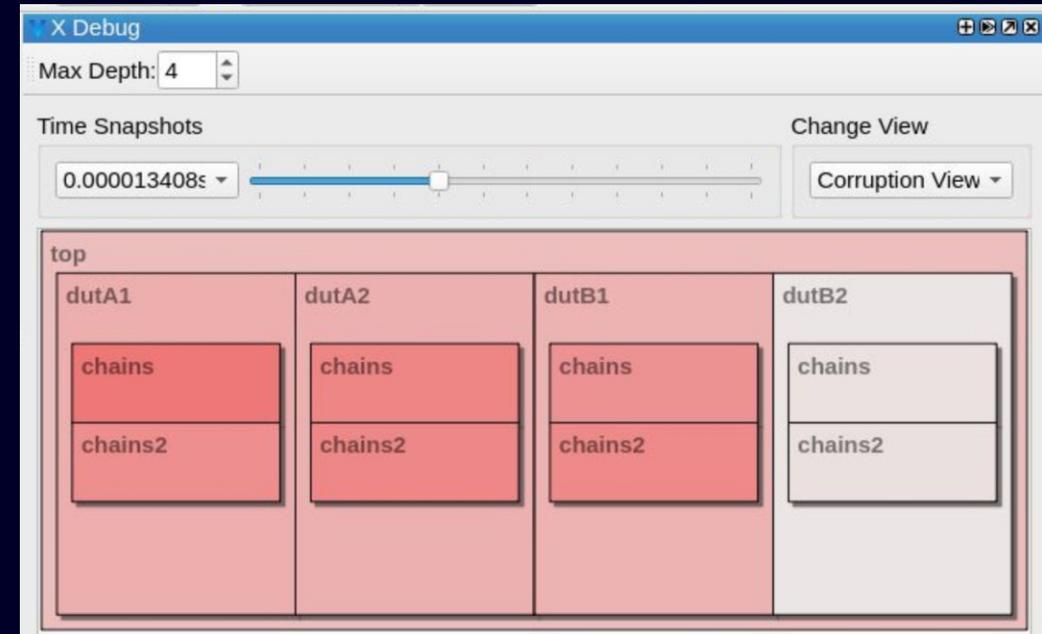


SimXACT: Xdebug – Treemap view & Corruption view

➤ Treemap view

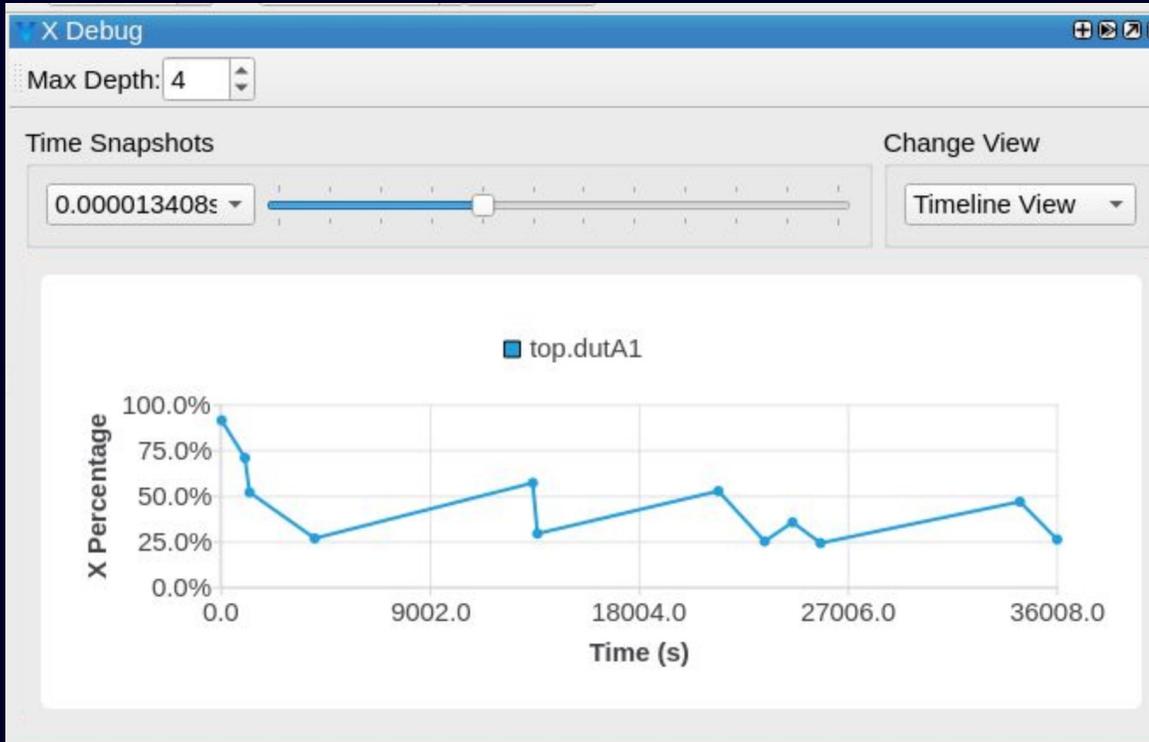


➤ Corruption view



SimXACT: Xdebug – X Timeline view & X Drivers view

➤ X Timeline view



➤ X Drivers view

X Drivers	Time
▼ top.dutA1.sum[0]	
🔍 top.dutA1.add_off...	0.000013408s
▼ top.dutA1.sum[1]	
🔍 top.dutA1.add_off...	0.000013408s
▼ top.dutA1.sum[2]	
🔍 top.dutA1.add_off...	0.000013408s
▼ top.dutA1.sum[3]	
🔍 top.dutA1.add_off...	0.000013408s

SimXACT: X Debug (contd..)

➤ Generating an X Debug database:

- ❑ Add option `qopt -xdebug_dump`

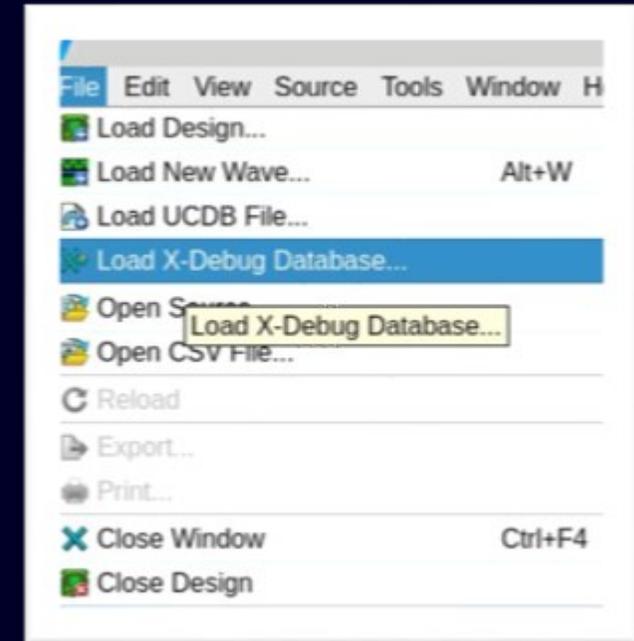
- ❑ Add option `qsim -`

`xdebug_dump=+file=filename+dut=dut_name+interval=interval+x_change=`
`change+type=auto`

`,ff,output,reg,var,net`

- ❑ Loading XDebug database in visualizer:

- ❑ `visualizer design.bin +xdebugdb=xdebug.xdb +q1xdebug`



Constraint Debug

Constraint Debug – Use Model

Available in **Live sim** mode in **Questa One Sim** only

Use Model

- vlog test.sv
- **qopt -debug -designfile design.bin top -o opt**
- **qsim design.bin opt -qwavedb=+signal+class+constraint_debug+...**
- **Add Randomize breakpoint**
- **Run**
- **View -> Constraint Debug**

Randomize Breakpoints

Available on randomize calls:

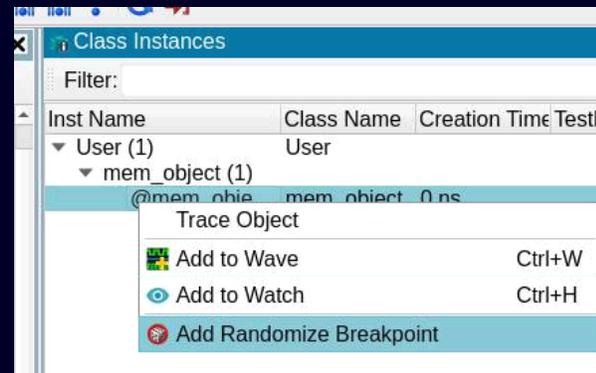
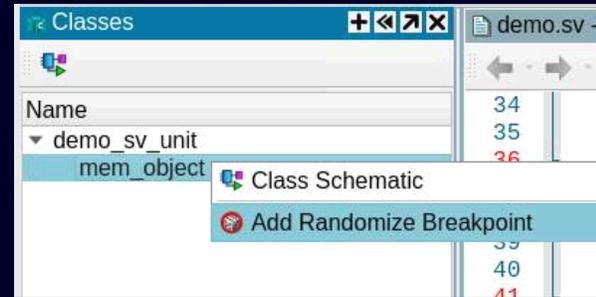
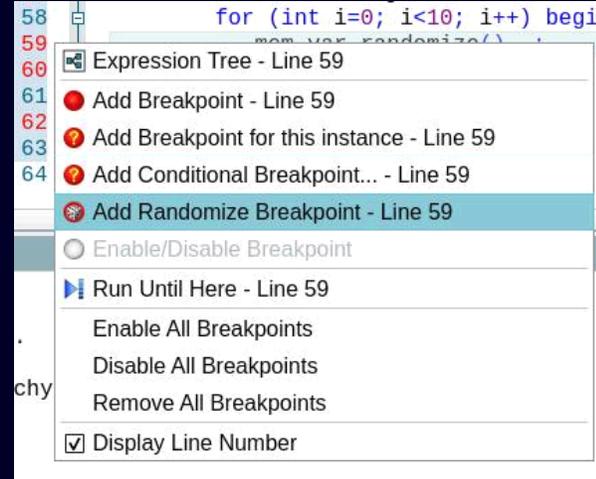
1. From src window
2. From Classes window
3. From Class Instances window

4. Tcl command

bp **-randomize** <src_file> <line_number>

bp **-randomize** -class <class_name>

bp **-randomize** -class_instance <class_instance>



Constraint Debug Window

The screenshot displays the Constraint Debug Window for a class named 'randomize#1'. The left pane shows the Verilog code for a memory object with various constraints. The top-right pane shows a table of random variables, and the bottom-right pane shows a hierarchical list of constraints.

Name	Type	Value	Inject	rand_mode
alloc_address	bit[15:0]	16'd27345	<input type="checkbox"/>	<input checked="" type="checkbox"/>
MemComponent	enum int	ExternalMem	<input type="checkbox"/>	<input checked="" type="checkbox"/>
MemKind	enum int	DRAM	<input type="checkbox"/>	<input checked="" type="checkbox"/>
num_of_bytes	bit[7:0]	8'd251	<input type="checkbox"/>	<input checked="" type="checkbox"/>
region_end_address	bit[15:0]	16'd37125	<input type="checkbox"/>	<input checked="" type="checkbox"/>
region_size	bit[15:0]	16'd12550	<input type="checkbox"/>	<input checked="" type="checkbox"/>
region_start_address	bit[15:0]	16'd24576	<input type="checkbox"/>	<input checked="" type="checkbox"/>
SecurityLevel	enum int	Basic	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Name	Constraint	constraint_mode	Unsatisf
\$implicit	(SecurityLevel inside { [Low:High] });	<input checked="" type="checkbox"/>	0
\$implicit	(MemKind inside { [SRAM:SODIMM] });	<input checked="" type="checkbox"/>	0
\$implicit	(MemComponent inside { [InternalMem:SimulationMem] });	<input checked="" type="checkbox"/>	0
region_c	...	<input checked="" type="checkbox"/>	0
region_c	(region_start_address inside { 4096, 24576, 40960, 53248 });	<input checked="" type="checkbox"/>	0
region_c	(region_end_address == ((region_start_address + region_size) - num_of_bytes));	<input checked="" type="checkbox"/>	0
region_c	(region_size == (num_of_bytes * 50));	<input checked="" type="checkbox"/>	0
addr_c	...	<input checked="" type="checkbox"/>	0
addr_c	(alloc_address >= region_start_address);	<input checked="" type="checkbox"/>	0
addr_c	((alloc_address + num_of_bytes) < region_end_address);	<input checked="" type="checkbox"/>	0
num_of_bytes_c	(num_of_bytes > (37326 - region_end_address));	<input checked="" type="checkbox"/>	0
traits_c	...	<input checked="" type="checkbox"/>	0
traits_c	if ((region_start_address == 4096)) (((MemComponent == InternalMem) (MemComponent == ExternalMem) (MemComponent == SimulationMem)));	<input checked="" type="checkbox"/>	0
traits_c	if ((region_start_address == 24576)) (((MemComponent == InternalMem) (MemComponent == ExternalMem) (MemComponent == SimulationMem)));	<input checked="" type="checkbox"/>	0
traits_c	if ((region_start_address == 40960)) (((MemComponent == InternalMem) (MemComponent == ExternalMem) (MemComponent == SimulationMem)));	<input checked="" type="checkbox"/>	0
traits_c	if ((region_start_address == 53248)) (((MemComponent == InternalMem) (MemComponent == ExternalMem) (MemComponent == SimulationMem)));	<input checked="" type="checkbox"/>	0

Random Variables Pane

Constraints Pane

- Recent Changes
 - Type matches the variables window type
 - Changing Radix
 - Constraints hierarchal representation

Constraint Debug Window

Constraints Pane - Conflicting Constraints

Show: Constraints - 2 unsatisfiable constraints

Name	Constraint
\$override	(region_size == 16'h002d);
▼ region_c	...
region_c	(region_size == (num_of_bytes * 50));
region_c	(region_start_address inside { 4096, 24576, 40960, 53248 });
region_c	(region_end_address == ((region_start_address + region_size) - 1));
\$implicit	(SecurityLevel inside { [Low:High] });
\$implicit	(MemKind inside { [SRAM:SODIMM] });
\$implicit	(MemComponent inside { [InternalMem:SimulationMem] });
▼ addr_c	...
addr_c	(alloc_address >= region_start_address);
addr_c	((alloc_address + num_of_bytes) < region_end_address);
num_of_bytes_c	(num_of_bytes > (37326 - region_end_address));
▼ traits_c	...
traits_c	if ((region_start_address == 4096)) {((MemComponent == InternalMem) && (MemKind == S
traits_c	if ((region_start_address == 24576)) {((MemComponent == ExternalMem) && (MemKind ==
traits_c	if ((region_start_address == 40960)) {((MemComponent == PortableMem) && (MemKind ==
traits_c	if ((region_start_address == 53248)) {((MemComponent == SimulationMem) && (MemKind =

Constraint Debug Window

Distribution Analysis

The screenshot displays a software interface with a code editor on the left and a 'Distribution Analysis' window in the center. The code editor shows a Verilog-like code snippet with line numbers 32 to 59. The 'Distribution Analysis' window has a title bar 'Distribution Analysis' and a subtitle 'Distribution Analysis (class::randomize#11; Samples size: 10)'. It features a dropdown menu with 'c' selected, a '+' button, and a plot area. The plot is a bar chart with a y-axis from 0 to 2 and an x-axis from 2 to 52. A tooltip over the bar at x=32 reads 'Value: 32 Count: 1'. To the right, a table lists constraint modes, random variables, and file locations.

constraint_mode	RandVars Id	File
✓	21	test.sv
✓	3,21	test.sv
✓	20	test.sv
✓	16	test.sv
✓	12	test.sv
✓	8	test.sv

What if Analysis

Constraint Debug (class::randomize#8)

Show: All Random Variables

Name	Type	Value	Inject	rand_mode
alloc_address	bit[15:0]	16'd35557	<input type="checkbox"/>	<input checked="" type="checkbox"/>
MemComponent	enum int	ExternalMem	<input type="checkbox"/>	<input checked="" type="checkbox"/>
MemKind	enum int	DRAM	<input type="checkbox"/>	<input checked="" type="checkbox"/>
num_of_bytes	bit[7:0]	8'd255	<input type="checkbox"/>	<input checked="" type="checkbox"/>
region_end_address	bit[15:0]	16'd37325	<input type="checkbox"/>	<input checked="" type="checkbox"/>
region_size	bit[15:0]	16'd12750	<input type="checkbox"/>	<input checked="" type="checkbox"/>
region_start_address	bit[15:0]	16'd24576	<input type="checkbox"/>	<input checked="" type="checkbox"/>
SecurityLevel	enum int	Basic	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Inject Selected Signal (1/1)

Signal Name:

Value:

OK Reset Skip Close

Show: All Constraints

Name	Constraint
\$implicit	(SecurityLevel inside { [Low:High] });
\$implicit	(MemKind inside { [SRAM:SODIMM] });
\$implicit	(MemComponent inside { [InternalMem:SimulationMem] });
addr_c	...
addr_c	(alloc_address >= region_start_address);
addr_c	((alloc_address + num_of_bytes) < region_end_address);
num_of_bytes_c	(num_of_bytes > (37326 - region_end_address));
region_c	...
region_c	(region_start_address inside { 4096, 24576, 40960, 53248 });
region_c	(region_end_address == ((region_start_address + region_size) - 1));
region_c	(region_size == (num_of_bytes * 50));
traits_c	...
traits_c	if ((region_start_address == 4096)) {((MemComponent == InternalMem) && (MemKind == SRAM) &&
traits_c	if ((region_start_address == 24576)) {((MemComponent == ExternalMem) && (MemKind == DRAM) &&
traits_c	if ((region_start_address == 40960)) {((MemComponent == PortableMem) && (MemKind == MRAM) &&
traits_c	if ((region_start_address == 53248)) {((MemComponent == SimulationMem) && (MemKind == SODIM



Constraint Debug (class::randomize#2)

Show: All Random Variables

Name	Type	Value	Inject	rand_mode
alloc_address	bit[15:0]	16'h60b5	<input type="checkbox"/>	<input checked="" type="checkbox"/>
MemComponent	enum int	ExternalMem	<input type="checkbox"/>	<input checked="" type="checkbox"/>
MemKind	enum int	DRAM	<input type="checkbox"/>	<input checked="" type="checkbox"/>
num_of_bytes	bit[7:0]	8'h6a	<input type="checkbox"/>	<input checked="" type="checkbox"/>
region_end_address	bit[15:0]	16'h91be	<input type="checkbox"/>	<input checked="" type="checkbox"/>
region_size	bit[15:0]	16'h14b4	<input type="checkbox"/>	<input checked="" type="checkbox"/>
region_start_address	bit[15:0]	16'h6000	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
SecurityLevel	enum int	Basic	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Show: All Constraints

Name	Constraint	constraint_mode	Unsatis
\$implicit	(SecurityLevel inside { [Low:High] });	<input checked="" type="checkbox"/>	0
\$implicit	(MemKind inside { [SRAM:SODIMM] });	<input checked="" type="checkbox"/>	0
\$implicit	(MemComponent inside { [InternalMem:SimulationMem] });	<input checked="" type="checkbox"/>	0
\$override	(region_start_address == 16'h6000);	<input checked="" type="checkbox"/>	0
addr_c	...	<input checked="" type="checkbox"/>	0
num_of_bytes_c	(num_of_bytes > (37326 - region_end_address));	<input checked="" type="checkbox"/>	0
region_c	...	<input checked="" type="checkbox"/>	0
traits_c	...	<input checked="" type="checkbox"/>	0

2025 upcoming and on-demand webinars

<https://eda.sw.siemens.com/en-US/eda-events/>

20
Webinars

Unlocking the Power of QuestaSim and Visualizer Integration

In this webinar, you will learn how you can get faster simulation runs, smaller memory footprint, and more visibility into your scripts. We will show you how to optimize your scripts and how to use the new Visualizer tool. Then we will show you how to use the new Visualizer tool. Our Next

Explore How to Protect Against Data Corruption with Formal Security Verification

In this webinar, we will explore essential capabilities such as basic line stepping, dynamic topology verification, and how to identify and fix errors in your workflows. This session will delve into the advanced features of Avery's PCIe Verification IP, including dynamic error injection, error generation, error injection, and error detection.

An End-to-End Functional Safety Solution for Automotive ICs Based on ISO 26262

In this webinar, you will learn more about Siemens EDA functional safety concepts and how they can be applied to your designs. We will cover concepts such as loop solutions, injection analysis, and how to integrate safety into your design process. In this webinar, you will learn more about Siemens EDA functional safety concepts and how they can be applied to your designs. We will cover concepts such as loop solutions, injection analysis, and how to integrate safety into your design process.

Streamlining FPU Verification with an Alternative to C-reference Model Approaches

In this webinar, we explore the powerful smart regression features of collaborative browser-based data-driven verification. You will then learn how to harness the full potential of Questa Verification IQ to boost efficiency and productivity in your verification efforts, take advantage of

Smart Regression: Optimize Regression Efficiency Using Questa Verification IQ Regression Navigator

In this webinar, we explore the powerful smart regression features of collaborative browser-based data-driven verification. You will then learn how to harness the full potential of Questa Verification IQ to boost efficiency and productivity in your verification efforts, take advantage of

PCIe Gen7 Verification with Siemens Avery Verification IP

This session will delve into the advanced features of Avery's PCIe Verification IP, including dynamic error injection, error generation, error injection, and error detection.

Faster Debug of Complex Testbenches using Visualizer

In this webinar, we will explore essential capabilities such as basic line stepping, dynamic topology verification, and how to identify and fix errors in your workflows.

Smart Debug: Accelerate Root Cause Analysis and Reduce Debug Turnaround Time with Questa Verification IQ Regression Navigator

This session will introduce the power of debugging code and functional coverage analysis. We will show you how to use the new Visualizer tool. Then we will show you how to use the new Visualizer tool. Our Next

Faster Debug Using QuestaSim Interactive Coverage Analysis

This session we will explore the power of debugging code and functional coverage analysis. We will show you how to use the new Visualizer tool. Then we will show you how to use the new Visualizer tool. Our Next

Securing your FPGA Design from RTL through to the Bitstream

This session will briefly introduce practical tools such as the Siemens Analyze Architecture and VerifySecure technologies, highlighting how they support the overall security strategy. In addition, we will introduce

Breaking Barriers: Ethernet 1.6T, Infiniband, UALink, and UEC Verification for Next-Gen Connectivity

This session will introduce the power of debugging code and functional coverage analysis. We will show you how to use the new Visualizer tool. Then we will show you how to use the new Visualizer tool. Our Next

Improving FPGA Safety and Security

Standard, Compliance: FPGA Equivalence Checking to the Bitstream

Compliance: FPGA Equivalence Checking to the Bitstream

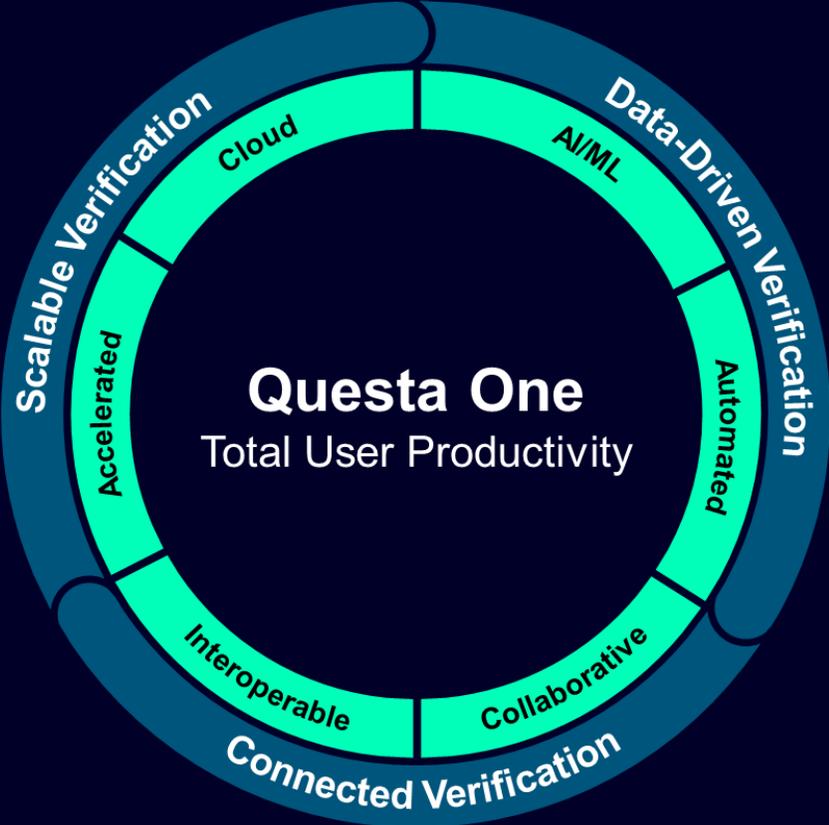
Security policies across various domains such as aerospace, embedded security, and automotive safety have been re-analyzed. We will show you how to use the new Visualizer tool. Then we will show you how to use the new Visualizer tool. Our Next

Solving the Semiconductor Verification Crisis: From Problem to Productivity

📅 Wednesday, May 21, 2025 ⌚ 8:00 AM Pacific Daylight Time ⌚ 1 hour

Questa One – Next Generation of Smart Verification Solution enabled by AI

Improving user productivity through the collective power of technology



Faster
Engines



Faster
Engineers



Fewer
Workloads



Delivering up to **5x** improved
Total User Productivity



Questions?

Contact

Published by Siemens EDA

Faiçal Chtourou

Field Application Engineer

E-mail faycal.chtourou@siemens.com

TEŞEKKÜRLER

CDT
TECH DAY