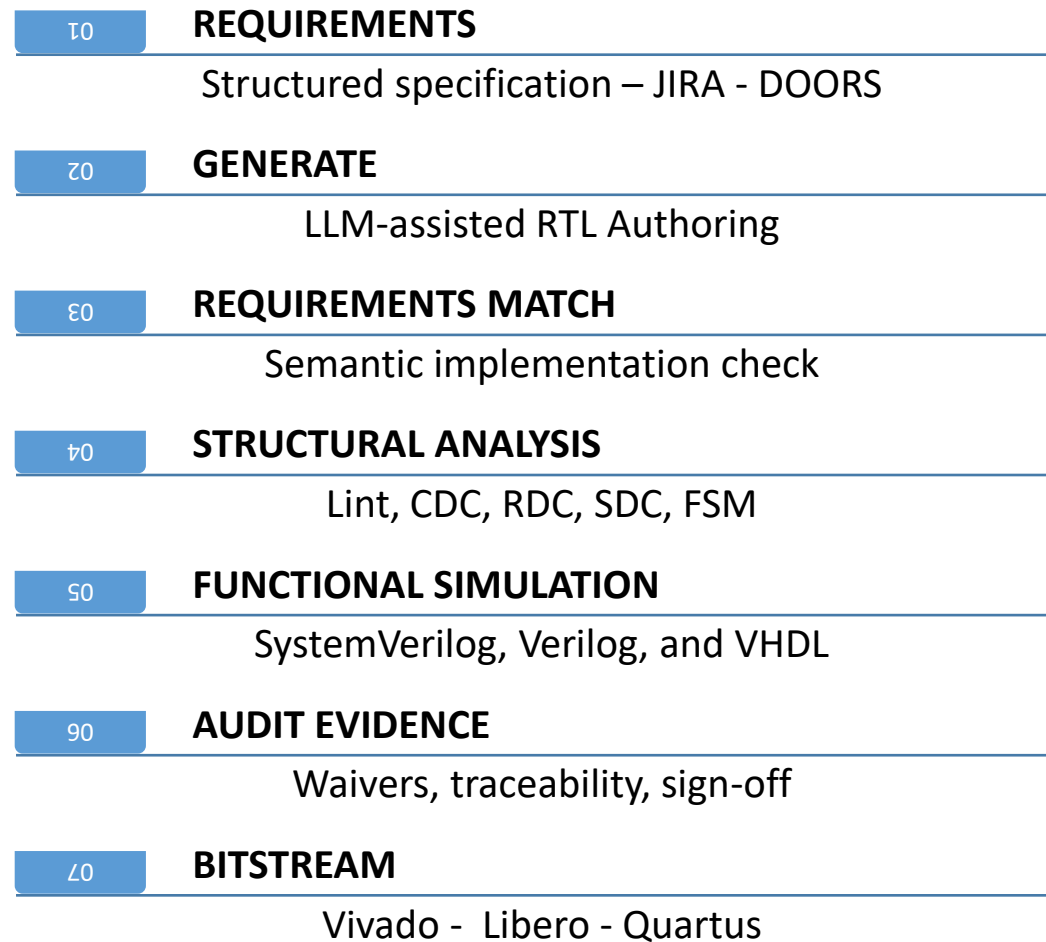


**Generate it. Prove it.
Sign off with Confidence.**

**AI-assisted high-reliability FPGA design and
verification**

Requirements to Bitstream

INTEGRATED VERIFICATION WORKFLOW



Editor and Generative AI

Natural language RTL generation

- RTL and testbench creation with generative AI
 - Smart-editor to verify as you code
 - RTL Analysis, dependency and lines of code views
 - Blue Pearl FPGA optimized LLM for on premise
 - Optional choice of your generative AI LLM
 - VSCode interface available

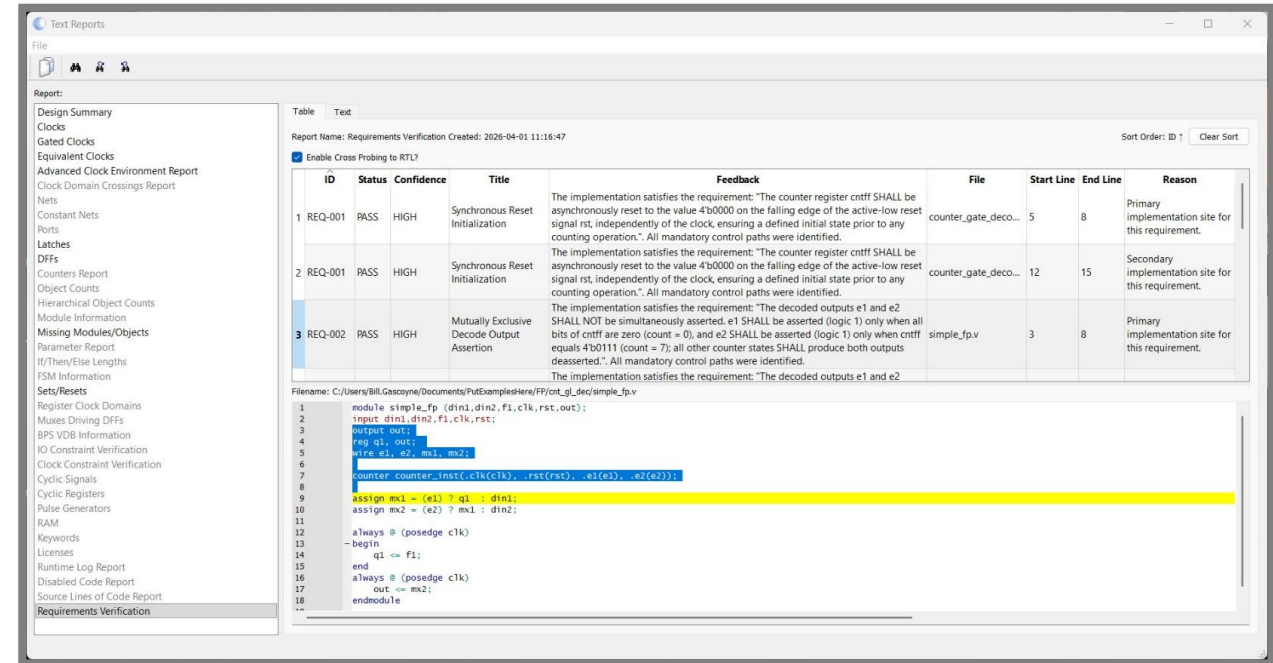
The screenshot displays the Blue Pearl Software Suite interface for a project named 'uart_controller'. It features several windows and panels:

- RTL Requirements:** A panel on the left with a 'Context files' section containing 'uart_controller.v' and a red '1. Add file' button. Below it are checkboxes for 'Include BPS messages' (checked) and 'Fix BPS-0396 Informational messages' (checked).
- RTL Generation Review:** A central dialog box titled 'RTL Generation Review' showing a side-by-side comparison of existing and generated RTL code. A red '4. Accept or reject from side-by-side comparison and editor' label points to the comparison area. The dialog includes 'Accept' and 'Cancel' buttons at the bottom.
- Code Editor:** The main window shows the 'uart_controller.v' file with Verilog code. A red '3. Click Generate button' label points to the 'Generate RTL' button at the bottom of the editor.
- RTL Browser:** A panel on the right showing a tree view of the design hierarchy, including 'uart_controller' and its sub-modules.
- Generated RTL:** A window on the right showing the generated Verilog code, including comments and logic for the UART controller.

BPS-AI Requirements Analysis

Ensure your design meets the requirements

- Starts with the specification requirements from Doors, Jama, Enterprise Architect etc.
- Catches functional issues early in the verification cycle
 - 480 billion parameter “thinking” LLM model
 - Analyzes code against requirement to determine probability RTL implements the requirement correctly
 - Cross probes to code, to reinforce learning
- Helps with third party assessor for safety critical applications e.g. ESA / DO-254
- Future security extension to conform to EU-CRA
- Available on prem or off



The screenshot displays the 'Text Reports' window. On the left is a navigation pane with a tree view of report categories. The main area shows a table of requirements with columns for ID, Status, Confidence, Title, Feedback, File, Start Line, End Line, and Reason. Below the table is a code editor showing the corresponding RTL code for the selected requirement.

ID	Status	Confidence	Title	Feedback	File	Start Line	End Line	Reason
1 REQ-001	PASS	HIGH	Synchronous Reset Initialization	The implementation satisfies the requirement: "The counter register cntff SHALL be asynchronously reset to the value 4'b0000 on the falling edge of the active-low reset signal rst, independently of the clock, ensuring a defined initial state prior to any counting operation". All mandatory control paths were identified.	counter_gate_deco...	5	8	Primary implementation site for this requirement.
2 REQ-001	PASS	HIGH	Synchronous Reset Initialization	The implementation satisfies the requirement: "The counter register cntff SHALL be asynchronously reset to the value 4'b0000 on the falling edge of the active-low reset signal rst, independently of the clock, ensuring a defined initial state prior to any counting operation". All mandatory control paths were identified.	counter_gate_deco...	12	15	Secondary implementation site for this requirement.
3 REQ-002	PASS	HIGH	Mutually Exclusive Decode Output Assertion	The implementation satisfies the requirement: "The decoded outputs e1 and e2 SHALL NOT be simultaneously asserted; e1 SHALL be asserted (logic 1) only when all bits of cntff are zero (count = 0), and e2 SHALL be asserted (logic 1) only when cntff equals 4'b0111 (count = 7); all other counter states SHALL produce both outputs deasserted". All mandatory control paths were identified. The implementation satisfies the requirement: "The decoded outputs e1 and e2	simple_fp.v	3	8	Primary implementation site for this requirement.

```
1 module simple_fp (din1, din2, f1, clk, rst, out);
2   input din1, din2, f1, clk, rst;
3   output out;
4   reg q1, out;
5   wire e1, e2, mx1, mx2;
6   counter_counter_inst(c1k(clk), rst(rst), e1(e1), e2(e2));
7
8   assign mx1 = (e1) ? q1 : din1;
9   assign mx2 = (e2) ? mx1 : din2;
10
11   always @ (posedge clk)
12   -begin
13     q1 <= f1;
14   -end
15   always @ (posedge clk)
16     out <= mx2;
17
18 endmodule
```

Analyze RTL AI

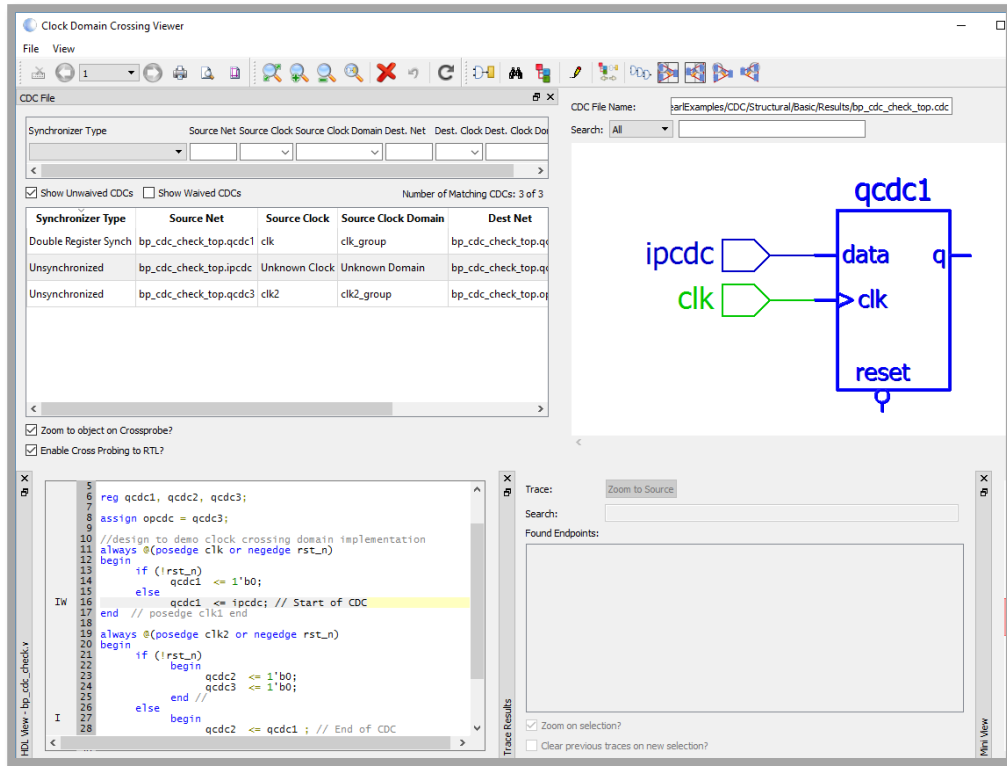
Clean and readable code while avoiding potential issues

- Stimulus-free advanced RTL analysis
- Integrated AI generated code fixes with your LLM of choice
- Company Specific and Industry Standard coding style checks before simulation and synthesis
- Over 400 checks that can be grouped into company-specific checklists
- Deterministic
- Includes DO-254, STARC, FPGA and ASIC packages



Clock and Reset Domain Crossing Analysis AI

Avoid metastability issues



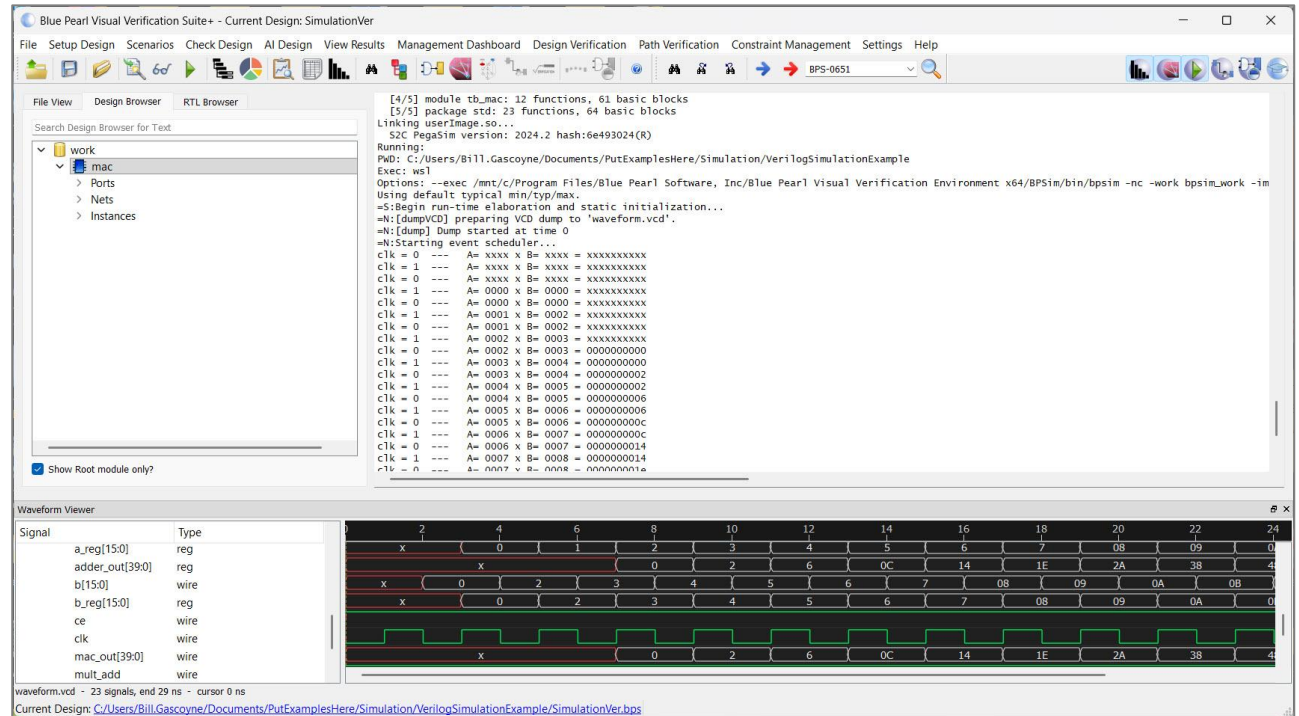
Clock Domain Crossing Analysis

- Easily runs CDC / RDC analysis using different modes
- Integrated AI generated code fixes with your LLM of choice
- Tcl parser to read in familiar inputs where clocks and domains have already been defined
- Identifies synchronization issues between interacting clocks and reset
- Patented User Grey Cells enable verification with black boxes, encrypted and hard IP
- Generates SDC and assertions for simulation

BPSim – RTL Simulation

High-performance RTL simulator at a fraction of the cost

- SystemVerilog, Verilog, and VHDL
- UVM, VMM, and OVM
- Cocotb support
- SDF and timing checks
- Full DPI and VPI support
- Incremental and parallel compilation
- Competitive compilation and runtime



Signoff With Confidence

- An LLM can generate RTL in seconds, however it cannot produce the required certification evidence including **traceability**, **validation** and **verification**!
- Our solution is built around three tenets required for DO-254 or IEC 61508 audits

Deterministic

The same RTL, the same constraints, the same report — every run, every engineer, every machine

No probabilistic output

No drift between sessions

Auditable

Every finding traceable to a named rule

Every waiver documented with rationale

Built for the audit trail your certification authority will read

Reproducible

Identical input yields identical output, on Windows or Linux, for node-locked or floating licenses, in command line or GUI

What you sign off is what your successor will see five years from now

To schedule a private demo while at DAC,
please contact
info@bluepearlsolutions.com