

Quick start

From the download to your first waveforms.

Version 0.8.0

JanusHDL quick start

JanusHDL is an HDL simulator: VHDL-1993 to VHDL-2019 with PSL today, Verilog and SystemVerilog next. Free of charge, for anyone and any use. This page takes you from the download to your first waveforms in about ten minutes.

1. Install

Windows 10 or 11. Unzip the download anywhere, for example to `C:\Tools\janushdl`, and add its `bin` folder to your `PATH` (or run the programs from there).

Linux (x86-64 or 64-bit ARM, glibc 2.28 or later):

```
mkdir -p ~/tools
tar -xzf janushdl-0.7.1-linux-x86_64.tar.gz -C ~/tools
export PATH=~/tools/janushdl-0.7.1-linux-x86_64/bin:$PATH
```

Check it: `jhd1 --version` prints the version. The download holds three programs: `jhd1` (the command line), `jhd1w` (the window) and `jhd1-lsp` (a language server for your own editor).

2. Your first simulation

Save these two files in an empty folder.

counter.vhd :

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity counter is
    port (clk, rst : in std_logic; q : out unsigned(3 downto 0));
end entity;

architecture rtl of counter is
    signal n : unsigned(3 downto 0) := (others => '0');
begin
    process (clk) begin
        if rising_edge(clk) then
            n <= (others => '0') when rst = '1' else n + 1;
        end if;
    end process;
    q <= n;
end architecture;
```

counter_tb.vhd:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity counter_tb is end entity;

architecture sim of counter_tb is
    signal clk, rst : std_logic := '1';
    signal q : unsigned(3 downto 0);
begin
    clk <= not clk after 5 ns;
    dut : entity work.counter port map (clk, rst, q);
    process begin
        wait for 12 ns;
        rst <= '0';
        wait for 100 ns;
        assert q = 10 report "q is " & to_string(to_integer(q)) severity error;
        report "done: q = " & to_string(to_integer(q));
        std.env.finish;
    end process;
end architecture;
```

Run it:

```
> jhdl counter.vhd counter_tb.vhd --top counter_tb
112 ns (delta 0) counter_tb.vhd:18:12: note: done: q = 10
simulation of counter_tb ended at 112 ns (stopped); analysis 24 ms, elaboration 1 ms, run 1 ms
```

`jhdl` analyzes the files in the order given, elaborates the top and runs it. The exit code is 0, or 1 if a report or assertion of severity `error` or `failure` came up, so scripts and CI can rely on it.

3. See the waveforms

Record every signal while the simulation runs, then open the recording:

```
jhdl counter.vhd counter_tb.vhd --top counter_tb --wave run.jhw
jhdw run.jhw
```

Or do it all in the window: start `jhdw`, choose **Simulate** → **Run simulation...** (Ctrl+Shift+R), and pick `counter_tb.vhd` or a file list. The waves grow as the simulation runs. Double-click signals in the **Signals** panel to add them to the waves.

A recording of any size opens: the window reads only what the screen shows.

4. Bigger designs

List the files in a **file list** and give it with `-f`; `library NAME` lines choose the library of the files after them:

```
# project.f
library mylib
src/fifo.vhd
src/top.vhd
library work
tb/top_tb.vhd
```

```
jhdl1 -f project.f --top top_tb --stop-time 10us --wave run.jhw
```

Useful options: `--std 93|02|08|19` (VHDL version, 2008 by default), `--stop-time 10us`, `-gWIDTH=16` (a generic of the top), `--cache FILE` (skip the analysis when no file changed). `jhdl1 --help` lists them all.

For a design you come back to, a **project file** (`.jhd1proj`) keeps the sources with their libraries and VHDL versions, the top, its generics and the options: **File** → **New project...** in the window, or `jhdl1 run my.jhd1proj` on the command line.

5. UVVM, OSVVM and VUnit

Install them from their own projects and analyze their sources with your design: their files first, each in its own library, then yours. UVVM declares a few shared variables of types that are not protected, which the standard forbids since VHDL-2002: run it with `--compat` (section 6). OSVVM runs without it. The UVVM tutorial on the web site gives a complete file list for UVVM's demos. VUnit's `run.py` scripts run through the front ends described next.

6. Your existing scripts

The `compat` folders of the download hold small programs named like the commands your scripts already call. Put one of them first on your path instead of `bin`, and your scripts run on JanusHDL as they are.

Code that relies on something the standard forbids (some libraries do, UVVM among them) runs with `--compat`: each such spot is then a warning with the section of the standard, instead of an error, and the error says so when you need it. The front ends in `compat` always use it.

7. The window at a glance

To	Press
Find any command	Ctrl+Shift+P or F1
Run a simulation / continue / pause	Ctrl+Shift+R / F5 / F6
Zoom to fit / in / out	F / + / -

To	Press
Jump to the previous / next change	Ctrl+← / Ctrl+→
Place the cursor / the second cursor	click / Shift+click
Add a marker at the cursor	M
Find a value of the selected signal	/ then Enter
Open a file in the code editor	Ctrl+E
Show or hide the terminal	Ctrl+J

Right-click a signal's name for its radix, color, analog drawing and groups. The **Why this value?** panel shows the drivers of the selected signal at the cursor, and what they read; the **Delta cycles** panel shows what happened within one time step.

Where next

The full manual, the tutorials and the specifications are on the web site, at que.no/janushdl.

Which way of working fits you: the window with or without a project, the command line, your existing scripts, or your own editor? que.no/janushdl/ways-to-work.html sums them up and says where each is explained.