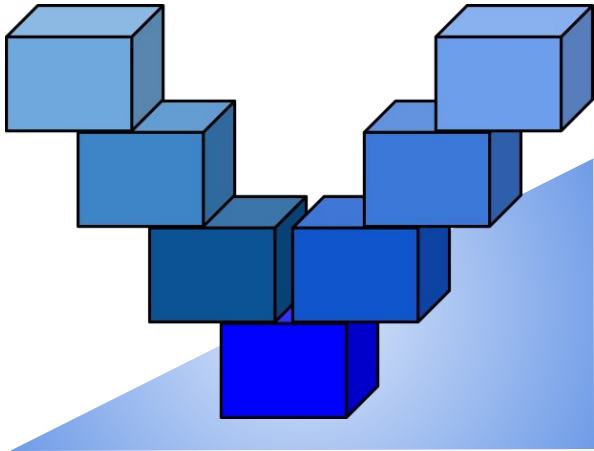
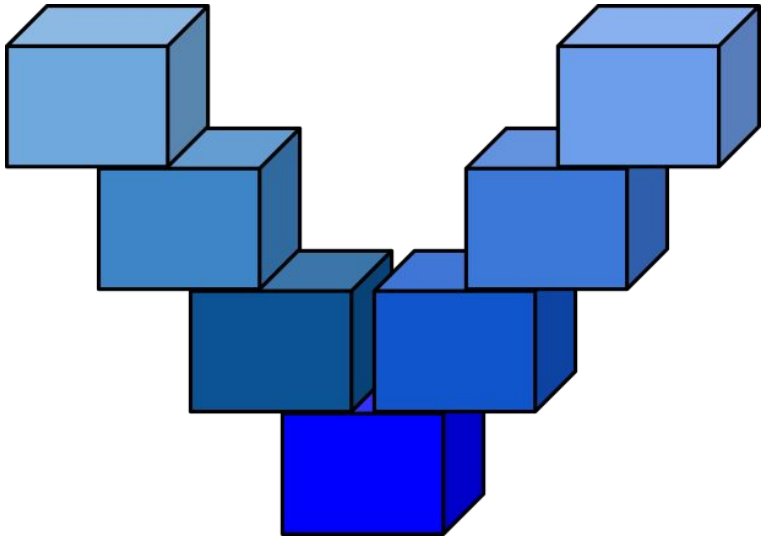
SystemVerilog IDE integration with Verible Language Server Support

Henner Zeller | WOSET 2021



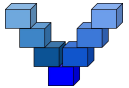
Verible is a suite of SystemVerilog tools

- Verible is a suite of SystemVerilog code quality and introspection tools.
- Started by David Fang, shown on WOSET 2018, published by Google in 2019 under Apache license and now under Chips Alliance roof.



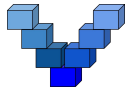
Tools

- Command line
  - *direct project use*
- Github integration
  - *automatic review*
- Language server
  - *features available directly in editor/IDE*



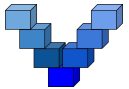
Command Line Utility: Linting with

`verible-verilog-lint`



Avoid potential bugs with linter

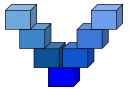
```
verible@WOSET-2021$ cat truncated_numeric_literal.sv
module truncated_numeric_literal;
  assign a = 4'h1F;
endmodule
verible@WOSET-2021$ verible-verilog-lint truncated_numeric_literal.sv
truncated_numeric_literal.sv:2:17: Number 4'h1F occupies 5 bits, truncated to 4 bits. [Style: number-lit
erals] [truncated-numeric-literal]
verible@WOSET-2021$
```



Highlight readability issues

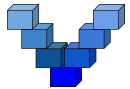
For example if you always want to have bit-patterns written with all digits, e.g. `4'b0001`
… there is a rule for that.

```
verible@WOSET-2021$ cat undersized_binary_literal.sv
module undersized_binary_literal;
  localparam logic [3:0] Foo = 4'b1;
endmodule
verible@WOSET-2021$ verible-verilog-lint undersized_binary_literal.sv
undersized_binary_literal.sv:2:35: Binary literal 4'b1 has less digits than expected for 4 bits. [Style:
 number-literals] [undersized-binary-literal]
verible@WOSET-2021$
```



… and apply available autofix

```
verible@WOSET-2021$ verible-verilog-lint --autofix=inplace-interactive undersized_binary_literal.sv
undersized_binary_literal.sv:2:35: Binary literal 4'b1 has less digits than expected for 4 bits. [Style:
 number-literals] [undersized-binary-literal]
[ 1. Alternative Left-expand leading zeroes ]
@@ -1,3 +1,3 @@
 module undersized_binary_literal;
-  localparam logic [3:0] Foo = 4'b1;
+  localparam logic [3:0] Foo = 4'b0001;
 endmodule
[ 2. Alternative Replace with decimal ]
@@ -1,3 +1,3 @@
 module undersized_binary_literal;
-  localparam logic [3:0] Foo = 4'b1;
+  localparam logic [3:0] Foo = 4'd1;
 endmodule
Autofix is available. Apply? [1,2,y,n,a,d,A,D,p,P,?] 1
verible@WOSET-2021$ cat undersized_binary_literal.sv
module undersized_binary_literal;
  localparam logic [3:0] Foo = 4'b0001;
endmodule
verible@WOSET-2021$
```



There are many rules available

- There are close to 60 rules available now, and it relatively easy to add more
- Each rule can be enabled or disabled in your project either with a command line or a configuration file.
- Waiver files allow for fine-grained control per file and line

Lint rules are auto-documented on this page

https://chipsalliance.github.io/verible/lint.html

Or on the command line

```
verible-verilog-lint --help_rules=all
```

macro-string-concatenation

Concatenation will not be evaluated here. Use `"..."` instead. See [Style: defines].

Enabled by default: false

mismatched-labels

Check for matching begin/end labels. See [Style: mismatched-labels].

Enabled by default: false

module-begin-block

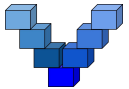
Checks that there are no begin-end blocks declared at the module level. See [Style: floating-begin-end-blocks].

Enabled by default: true

module-filename

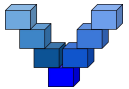
If a module is declared, checks that at least one module matches the first dot-delimited component of the file name. Depending on configuration, it is also allowed to replace underscore with dashes in filenames. See [Style: file-names].

Parameter



Command Line Utility: Formatting with

`verible-verilog-format`



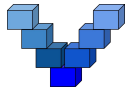
The formatter prettifies

In-place formatting

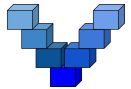
```
module my_module #(
    int               width  = 2,
    my_pkg::length_t length = 4
) (
    input  logic       clk,
    input  wire [7:0] data,
    output reg         result
);
  initial begin
    foo[0] <= 1'b1;
    bar    <= 1'b0;
    next   <= prev ^ xyz;
  end
endmodule
```

Interactive session...

```
$ verible-verilog-format-changed-lines-interactive.sh
[transform-interactive.sh]  Transformation: verible-verilog-format
[transform-interactive.sh]  Patch: /tmp/tmp.GEnxFtj/interactive.patch
--- opentitan/hw/ip/aes/rtl/aes_core.sv
+++ NEW/opentitan/hw/ip/aes/rtl/aes_core.sv
@@ -32,3 +32,3 @@
   // Signals
-  logic                 foo_bar;
+  logic foo_bar;
   logic                 ctrl_qe;
Apply this hunk? [y,n,a,d,s,q,?] n
@@ -107,3 +107,3 @@

-      logic   illogical;
+  logic illogical;

Apply this hunk? [y,n,a,d,s,q,?] n
```

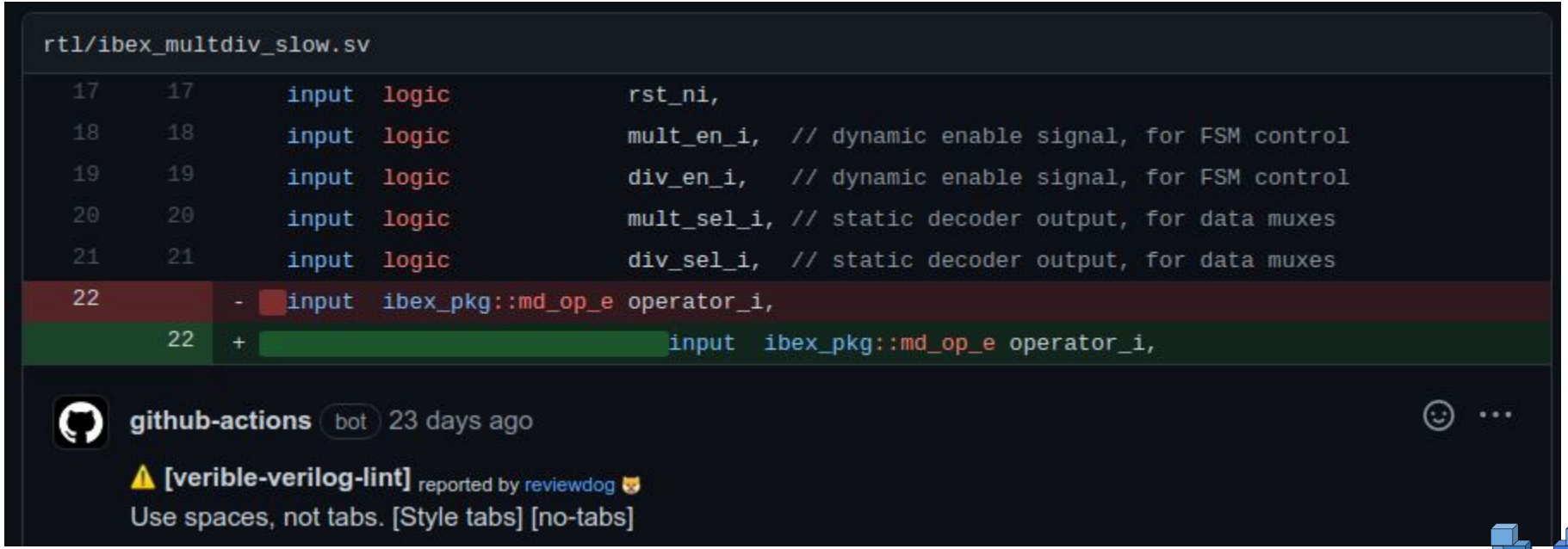


Github integration



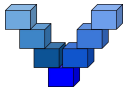
A github action provides reviews

Use the provided github action to directly comment in your pull requests
https://github.com/chipsalliance/verible-linter-action



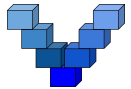
Verible: other tools

- `verible-verilog-project` -- **extracting dependencies, creating file-lists etc.**
- `verible-verilog-kyte-extractor` -- **Generate a kythe index to cross-reference code**
- `verible-verilog-diff` -- **lexical comparison to verify contents**
- `verible-verilog-preprocess`
- `verible-verilog-obfuscate` -- **obfuscated test case preparation**
  - Preserves byte/line offsets within files, to retain diagnostic locations
  - Supports multi-file operation, saving obfuscation maps, and decoding



IDE integration with Language server

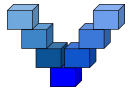
`verible-verilog-ls`



A Language Server augments your IDE

- The Language Server Protocol (LSP) is a standard protocol that allows external tools provide useful features to editors and IDEs. These tools are called language servers.
- Features can show syntax and linting issues, formatting code, providing fixes or refactoring features.
- Implementing the protocol, Verible now can provide its features directly in your IDE
- Most editors used for programming provide some LSP integration; `verible-verilog-ls` has been tested with emacs, vi, sublime, kate and vscode.

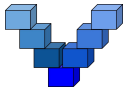
The Verible language server `verible-verible-ls` provides edit-enhancing features via the language server protocol.



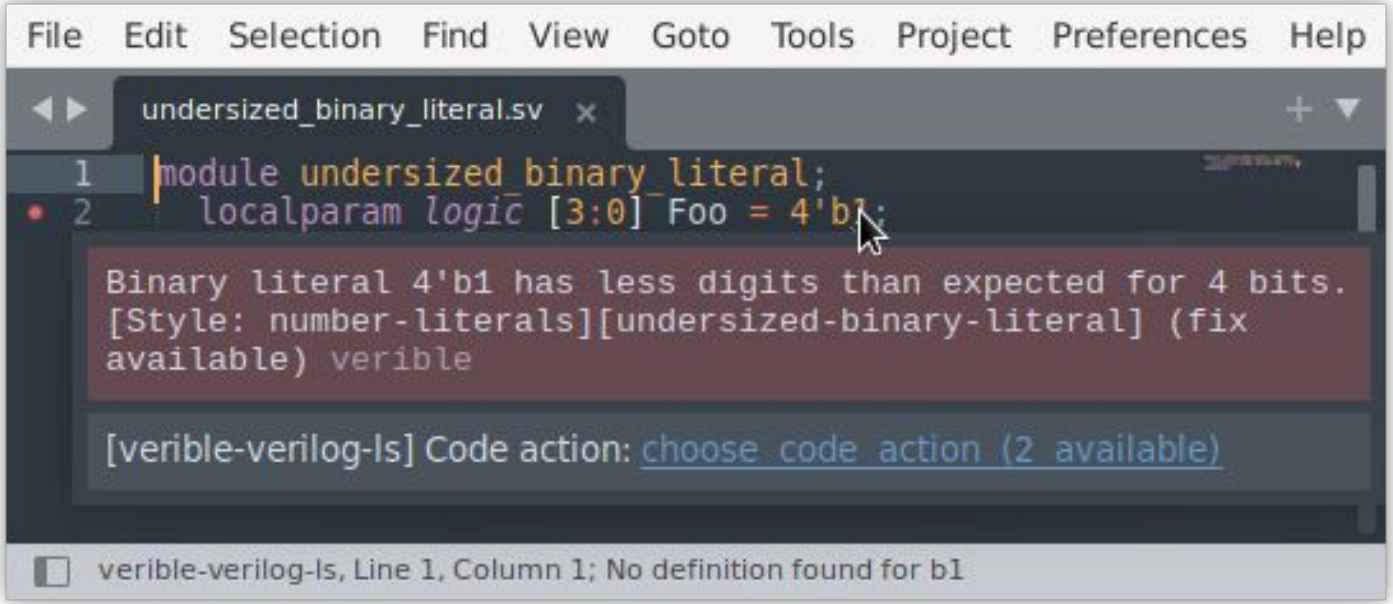
Find issues while typing...

```
    // Assume that rst_ni is low for the first cycle and not true after that.
    assume (~((f_startup_count == 2'd0) ^ ~rst_ni));

    // There is a feed-through path from branch_i to req_o which isn't squashed when in reset. Assume
    // that branch_i isn't asserted when in reset.
    assume (`IMPLIES(!rst_ni, !branch_i));
  end

  // Several of the protocol checks are only valid when there is a valid address. This is false
  // after reset. It becomes true after any branch after reset and then false again on any returned
  // error (because the straight-line address depends on the presumably-bogus rdata).
  logic f_addr_valid;
  always_ff @(posedge clk_i or negedge rst_ni) begin
    if (!rst_ni) begin
      f_addr_valid <= 1'b0;
    end else begin
      if (branch_i) begin
!       f_addr_valid = 1'b1;
      end else if (valid_o & ready_i & err_o) begin
!       f_addr_valid = 1'b0;
      end
    end
  end

  // Reset assumptions
-:**-  formal_tb.sv   13% L116  Git-master  (Verilog Flymake[0 0 35] WK EditorConfig LSP[verible-ls:22515])
```

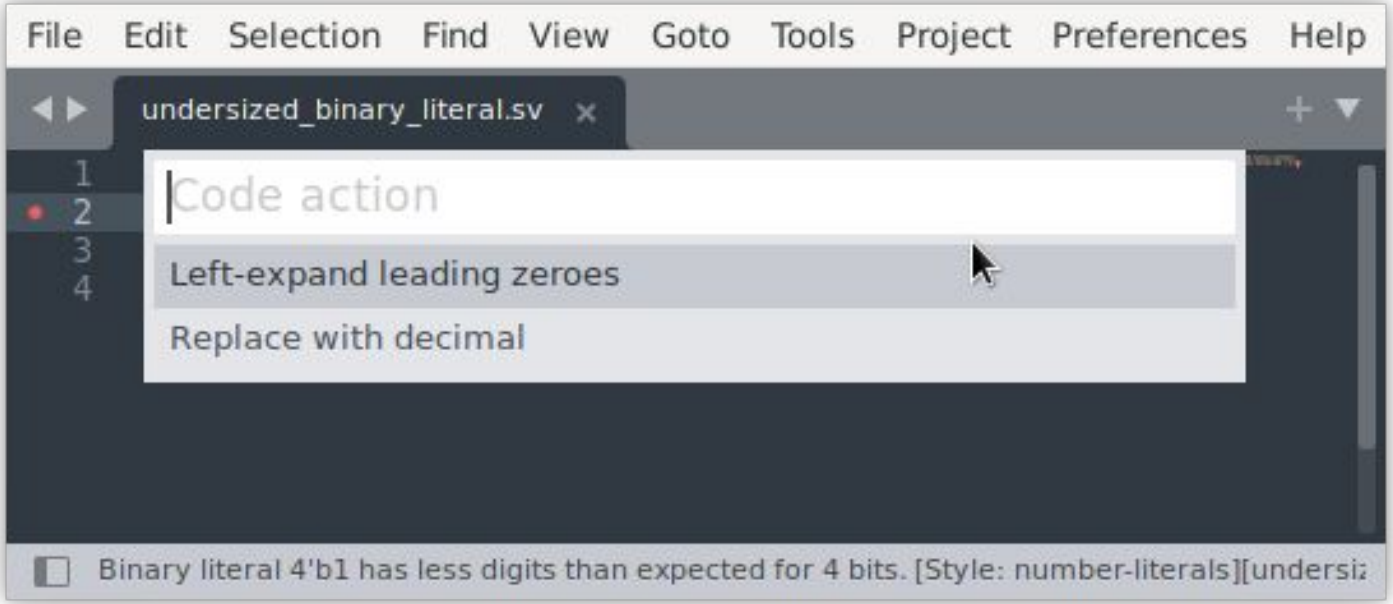
Use blocking assignments, at most, for locals inside 'always_ff' sequential blocks. [Style: sequential-logic][always-ff-non-blocking]



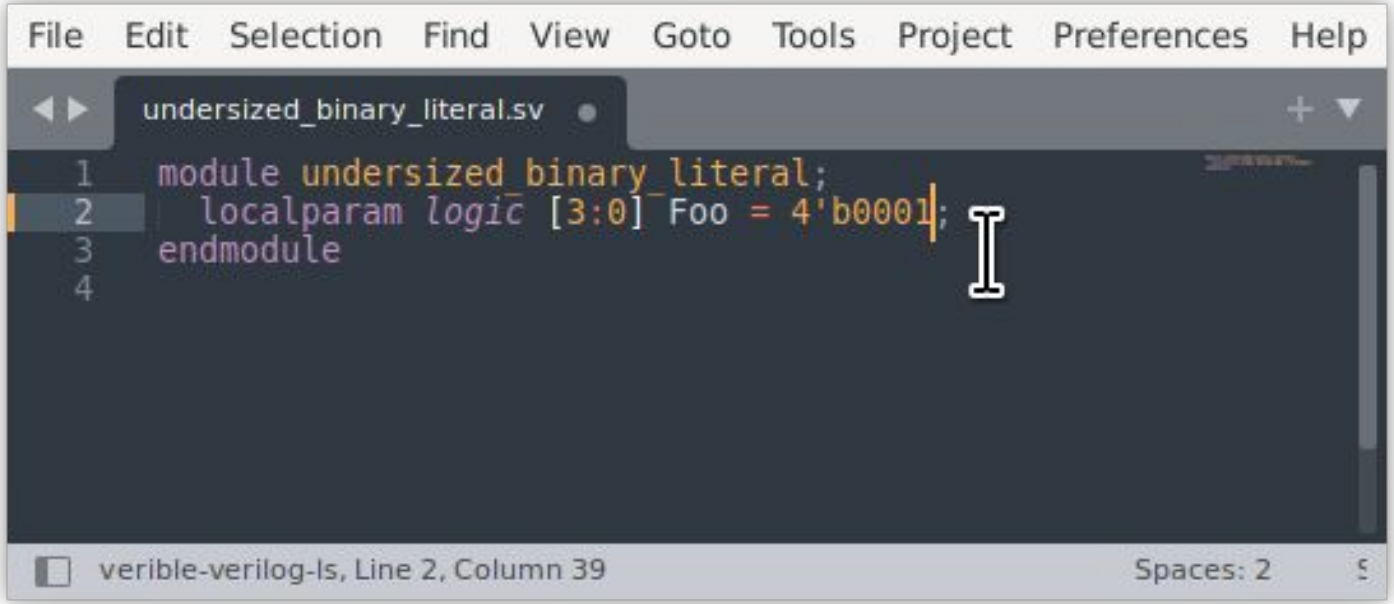
Language Server: show lint violation



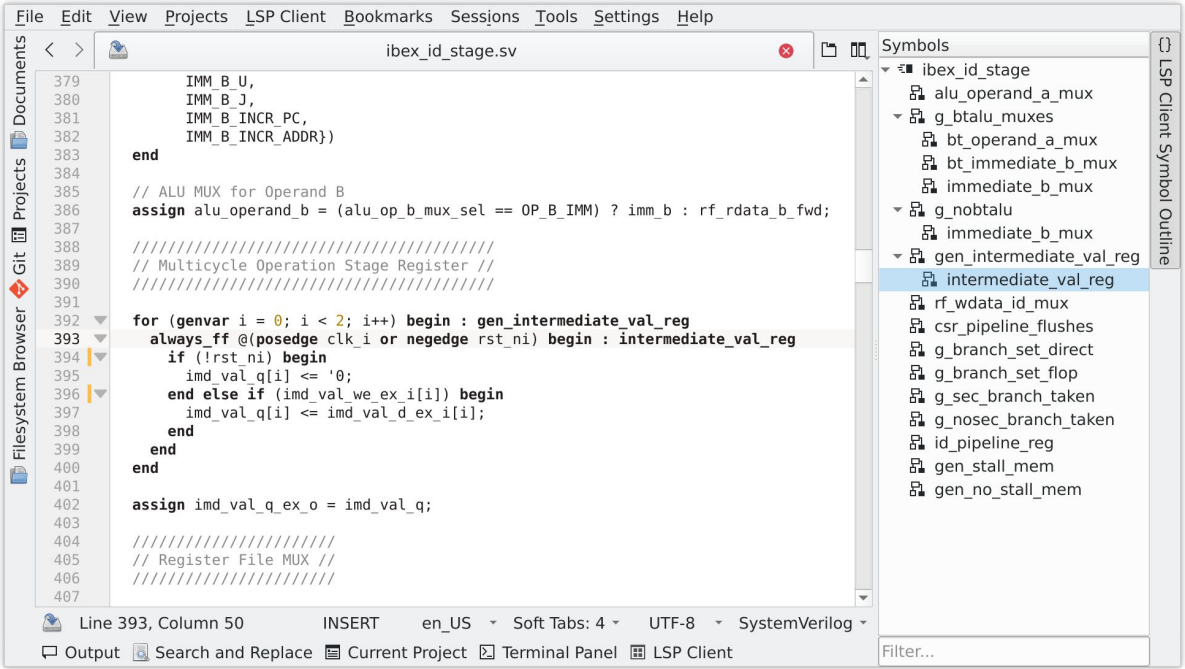
Language Server: choose the fix

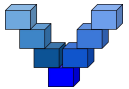


Done. Fix applied.

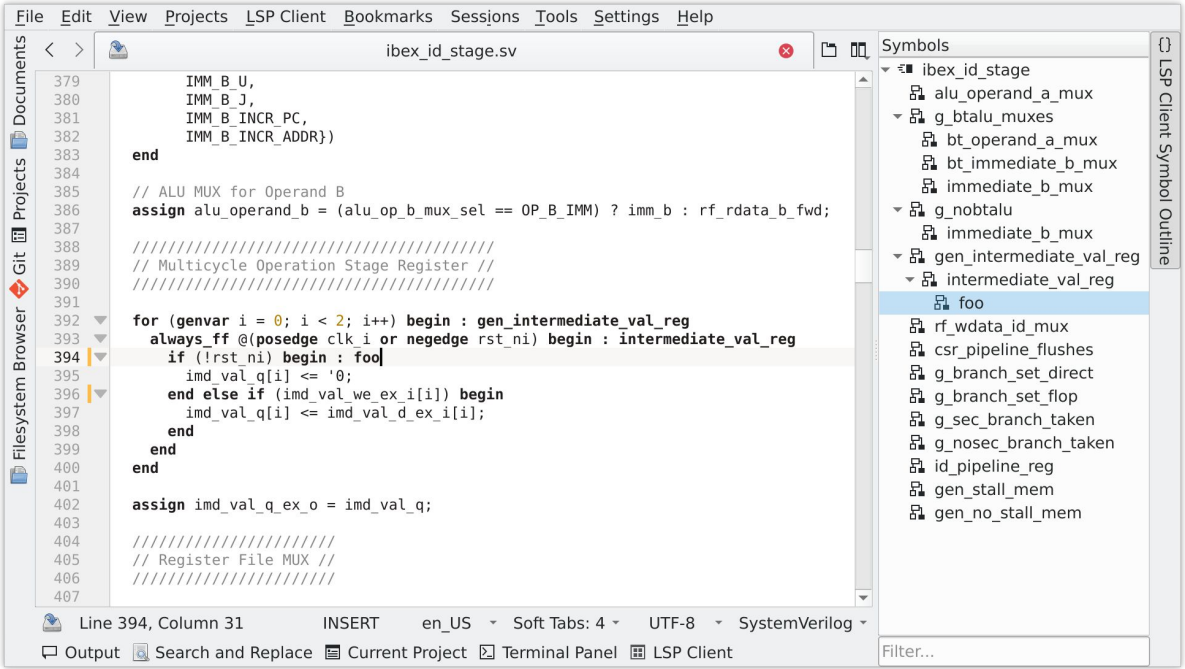


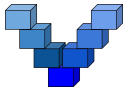
Hierarchy view shows block labels



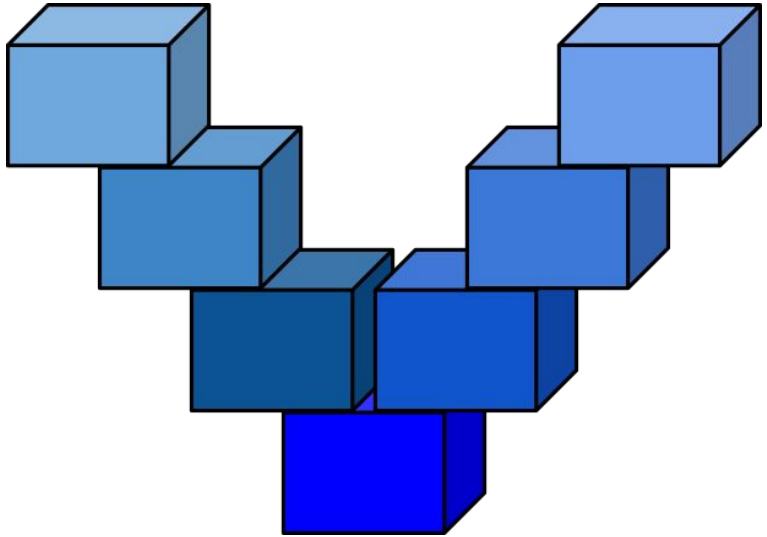


Immediate update with your edits





```
    end
  $finish;
endtask
endmodule
```



https://github.com/chipsalliance/verible