LSU EE 4755 Homework 1 Due: 14 September 2026

Formatted 17:54, 5 September 2026

For instructions visit `https://www.ece.lsu.edu/koppel/v/proc.html`. *For the complete Verilog for this assignment without visiting the lab follow* `https://www.ece.lsu.edu/koppel/v/2026/hw01.v.html`.

Collaboration Rules

Each student is expected to complete his or her own assignment. It is okay to work with other students and to ask questions in order to get ideas on how to solve the problems or how to overcome some obstacle (be it a question of Verilog syntax, interpreting error messages, how a part of the problem might be solved, etc.) It is also acceptable to seek out digital design resources for help on Verilog, digital design, etc. It is okay to make use of AI LLM tools such as ChatGPT and Copilot to generate sample Verilog code. (Do not assume LLM output is correct. Treat LLM output the same way one might treat legal advice given by a lawyer character in a movie: it may sound impressive, but it can range from sage advice to utter nonsense.)

After availing oneself to these resources **each student is expected to be able to complete the assignment alone**. Test questions will be based on homework questions and **the assumed time needed to complete the question will be for a student who had solved the homework assignment on which it was based**.

Problem 0: Following instructions at `https://www.ece.lsu.edu/koppel/v/proc.html`, set up your class account, copy the assignment, and run the Verilog simulator on the unmodified homework file, `hw01.v`. Do this early enough so that minor problems (*e.g.*, password doesn't work) are minor problems.

Homework Introduction

The point of this assignment is to exercise skills in module instantiation and declaring ports and parameters. To solve the problem one just connects modules together correctly, there is no real digital design. (Feel free to solve 2023 Homework 5 if you feel like you're missing out on design, though you might wait until later in the course.)

The assignment file includes a full testbench, one written for the 2023 assignment on which it was based. The detailed output was to be used to debug the design. For this (2026 Homework 1) assignment the output will indicate whether things are working, but students are not expected to use detailed error messages to figure out what went wrong.

There are differences in this assignment that affect the testbench output. This assignment includes an output named `match_pos` that did not appear in the 2023 assignment, and the testbench checks this new output. The testbench tests output `uniq_bvec`, which does not appear in this assignment. The value tested is produced by a solution to the 2023 assignment, `uniq_vector_seq`.

Testbench

To compile your code and run the testbench press `F9` in an Emacs buffer in a properly set up account. Once you start solving the problem, it is likely that you'll get compiler error messages, and so you won't see the testbench run until those are fixed. The testbench will test the module for Problem 1 in this assignment. The beginning of the testbench output, which may quickly scroll by, will look something like this:

```
Compilation started at Sat Sep  5 16:53:41
```

```
xrun -sv -batch -exit hw01.v
TOOL: xrun(64) 26.03-s005: Started on Sep 05, 2026 at 16:53:41 CDT
xrun(64): 26.03-s005: (c) Copyright 1995-2026 Cadence Design Systems, Inc.
file: hw01.v
module worklib.ereg:v
errors: 0, warnings: 0
```

At the end of the testbench output is a tally of the number of errors in each module. For an unmodified assignment the output near the end will be:

```
Trace, uniq_bvec: t=36,  1001
[  ,   ],  1,  0,  0,  1  <-- uniq_bvec
[69, 69], 37, 88, 88, 14  <-- Element
[ 0,  0],  0,  0,  0,  0  <-- Start
For n=4, s=0: done with 100000 tests. Errors 0 n_match, 0 match_pos.
End of tests n= 4, we=4, wc=3: 0 bvec errors, 0 n_match errors, 0 match_pos errors
for 99992 tests.
End of tests n= 4, we=8, wc=4: 0 bvec errors, 0 n_match errors, 0 match_pos errors
for 99992 tests.
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL: xrun(64) 26.03-s005: Exiting on Sep 05, 2026 at 17:19:30 CDT  (total: 00:00:01)
```

The tally of errors are shown in the lines starting `End of tests`. The `uniq_vec_str` module is instantiated twice, the end-of-tests message shows the values of `n`, `we`, and `wc` used in the instantiation.

The output above is for a correct solution, so all error tallys are zero. The output below is from an unmodified assignment. The `bvec` error tally is zero because the module in this assignment does not generate that signal.

```
Error:  n_match: z != 1 (correct)
Error:  match_pos: z != 4 (correct)
[  ,   ],  1,  0,  0,  1  <-- uniq_bvec
[69, 69], 37, 88, 88, 14  <-- Element
[ 0,  0],  0,  0,  0,  0  <-- Start
For n=4, s=0: done with 100000 tests. Errors 99992 n_match, 99992 match_pos.
End of tests n= 4, we=4, wc=3: 0 bvec errors, 99992 n_match errors, 99992 match_pos
errors for 99992 tests.
End of tests n= 4, we=8, wc=4: 0 bvec errors, 99992 n_match errors, 99992 match_pos
errors for 99992 tests.
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
TOOL: xrun(64) 26.03-s005: Exiting on Sep 05, 2026 at 17:18:29 CDT  (total: 00:00:01)
```

For this assignment it is not necessary to understand the lines ending `uniq_bvec`, `Element`, and `Start`. (Those who are curious can look up 2023 Homework 5.) Helpful for this assignment are the lines showing `n_match` and `match_pos`. If their values are `z` (as they are above) that means that nothing is connected to the nets for `n_match_0` and `match_pos_0` (see module `uniq_vector_str`).

Common Problems

Here are some common errors messages, which will be encountered when the code is compiled (for example, by pressing F9).

```
file: hw01.v
   mult mym( p, a[0], b[0] );
              !
xmvlog: *E,NODFNT (hw01.v,105!13): Implicit net declaration (p) is NOT allowed, since
'default_nettype is declared as NONE [19.2(IEEE 2001)].
```

The problem above is that `p` was never declared. "Implicit net declaration" refers to a Verilog feature in which an object is assumed to be a `wire` if it had not been declared. That feature has been turned off since it can hide typos. The solution is to declare something like `uwire [um-1:0] p;`.

```
   abc #(p1,p2) abc0( abc_out, c2, c2,
                                 |
xmelab: *F,CUVMPW (./hw01-sol.v,59|28): port sizes differ in port connection(8/9) for
the instance(testbench.genblk1[1].tb.genblk1.uvs.col0) .
xrun: *E,ELBERR: Error (*E) or soft error (*SE) occurred during elaboration (status
2), exiting.
```

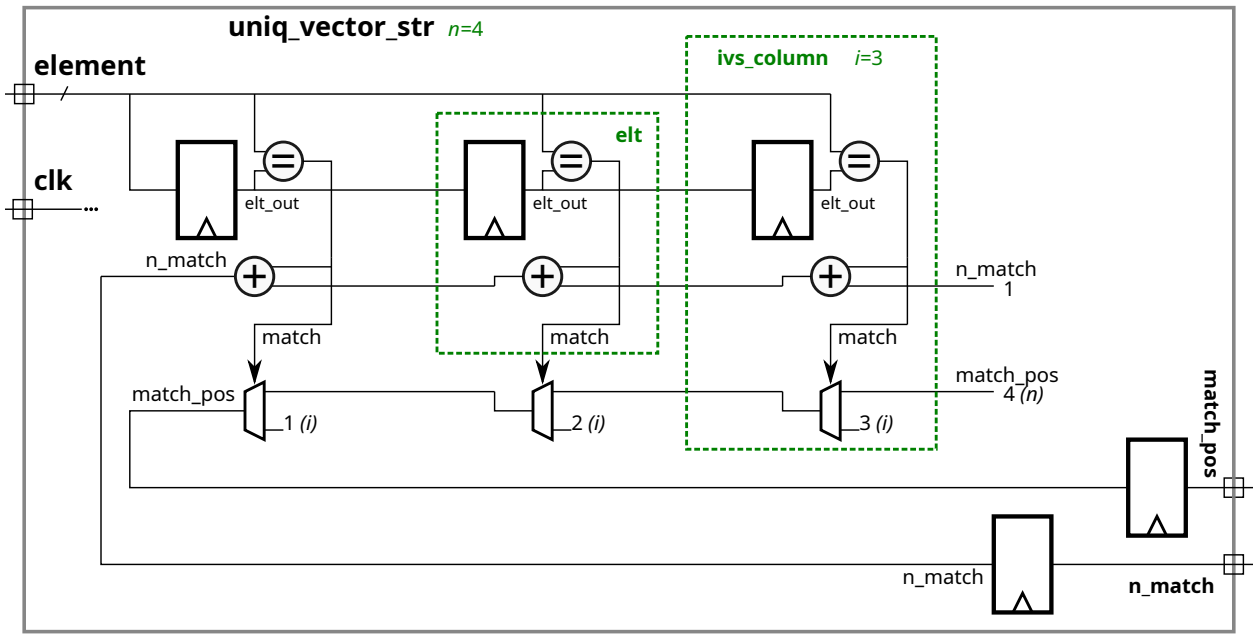
The problem above is due to the number of bits in `abc_out` being different than the number of bits in the first port of `abc`. The error can be fixed by either changing the declaration of `abc_out` or by changing the declaration to the first port of `abc`.

Helpful Examples

An easier past assignment like this one is 2025 Homework 1. In the 2025 assignment the hardware is simpler and consists only of 1-bit wires. Also look at 2019 Homework 1, in which one needs to code modules based on hardware shown in green boxes in a circuit diagram. Another good past assignment to look at is 2024 Homework 1. Like this assignment, a module, say `dot4`, is completed by instantiating other modules.

Unlike the others, in this assignment the major challenge is to correctly come up with the port and parameter declarations for one of the modules.

Problem 1: Appearing below is a diagram of a module, `uniq_vector_str`, based on 2023 Homework 5 and on 2023 Final Exam Problem 1, though extensively simplified. In the diagram there is a green rectangle labeled `elt` and a green rectangle labeled `ivs_column`. Notice that the hardware labeled `elt` appears three times, including an appearance in `ivs_column`. Also notice that the hardware in the `ivs_column` green box also appears three times. As some of you may have guessed by now, the Verilog for `uniq_vector_str` will instantiate `ivs_column` three times, which itself will instantiate `elt`.



In the homework file, `hw01.v`, there is a completed module named `elt` implementing the `elt` hardware in the diagram. The file also has an incomplete module named `ivs_column`, which when finished should implement the `ivs_column` hardware in the diagram. The file also has an incomplete module `uniq_vector_str`, which when finished should implement the illustrated hardware.

Complete modules `ivs_column` and `uniq_vector_str` following these guidelines:

☐ In module `ivs_column` there should be ☐ one instantiation of `elt` and ☐ one instantiation of `mux2` (a two-input multiplexor).

☐ In module `uniq_vector_str` there should be three instantiations of `ivs_column`.

☐ Provide ports on `ivs_column` that are needed to connect the three instances to each other and to hardware in `uniq_vector_str`. For example, there should be an input port for `element`, and both input and output ports for the `n_match` values.

☐ Also provide parameters on `ivs_column` and use those parameters in the instantiation of `elt` and ☐ set values for those parameters when instantiating `ivs_column` in `uniq_vector_str`.

☐ The lower input to the multiplexor is set to a value labeled `i`. That value `i` should be from a parameter, not an input.

- [ ] It will be necessary to cast the value of `i` to an packed vector of the correct size. Use `S'(V)` to cast object `V` into an `S`-bit vector.

  An important part of this assignment is correctly declaring the ports, including their sizes. Module `uniq_vector_seq` uses values of three sizes, 1 bit, such as `match`, `we`-bit, such as `element`, and `wc`-bit, such as `n_match`. Be sure to declare ports carrying values with the correct sizes.

- [ ] **Do not** use hard-coded sizes for ports (except of course for 1-bit values). Instead use parameters. See the `elt` module.

- [ ] Code `ivs_column` so that it can be used for any value of `n`, even though for this assignment `n=4`.

- [ ] Of course, the modules should compile without warnings and errors.

- [ ] The testbench should show zero errors.