**Introduction to SystemVerilog HDVL**

**Lab Manual**

**Lab 1: SystemVerilog Code Coverage and Assertions Coverage**

**Objective: *The objective of this lab is to get introduced to***

* **SystemVerilog Code Coverage and Assertions Coverage**

**Code Coverage**:

* Code coverage tells how well the HDL code has been exercised by the testbench.
* Code coverage reports about the efficiency of the test cases that is generated by the testbench.
* It is also known as quantitative coverage or structural coverage or implementation coverage since it reports the efficiency of the verification implementation.

**Types of Code Coverage:**

* Statement or Line Coverage
* Block or Segment Coverage
* Condition/ Expression Coverage /Focused Expression Coverage
* Branch/ Decision Coverage
* Toggle Coverage
* Path Coverage or Extended Branch Coverage
* FSM coverage

**Statement Coverage /Line Coverage:**

Statement or Line Coverage gives an indication of how many executable statements (lines) are covered in the simulation, by excluding lines like module, endmodule, comments, timescale etc.

This coverage is very important in all kinds of design and must be 100% for verification closure.

Statement coverage includes procedural statements, continuous assignment statements, conditional statement, and Branches for conditional statements etc.

**Block or Segment Coverage:**

A group of statements which are in the begin-end or if-else or case or wait or while loop or for loop etc. is called a block.

Block coverage gives the indication that whether these blocks are covered in simulation or not.

The dead-code in design code can be found by analyzing block coverage.

**Condition or Expression Coverage:**

This gives an indication how well variables and expressions (with logical operators) in conditional statements are evaluated.

Conditional coverage is the ratio of number of cases evaluated to the total number of cases present.

If an expression has Boolean operations like XOR, AND, OR ,etc the entries which is given to that expression to the total possibilities are indicated by expression coverage.

**Branch or Decision Coverage:**

In Branch coverage or Decision coverage reports, conditions like if-else, case and the ternary operator (?: ) statements are evaluated in both true and false cases.

**Toggle Coverage:**

Toggle coverage gives a report that how many times signals and ports are toggled during a simulation run. It also measures activity in the design, such as unused signals or signals that remain constant or less value changes.

**Path Coverage or Extended Branch Coverage:**

Path coverage tests all paths in the code.  A code path is the execution of the module’s entry point to the exit point.

Path coverage is more complicated than statement and branch coverage because the code may contain an unlimited number of paths.

**FSM coverage:**

FSM coverage reports, whether the simulation run could reach all the states and cover all possible transitions or arcs in a given state machine.

This is a complex coverage type as it works on behavior of the design, that means it interprets the synthesis semantics of the HDL design and monitors the coverage of the FSM representation of control logic blocks.

**Limitations of Code Coverage:**

Code coverage is an important indication for the verification engineer on how well the design code has been executed by the tests.

But it does not know anything about the design and what the design is supposed to do.

There is no way to find what is missing in the RTL code, as code coverage can only tell quality of the implemented code.

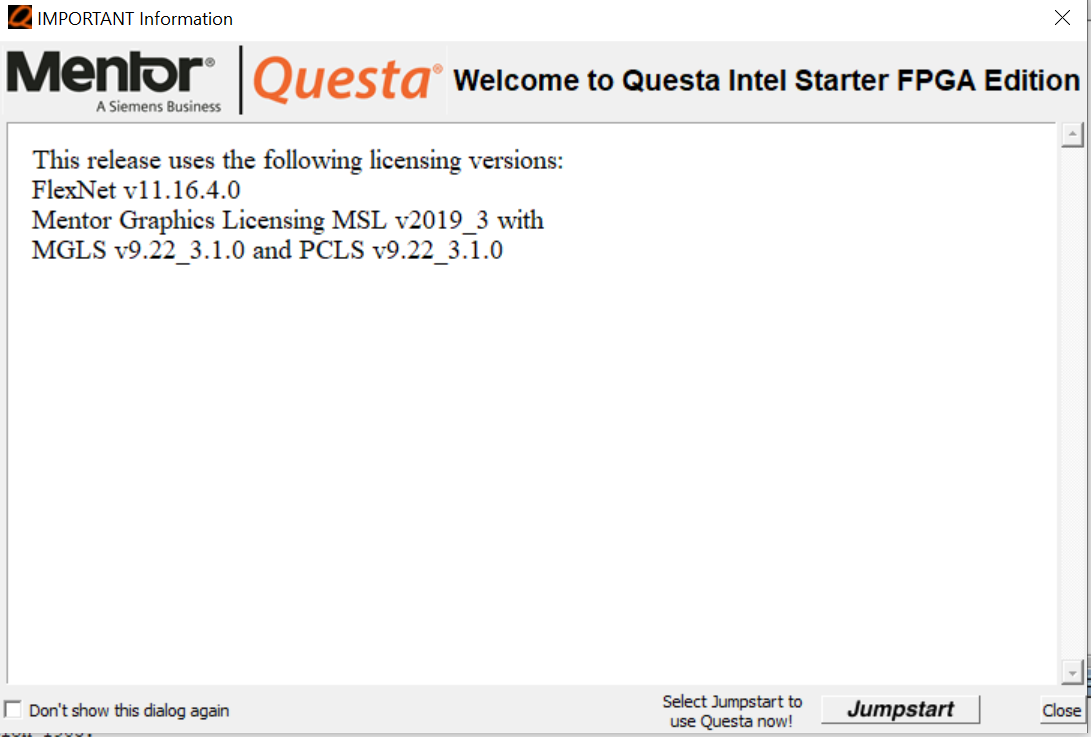
**Code Coverage Analysis in Questa – Intel FPGA starter edition**

The **Questa – Intel FPGA starter edition** supports the following coverages

* Statement or Line Coverage
* Condition/ Expression Coverage /Focused Expression Coverage
* Branch/ Decision Coverage
* Toggle Coverage
* FSM coverage

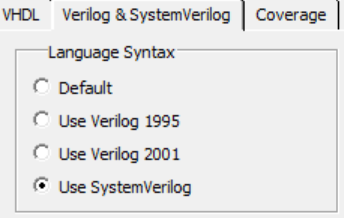
**Step 1: Compile the given RTL and TB files with code coverage options**

* Launch the **Questa-Intel FPGA starter edition** from the Windows start menu
* Close the Questa introductory window

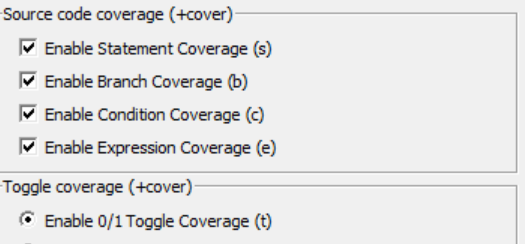


1. Set the project directory. From the QuestaSim File menu, select Change Directory. Browse to the location SV\_assertions/ lab1
2. Inspect the assertion\_ALU.sv and assertion\_ALU\_tb.sv files
3. Set the compile options to SystemVerilog and enable the code coverage metrics as shown below.

* In the Questa main window, Click Compile and select compile options and select Use SystemVerilog in the Verilog & SystemVerilog tab as shown below

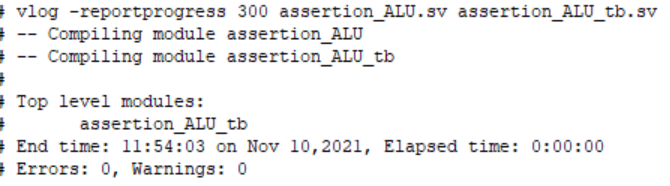


* Click the coverage tab and select the Enable statement, branch, condition, expression and toggle coverages as shown below



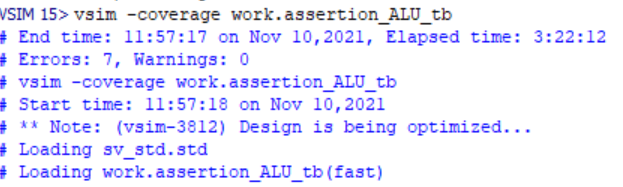
* Compile the assertion\_ALU.sv and assertion\_ALU\_tb.sv files using the vlog command

vlog assertion\_ALU.sv assertion\_ALU\_tb.sv

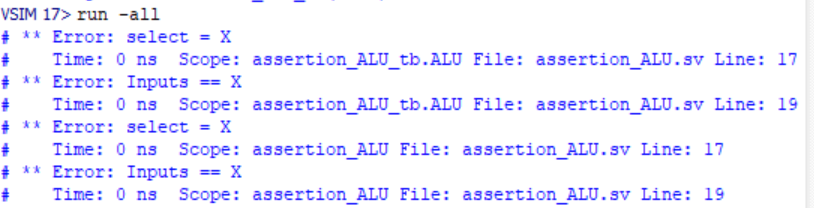


**Step 2: Simulating the testbench**

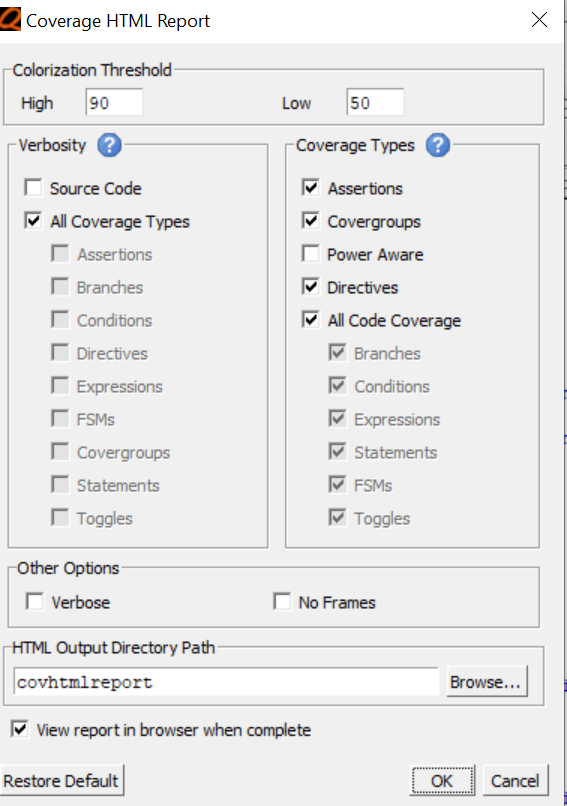
* Simulate the testbench with the code coverage options by the following command

 vsim -coverage assertion\_ALU\_tb

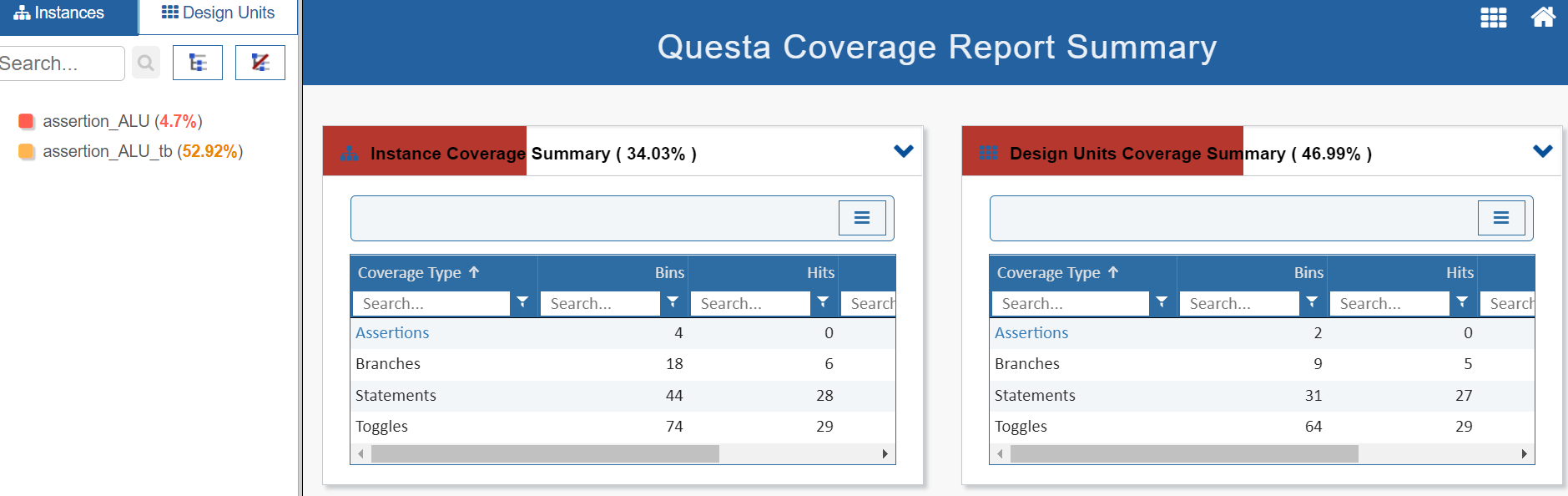
**Step 3: Run and analyze the code coverage and assertion reports**

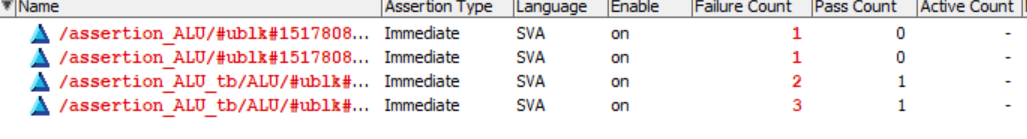


* Click tools in the Questa main menu and select coverage report and HTML
* Tools -> Coverage Report-> HTML
* Select all coverage types and click ok as shown below.



* You can view the Questa Coverage Report summary as shown below





Observations from the Questa Code Coverage and Assertions report.

1. Assertions are failed due to unknown (x) inputs in the testbench.
2. Statement coverage is not 100% since all the possible combinations of test cases are not provided in the test bench.
3. Branch coverage is not 100% since all the branches are not exercised by the testbench.
4. All the signals are not transited from 0 to 1 and 1 to 0.

Problem Statement:

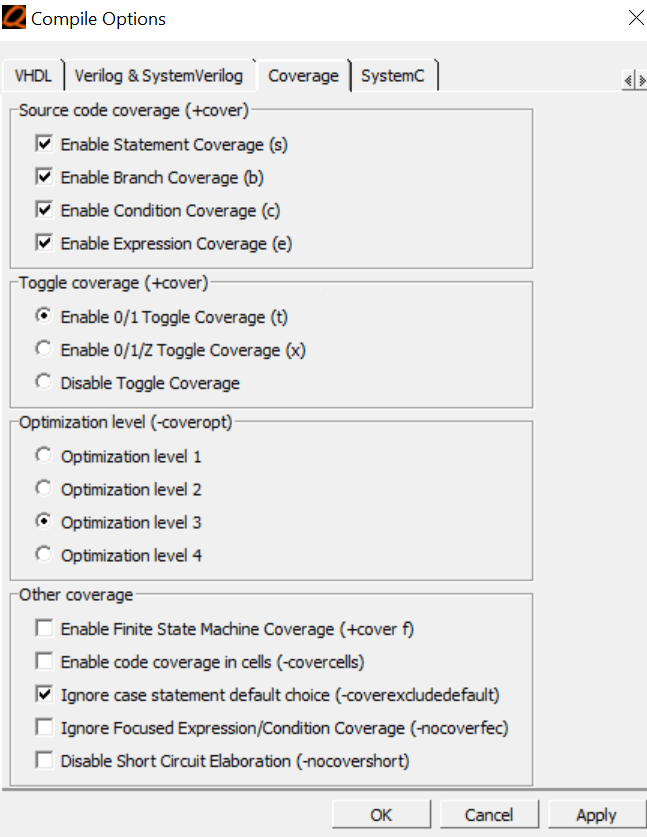
* Modify the given testbench in such a way that the code coverage and the assertion coverage should be 100%
* Goal: Code coverage 100% and Assertion Coverage 100%
* Capture the code coverage summary in your report.

How to achieve 100% branch coverage?

Branch coverage checks all the possible combinations along with the default case. If the default case is not specified in the RTL, you cannot achieve 100% branch coverage.

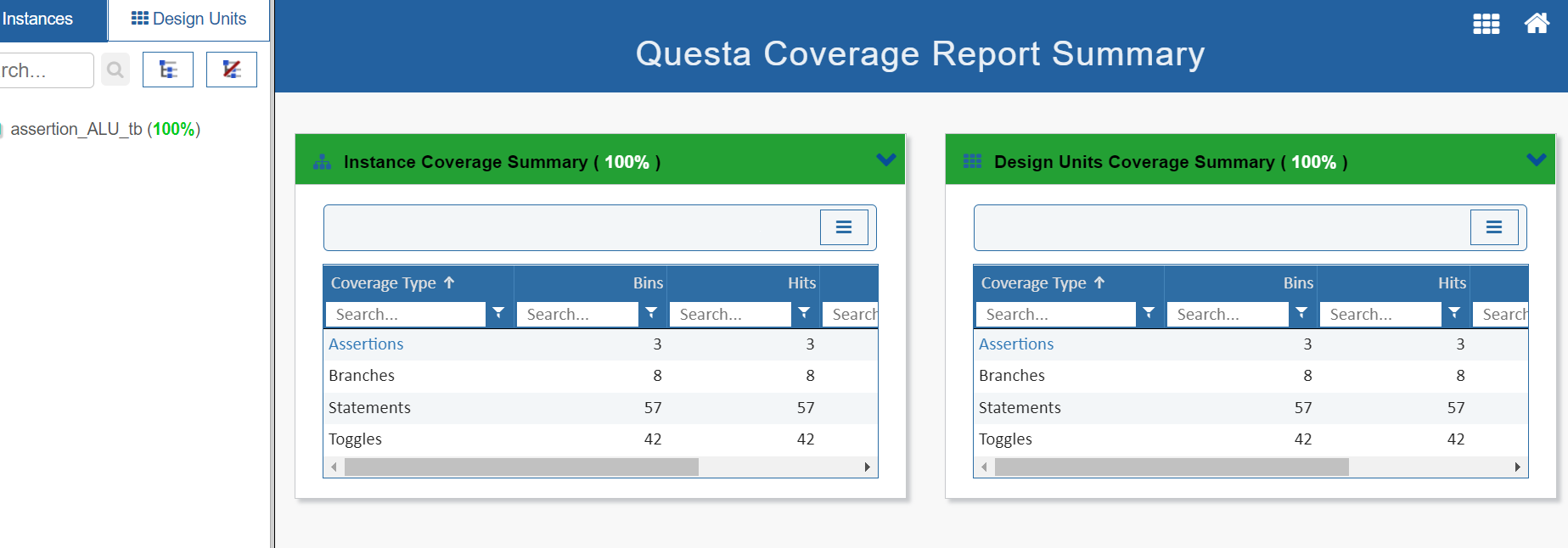
There is an option available in Questa compile options to ignore the default case while performing code coverage as shown below.

Compile-> Compile options and enable the option as shown below and perform the steps again to achieve 100% code coverage



You should get 100% code coverage and 100% assertion coverage after you modify the testbench with suitable testcases and ignoring the default case.

The final coverage should be similar as shown in the image below



END OF LAB EXERCISE

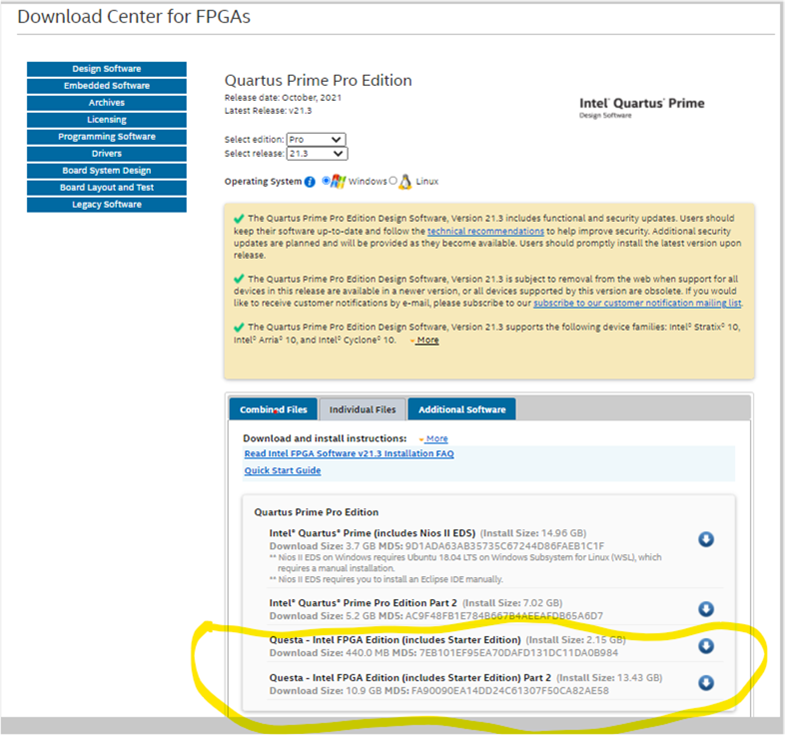
**Appendix**

**Questa - Intel FPGA starter edition – Installation details**

If you don’t already have the Questasim simulator, you will need it to run the labs.

<https://fpgasoftware.intel.com/21.3/?edition=pro&platform=windows>

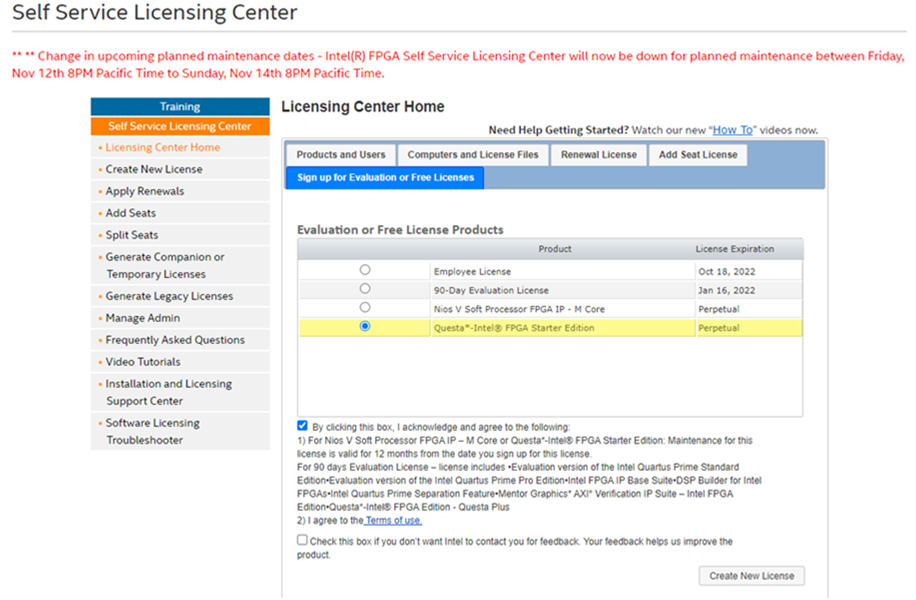
Download the files highlighted:



To obtain a license:

[https://fpgasupport.intel.com/Licensing/license/index.html#](https://fpgasupport.intel.com/Licensing/license/index.html)

Sign-in as an employee.



Click on create new license, follow the instructions on machine name and MAC address. You will receive an email with a new license from [authorization@intel.com](mailto:authorization@intel.com) .

You will need to edit/add environment variables Search 🡪 environment variables. Add as shown below.

