Lab 2: Combinational Circuits

CSC258H1 – Computer Organization

This document describes what you need to prepare and demonstrate for Lab 2. Section 2 briefly describes hierarchical design in Logisim Evolution. Section 3 describes the tasks you must complete *before* your lab session. Section 4 describes the tasks you complete *during* your lab session. The next section describes lab logistics in more detail.

Contents

1. Logistics

Even though you work in pairs during your lab session, you are assessed individually on your Lab Preparation ("pre-lab") and Lab Demonstration ("demo"). All pre-lab exercises are submitted electronically before your lab (see the course website for exact due dates, times, and the submission process). So, **before** each lab, you must read through this document and complete all the pre-lab exercises. During the lab, use your pre-lab designs to help you complete all the required in-lab actions. The more care you put into your pre-lab designs, the faster you will complete your lab.

Before beginning the pre-lab, read Section 2. The Lab Preparation must be completed individually and submitted online by the due date. Follow the steps in Section 3 for the pre-lab. Remember to **download the starter files**.

You must upload *every required file* for your pre-lab submission to be complete. But you do *not* need to include images that are not on the list of required files (even if those images are in your lab report). If you have questions about the submission process, please ask ahead of time. The required files for Lab 2's pre-lab (Section 3) are: `lab2_report.pdf`, `lab2_mux.circ`, `lab2_7seg.circ`, and `test_hex_decoder.txt`.

The Lab Demonstration must be completed during the lab session that you are enrolled in. During a lab demonstration, your TA may ask you to: go through parts of your pre-lab, run and simulate your designs in Logisim, and answer questions related to the lab. You may not receive outside help (e.g., from your partner) when asked a question.

2. Hierarchical Design in Logisim Evolution

Multiplexers are a common building block in digital circuits. In this section, we first describe how to convert a circuit, like a 2-to-1 multiplexer, into a building block. We then describe how to test your circuits. Testing smaller circuits (e.g., 2-to-1 multiplexer) helps with the testing of your larger circuits (e.g., that use multiplexers).

2.1. Subcircuits

In Logisim Evolution, a `.circ` file can be composed of multiple circuits. There is always a top-level circuit. But it is possible to add more circuits called *subcircuits* (or *modules*). From your programming experience, this is analgous to creating a function.

Subcircuits are typically building blocks, like a 2-to-1 multiplexer, used in other subcircuits and/or the top-level module. Each time a subcircuit is used in another circuit, it is an *instance* of that module. Note that, if you label an instance, the name should be unique. We can create large circuits by using multiple modules of the same type. From your programming experience, this is analogous to function reuse.

Logisim Evolution includes a tutorial on hierarchical design. You can find it by launching Logisim Evolution and navigating to `Help > User's Guide`. There, you will find a section (and its subsections) called `Subcircuits`.

2.2. Testing Circuits

It is a good idea to verify the correctness of your circuits and subcircuits. Logisim Evolution provides a method to compare what your circuit actually outputs with what you expect it to output. You can accomplish this by navigating to `Simulate > Test Vector...`.

A test vector is essentially a plain text file containing a truth table. The truth table maps a set of inputs to the expected output. The first line of the file specifies the names of the inputs and outputs - these names must match those found in your circuit. In the lines that follow, you include the values to use for the input(s) and the value you expect at the output(s).

Figure 1 shows a circuit that consists of only an AND gate. Listing 1 shows the corresponding test vector for the circuit. The test vector must be saved as a text file before it can be loaded into Logisim Evolution. Once loaded, the status of each row is given (Figure 2).

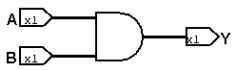


Figure 1: Contrived Circuit

```
# my_test_vector.txt
A B Y
0 0 0
0 1 0
1 0 0
1 1 1
```

Listing 1: Test Vector

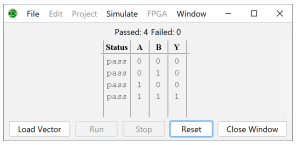


Figure 2: Test Outcomes

3. Lab Preparation

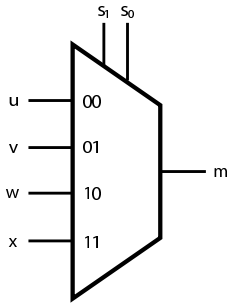
Make sure you have read Section 2 before beginning your pre-lab.

By the end of this pre-lab, you should be able to:

- Reuse subcircuits in other subcircuits (Hierarchical Design).
- Use automated tools in Logisim Evolution to build two-level combinational logic.
- Verify the correctness of circuits in Logisim Evolution using test vectors.

3.1. Part I

In this part, you create a 4-to-1 multiplexer (shown in Figure 3) using three different methods: hierarchical design with logic gates, two-level logic, and hierarchical design with a CMOS gate. Note that the truth table for a 4-to-1 multiplexer in Table 1 is given in a short-hand form. A real truth table would consist of rows enumerating all possible combinations of values of inputs u, v, w, x, in addition to s_0 and s_1, and show the value (0 or 1) of the output m for each row of the truth table.



$s_1 s_0$	m
00	u
01	v
10	w
11	x

Figure 3: Symbol for a 4-to-1 multiplexer Table 1: Truth table for a 4-to-1 multiplexer

Begin by opening the starter file `lab1_mux.circ` in Logisim Evolution. To create a 4-to-1 multiplexer using two-level logic, perform the following steps:

1. Open the sub-circuit `TwoLevelMux41`. Draw a schematic that implements a 4-to-1 multiplexer using only the standard logic gates (e.g., AND, OR, NOT; not TTL).
2. Verify that you have a correct solution with Logisim Evolution. Navigate to `Project` then `Analyze Circuit` and compare the software-generated truth table with what you expect to see.
3. Export the `TwoLevelMux41` schematic as an image and include it in your report. Save your changes and remember to upload your files to MarkUs.

Next, to create a 4-to-1 multiplexer using logic gates and hierarchical design, perform the following steps:[1]

1. Open the sub-circuit `Mux21`. Draw a schematic that implements a 2-to-1 multiplexer using only the standard logic gates (e.g., AND, OR, NOT; not TTL).
2. Open the sub-circuit `HierarchicalMux41`. Create and connect multiple instances of the `Mux21` subcircuit to impelement a 4-to-1 multiplexer.
3. Verify that you have a correct solution with Logisim Evolution. Navigate to `Project` then `Analyze Circuit` and compare the software-generated truth table with what you expect to see.

[1] If you need help, follow the `Subcircuits` tutorial in Logisim Evolution.

4. Export the `HierarchicalMux41` schematic as an image and include it in your report. Save your changes and remember to upload your files to MarkUs.

Finally, to create a 4-to-1 multiplexer using a CMOS gate, perform he following steps:

1. Open the sub-circuit `AOI22`. The schematic is complete and working; *do not change it.* Instead, analyse the circuit to understand how its output relates to its inputs.
2. Open the sub-circuit `InvMux21`. The schematic is complete and working; *do not change it.* Instead, analyse the circuit to understand how its output relates to its inputs.
3. Open the sub-circuit `AOI22Mux41`. Create and connect multiple instances of the `InvMux21` subcircuit to impelement a 4-to-1 multiplexer.
4. Verify that you have a correct solution with Logisim Evolution. Navigate to `Project` then `Analyze Circuit` and compare the software-generated truth table with what you expect to see.
5. Export the `AOI22Mux41` schematic as an image and include it in your report. Save your changes and remember to upload your files to MarkUs.

When you are done, use the `MuxChecker` circuit to see whether all three of your designs behave the same way. In your lab report, answer the following questions:

1. Assume each logic gate is made following the CMOS process. How many transistors are required for each of your 4-to-1 multiplexer designs? Show your work by identifying how many transistors are necessary for each gate and sub-circuit used.

3.2. Part II

In this part, you design a decoder for a 7-segment display (Figure 4). As the name implies, the display has seven segments (lines) that can be ON or OFF. The 7-segment display shown in Figure 4 has a number associated with each segment. So we can refer to a segment by its number. For example, the top-most horizontal segment is S_0, which we can turn ON or OFF by driving pin 0 (third dot from the left at the very top of the display).

The segments are arranged in a way such that you can visualize the numbers 1 to F in hexadecimal. See Table 2 for a mapping of decoder inputs to the characters that need to be displayed. The decoder you are designing takes, as input, 4 bits and produces a 7-bit output that controls the LEDs in the 7-segment display. To design the decoder, you need to identify when a segment in the display is on or off.

Since there are seven segments, there are seven Boolean functions we need to derive, one for each segment. Then, we should use Karnaugh maps to ensure these Boolean functions are minimal. Finally, we would implement each Boolean function as its own

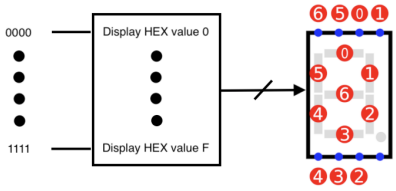


Figure 4: A decoder driving a 7-segment display

$D_{3:0}$	Character
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	b
1100	c
1101	d
1110	E
1111	F

Table 2: The desired behaviour of the decoder

subcircuit. Fortunately, Logisim Evolution can do most of this work for us. However, we do need to provide Logisim Evolution with the correct truth table.

CAUTION

Example 2.10 in the textbook is a very useful starting point. But be careful, the textbook example is only concerned with visualizing the characters 0 to 9. This lab is also asking you to visualize the outputs associated with A, b, c, d, E, F (Table 2).

Perform the following steps:

1. In your lab report, derive seven truth tables, one for each segment of the 7-segment decoder. Another way to ask this question is: which segments should be on (and which should be off) for a given character?
2. Open the starter file `lab2_7seg.circ`. You will see seven subcircuits named `Hex0`, `Hex1`, ... `Hex6`.
3. Open the subcircuit `Hex0` and navigate to `Project` then `Analyse Circuit`. Select the `Table` tab and notice that Logisim Evolution has enumerated all input combinations. But the outputs are all unknown.
   a) Type in the correct output (0 or 1) for each input combination based on your truth table for Segment 0.

b) Click on `Build Circuit`. Ensure that the `Destination Project` is `lab2_7seg` and `Circuit Name` is the subcircuit you are working on (e.g., `Hex0`).

c) Click `Ok` then select `Yes` to the prompt `Are you sure you want to replace the circuit...`.

d) Select the `Minimized` tab to inspect the Karnaugh Map. Make sure you understand what Logisim Evolution has done.

e) Select the `Table` tab to inspect the Boolean equation and include it in your lab report.

f) Export the schematic as an image and include it in your report.

4. Repeat the previous step for the remaining six segments' subcircuits.

5. Inspect the subcircuit `HexDecoder`. The schematic is complete; *do not change it.* But it will only work after you have implemented each of the segment subcircuits correctly. Use Logisim Evolution to inspect its truth table and compare it to the one from your lab report.

When you are done, use the `HexChecker` circuit to see whether your decoder behaves as expected. After you are certain that your decoder is correct, create a test vector for the `HexDecoder` subcircuit called `test_hex_decoder.txt`. Ensure that your `HexDecoder` passes your own test vector. Save your changes and remember to upload your files to MarkUs.

4. Lab Demonstration

By the end of this lab demonstration, you should be able to:

- Describe how you might build more complex circuits using hierarchical design
- Translate functional specifications into optimized Boolean equations and their corresponding circuits
- Synthesise your Logisim Evolution circuits onto the DE1-SoC board and observe their behaviour through the board's input and output

4.1. Part I

In Logisim Evolution, design an 8-to-1 multiplexer using instances of 2-to-1 and/or 4-to-1 multiplexers. You may use Logisim Evolution's built-in multiplexer if you wish (you can find it under `Plexers`), but only for the 2-to-1 and 4-to-1 configurations. When you are done, demonstrate your working design to your TA.

4.2. Part II

Synthesise the `HexDecoder` sub-circuit (Section 3.2) and download it onto the DE1-SoC board (see Appendix A). You should map the input pins (i.e., `D0, D1, D2, D3`) to switches. You should map the output pins to the corresponding segments of the board's seven-segment display. When you are done, demonstrate your working board to your TA.

Note: With our reference solution, the `Synthesize & Download` step takes many minutes. So make sure your design works correctly in Logisim Evolution before initiating this step. If you are curious whether things are actually working, see Appendix B for the output that was generated by Logisim Evolution when we synthesized our `HexDecoder` solution.

A. Instructions for synthesis

This section enumerates the process for synthesising a Logisim Evolution circuit to the FPGA board found in the labs.

CAUTION

Make sure you are using a lab computer (i.e., a desktop computer in BA3135, BA3145, BA3155, or BA3165) and have opened Logisim Evolution 3.8.0. Also make sure you have `DE1_SOC_7seg.xml` downloaded on the lab computer (see your starter files). Finally, make sure the DE1-SoC board is turned on.

1. Navigate to `FPGA > Synthesize & Download`. A small window should open (Figure 5).
2. In the `Target board` panel of the window, select `Other` from the pull-down menu. An open file dialogue should open.
3. Navigate to, and `Open`, the `DE1_SOC_7seg.xml` file that you downloaded (e.g., see your starter files).
4. Click on the `Settings` button just above the `Target board` panel. A Preferences window should open (Figure 6). You should see `DE1_SOC_7seg` under `External FPGA Board list:`.
5. Click on the `Browse` button beside `Workspace location`. An open file dialogue should open.
6. Create a new subdirectory in your home directory and select it as your workplace location (e.g., `W:\csc258`).

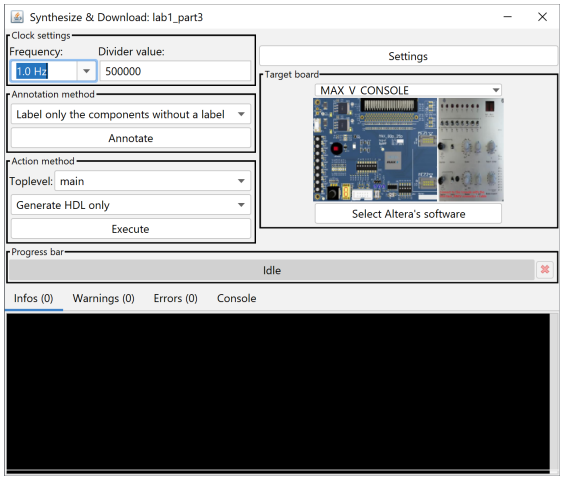


Figure 5: The 'Synthesize & Download' window

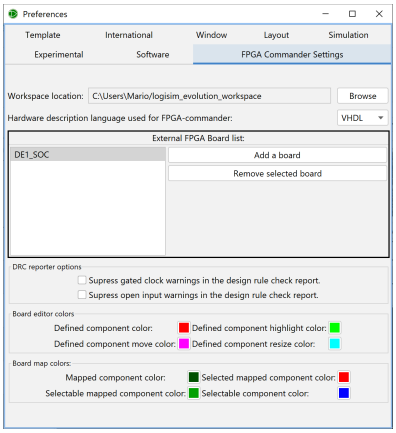


Figure 6: The Preferences: FPGA Commander tab

7. Navigate to the `Software` tab (Figure 7). Click on `Browse` next to the `Altera/Intel Quartus toolpath`. An open folder dialogue should open.
8. Navigate to, and `Open`, the `C:\DESL\Quartus18\quartus\bin64` folder.
9. Close the `Preference` window. Everything should now be configured.
10. In the `Action method` panel (Figure 5), select the appropriate Toplevel circuit. From the pull-down menu, make sure that `Synthesize & Download` (not `Generate HDL only`) is selected.

    Note: If you don't see this option in the pull-down menu, it is because the Altera software has not been correctly setup. In this situation, you should see a button

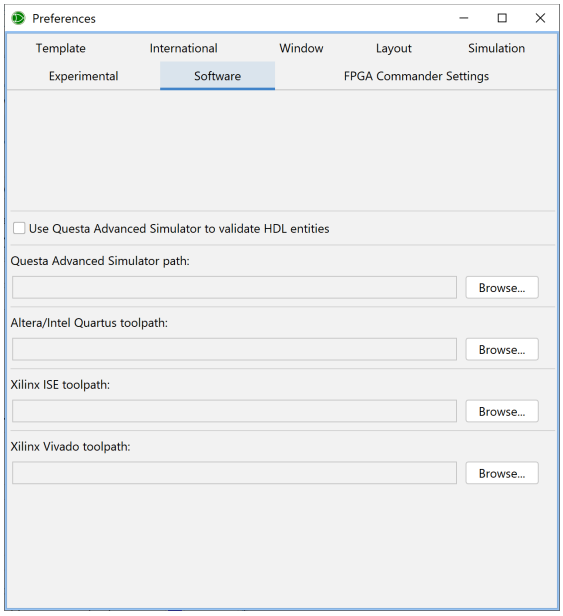


Figure 7: The Preferences: Software tab

for selecting the Altera software underneath the image of the board. Click on this button and open the `C:\DESL\Quartus18\quartus\bin64` folder.

11. Click on `Execute` under `Action method`. A window should open (Figure 8).

    With this version of Logisim Evolution, your circuit's inputs can be mapped to the buttons and switches on the physical DE1-SoC board. Similarly, your circuit's outputs can be mapped to the LEDs, or individual segments on the 7 segment displays.

    Note: For the circuit you are synthesising, you must use input pins and output pins in order to be able to map them.

12. Map the pins in the `Unmapped List:`. First, select one of the pins. The components that this pin can be mapped to are highlighted in translucent red on the image of the board. Click on one to map the pin. Repeat until all pins have been mapped to a physical component on the DE1-SoC board.

13. Click on `Done` to start synthesising your circuit. This **will** take time.

14. Select `Yes, download` when presented with the option.

15. Test your circuit by toggling switches and observing the LEDs and 7 segment displays.

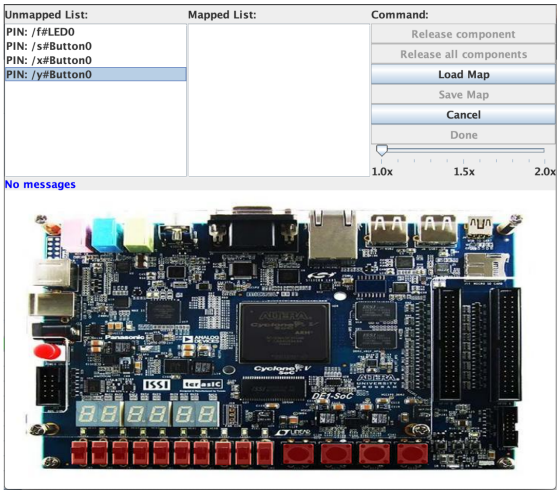


Figure 8: The board mapping window

B. Synthesize & Download Log for HexDecoder

```
==>
==> Creating Altera project files
==>
Info: ******************************************************************
Info: Running Quartus Prime Shell
    Info: Version 18.0.0 Build 614 04/24/2018 SJ Standard Edition
    Info: Copyright (C) 2018  Intel Corporation. All rights reserved.
    Info: Your use of Intel Corporation's design tools, logic functions
    Info: and other software and tools, and its AMPP partner logic
    Info: functions, and any output files from any of the foregoing
    Info: (including device programming or simulation files), and any
    Info: associated documentation or information are expressly subject
    Info: to the terms and conditions of the Intel Program License
    Info: Subscription Agreement, the Intel Quartus Prime License Agreement,
    Info: the Intel FPGA IP License Agreement, or other applicable license
    Info: agreement, including, without limitation, that your use is for
    Info: the sole purpose of programming logic devices manufactured by
    Info: Intel and sold by Intel or its authorized distributors.  Please
    Info: refer to the applicable agreement for further details.
    Info: Processing started: Thu Sep 14 16:11:25 2023
Info: Command: quartus_sh -t ..\scripts\AlteraDownload.tcl
Info (23030): Evaluation of Tcl script ..\scripts\AlteraDownload.tcl was successful
Info: Quartus Prime Shell was successful. 0 errors, 0 warnings
    Info: Peak virtual memory: 4628 megabytes
    Info: Processing ended: Thu Sep 14 16:11:26 2023
    Info: Elapsed time: 00:00:01
    Info: Total CPU time (on all processors): 00:00:00

==>
==> Optimizing Altera project
==>
Info: ******************************************************************
Info: Running Quartus Prime Analysis & Synthesis
    Info: Version 18.0.0 Build 614 04/24/2018 SJ Standard Edition
    Info: Copyright (C) 2018  Intel Corporation. All rights reserved.
```

```
    Info: Your use of Intel Corporation's design tools, logic functions
    Info: and other software and tools, and its AMPP partner logic
    Info: functions, and any output files from any of the foregoing
    Info: (including device programming or simulation files), and any
    Info: associated documentation or information are expressly subject
    Info: to the terms and conditions of the Intel Program License
    Info: Subscription Agreement, the Intel Quartus Prime License Agreement,
    Info: the Intel FPGA IP License Agreement, or other applicable license
    Info: agreement, including, without limitation, that your use is for
    Info: the sole purpose of programming logic devices manufactured by
    Info: Intel and sold by Intel or its authorized distributors.  Please
    Info: refer to the applicable agreement for further details.
    Info: Processing started: Thu Sep 14 16:11:27 2023
Info: Command: quartus_map logisimTopLevelShell --optimize=area
Warning (18236): Number of processors has not been specified which may cause overloading on shared machines.  Set the gl

MANY LINES OMITTED

Info (21057): Implemented 18 device resources after synthesis - the final resource count might be different
    Info (21058): Implemented 4 input pins
    Info (21059): Implemented 7 output pins
    Info (21061): Implemented 7 logic cells
Info: Quartus Prime Analysis & Synthesis was successful. 0 errors, 1 warning
    Info: Peak virtual memory: 4871 megabytes
    Info: Processing ended: Thu Sep 14 16:14:09 2023
    Info: Elapsed time: 00:02:42
    Info: Total CPU time (on all processors): 00:00:10

==>
==> Altera synthesizing, P&R, and generating bit file; this may take a while
==>
Info: *******************************************************************
Info: Running Quartus Prime Shell
    Info: Version 18.0.0 Build 614 04/24/2018 SJ Standard Edition
    Info: Copyright (C) 2018  Intel Corporation. All rights reserved.
    Info: Your use of Intel Corporation's design tools, logic functions
    Info: and other software and tools, and its AMPP partner logic
    Info: functions, and any output files from any of the foregoing
    Info: (including device programming or simulation files), and any
    Info: associated documentation or information are expressly subject
    Info: to the terms and conditions of the Intel Program License
    Info: Subscription Agreement, the Intel Quartus Prime License Agreement,
    Info: the Intel FPGA IP License Agreement, or other applicable license
    Info: agreement, including, without limitation, that your use is for
    Info: the sole purpose of programming logic devices manufactured by
    Info: Intel and sold by Intel or its authorized distributors.  Please
    Info: refer to the applicable agreement for further details.
    Info: Processing started: Thu Sep 14 16:14:11 2023
Info: Command: quartus_sh --flow compile logisimTopLevelShell
Info: Quartus(args): compile logisimTopLevelShell
Info: Project Name = W:/csc258/lab2_7seg_solution/HexDecoder/sandbox/logisimTopLevelShell
Info: Revision Name = logisimTopLevelShell
Info: *******************************************************************
Info: Running Quartus Prime Analysis & Synthesis
    Info: Version 18.0.0 Build 614 04/24/2018 SJ Standard Edition
    Info: Processing started: Thu Sep 14 16:14:13 2023
Info: Command: quartus_map --read_settings_files=on --write_settings_files=off logisimTopLevelShell -c logisimTopLevelSh
Warning (18236): Number of processors has not been specified which may cause overloading on shared machines.  Set the gl

MANY LINES OMITTED

Info: Quartus Prime Analysis & Synthesis was successful. 0 errors, 1 warning
```

```
    Info: Peak virtual memory: 4868 megabytes
    Info: Processing ended: Thu Sep 14 16:15:30 2023
    Info: Elapsed time: 00:01:17
    Info: Total CPU time (on all processors): 00:00:20
Info: *******************************************************************
Info: Running Quartus Prime Fitter
    Info: Version 18.0.0 Build 614 04/24/2018 SJ Standard Edition
    Info: Processing started: Thu Sep 14 16:15:33 2023
Info: Command: quartus_fit --read_settings_files=off --write_settings_files=off logisimTopLevelShell -c logisimTopLevelS
Info: qfit2_default_script.tcl version: #1

MANY LINES OMITTED

Info: Quartus Prime Fitter was successful. 0 errors, 5 warnings
    Info: Peak virtual memory: 6745 megabytes
    Info: Processing ended: Thu Sep 14 16:16:11 2023
    Info: Elapsed time: 00:00:38
    Info: Total CPU time (on all processors): 00:00:57
Info: *******************************************************************
Info: Running Quartus Prime Assembler
    Info: Version 18.0.0 Build 614 04/24/2018 SJ Standard Edition
    Info: Processing started: Thu Sep 14 16:16:18 2023
Info: Command: quartus_asm --read_settings_files=off --write_settings_files=off logisimTopLevelShell -c logisimTopLevelS
Warning (18236): Number of processors has not been specified which may cause overloading on shared machines.  Set the gl
Info (115030): Assembler is generating device programming files
Info: Quartus Prime Assembler was successful. 0 errors, 1 warning
    Info: Peak virtual memory: 4836 megabytes
    Info: Processing ended: Thu Sep 14 16:16:32 2023
    Info: Elapsed time: 00:00:14
    Info: Total CPU time (on all processors): 00:00:06
Info (293026): Skipped module Power Analyzer due to the assignment FLOW_ENABLE_POWER_ANALYZER
Info: *******************************************************************
Info: Running Quartus Prime Timing Analyzer
    Info: Version 18.0.0 Build 614 04/24/2018 SJ Standard Edition
    Info: Processing started: Thu Sep 14 16:16:33 2023
Info: Command: quartus_sta logisimTopLevelShell -c logisimTopLevelShell
Info: qsta_default_script.tcl version: #2
Warning (18236): Number of processors has not been specified which may cause overloading on shared machines.  Set the gl

MANY LINES OMITTED

Info: Quartus Prime Timing Analyzer was successful. 0 errors, 6 warnings
    Info: Peak virtual memory: 5130 megabytes
    Info: Processing ended: Thu Sep 14 16:16:59 2023
    Info: Elapsed time: 00:00:26
    Info: Total CPU time (on all processors): 00:00:05
Info (293000): Quartus Prime Full Compilation was successful. 0 errors, 13 warnings
Info (23030): Evaluation of Tcl script c:/desl/quartus18/quartus/common/tcl/internal/qsh_flow.tcl was successful
Info: Quartus Prime Shell was successful. 0 errors, 13 warnings
    Info: Peak virtual memory: 4641 megabytes
    Info: Processing ended: Thu Sep 14 16:17:01 2023
    Info: Elapsed time: 00:02:50
    Info: Total CPU time (on all processors): 00:00:03

===
===> Detected connected boards
===
1) DE-SoC [USB-1]
Info: *******************************************************************
Info: Running Quartus Prime Programmer
    Info: Version 18.0.0 Build 614 04/24/2018 SJ Standard Edition
    Info: Copyright (C) 2018  Intel Corporation. All rights reserved.
```

```
    Info: Your use of Intel Corporation's design tools, logic functions
    Info: and other software and tools, and its AMPP partner logic
    Info: functions, and any output files from any of the foregoing
    Info: (including device programming or simulation files), and any
    Info: associated documentation or information are expressly subject
    Info: to the terms and conditions of the Intel Program License
    Info: Subscription Agreement, the Intel Quartus Prime License Agreement,
    Info: the Intel FPGA IP License Agreement, or other applicable license
    Info: agreement, including, without limitation, that your use is for
    Info: the sole purpose of programming logic devices manufactured by
    Info: Intel and sold by Intel or its authorized distributors.  Please
    Info: refer to the applicable agreement for further details.
    Info: Processing started: Thu Sep 14 16:17:22 2023
Info: Command: quartus_pgm --list
Info: Quartus Prime Programmer was successful. 0 errors, 0 warnings
    Info: Peak virtual memory: 4391 megabytes
    Info: Processing ended: Thu Sep 14 16:17:23 2023
    Info: Elapsed time: 00:00:01
    Info: Total CPU time (on all processors): 00:00:00

==>
==> Downloading design to the board.
==>
Info: *******************************************************************
Info: Running Quartus Prime Programmer
    Info: Version 18.0.0 Build 614 04/24/2018 SJ Standard Edition
    Info: Copyright (C) 2018  Intel Corporation. All rights reserved.
    Info: Your use of Intel Corporation's design tools, logic functions
    Info: and other software and tools, and its AMPP partner logic
    Info: functions, and any output files from any of the foregoing
    Info: (including device programming or simulation files), and any
    Info: associated documentation or information are expressly subject
    Info: to the terms and conditions of the Intel Program License
    Info: Subscription Agreement, the Intel Quartus Prime License Agreement,
    Info: the Intel FPGA IP License Agreement, or other applicable license
    Info: agreement, including, without limitation, that your use is for
    Info: the sole purpose of programming logic devices manufactured by
    Info: Intel and sold by Intel or its authorized distributors.  Please
    Info: refer to the applicable agreement for further details.
    Info: Processing started: Thu Sep 14 16:17:23 2023
Info: Command: quartus_pgm -c "DE-SoC [USB-1]" -m jtag -o P;logisimTopLevelShell.sof@2
Info (213045): Using programming cable "DE-SoC [USB-1]"
Info (213011): Using programming file logisimTopLevelShell.sof with checksum 0x00AF787E for device 5CSEMA5F31@2
Info (209060): Started Programmer operation at Thu Sep 14 16:17:25 2023
Info (209016): Configuring device index 2
Info (209017): Device 2 contains JTAG ID code 0x02D120DD
Info (209007): Configuration succeeded -- 1 device(s) configured
Info (209011): Successfully performed operation(s)
Info (209061): Ended Programmer operation at Thu Sep 14 16:17:28 2023
Info: Quartus Prime Programmer was successful. 0 errors, 0 warnings
    Info: Peak virtual memory: 4435 megabytes
    Info: Processing ended: Thu Sep 14 16:17:28 2023
    Info: Elapsed time: 00:00:05
    Info: Total CPU time (on all processors): 00:00:01
```

