

**Department of Electrical & Electronic Engineering  
Imperial College London**

**EE2 Circuits & Systems**

**Lab 3 – Introduction to DE10-Lite and Quartus Prime Lite**

**Objectives**

By the end of this experiment, you should have achieved:

- Launching Quartus Prime Lite v20.1 on one of the Lab's PCs;
- Create a directory structure for this and subsequent Labs;
- Create a new project in Quartus and complete a basic 7-segment LED display decoder design using Quartus and SystemVerilog from start to finish;
- Program the Max 10 FPGA chip on the DE10-Lite board with your design;
- Explore and test your decoder design;
- Analyse the propagation delay of the decoder circuit

**Running Quartus Prime Lite**

Like the Labs in the first half of the Autumn term, you are expected to conduct the remaining Lab Sessions in Level 1 Lab. You will be using Intel/Altera software known as **Quartus Prime Lite** which are already installed locally on all the PCs.

For those who wish to install Quartus Prime Lite on their own personal laptop, **do so at your own "risk"**. Beware that this may take effort and time on your part.

If you do want to install Quartus on your own machine, you will need to have at least 30GB free disk space and at least 4GB of RAM (preferably 8GB of RAM) on your laptop.

**Intel processor based PC running Windows OS**

This method is the easiest to have Quartus Prime Lite installed. Quartus is one of the software available on Software Hub. Search for "**Quartus**" among all the Software Hub packages. You will find two are available. Select the package on the RIGHT.

Click "Launch" and Quartus Prime Lite v20.1 will be downloaded and onto your PC. Run Quartus at least once while you are on College wifi (so that the license is verified). Thereafter, Quartus will be cached locally on your PC and the license will be valid for another 30 days. The software package is large – you need to download a few gigabytes! Therefore, you are strongly advised to do this while on campus for the first time. Finally, you need to install the USB Blaster driver which you can download from the course webpage.

**Apple MacBook with Intel processor**

It is possible to run a virtual machine (e.g. Virtual Box), the Ubuntu (Linux) Operating System, and the Linux version of Quartus Prime Lite. Do this **ONLY** if you are familiar

with VM and Linux operating systems. Also note that Quartus Prime Lite was designed to run on Windows or Linux operating systems. There may be issues running Quartus on MacBooks under VM. If you have an Intel MacBook and want to use Quartus on this, please talk to Peter Cheung to obtain further instructions.

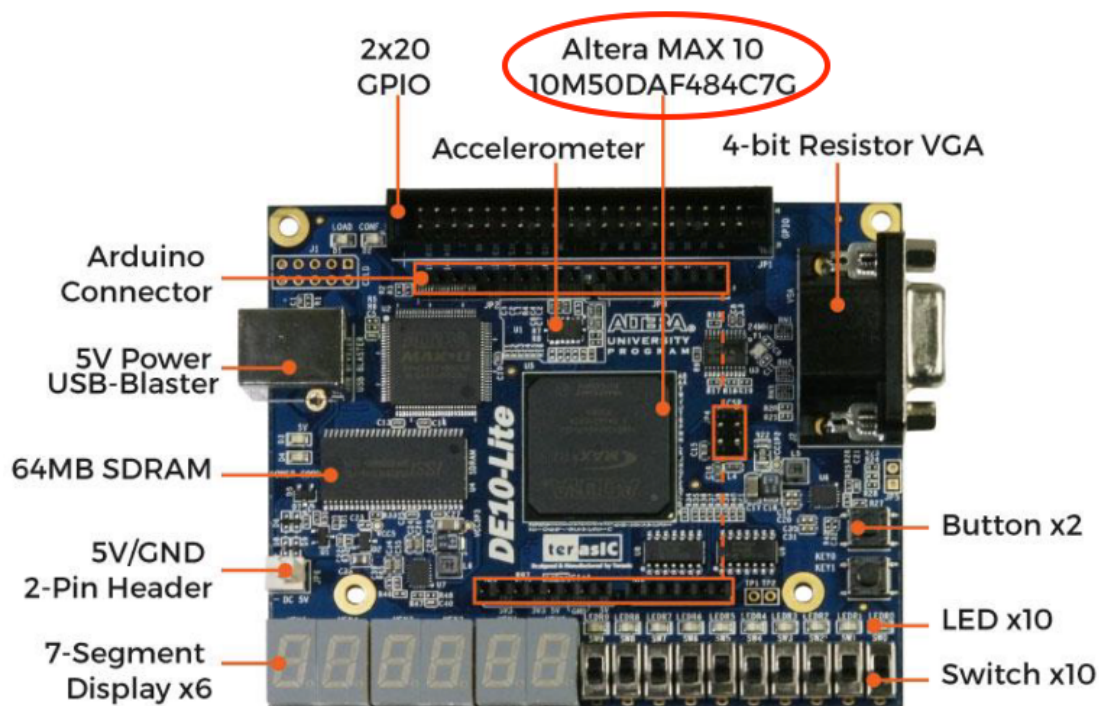
### Apple MacBook with M1 or M2 processor

You cannot run Quartus on MacBook with Apple silicon M1 or M2. You can only use the PC in the Lab. There are a few Laptops that you may be able to borrow overnight with your ID card from Level 1 stores.

### The DE10-Lite FPGA Board

Borrow a DE10-Lite FPGA board from level 1 stores with your ID card. You can keep this until the end of the Autumn term. Connect the DE10 board to your laptop using the USB cable provided. You should see the board LED displays cycling through all digits, showing that the board is working properly. The diagram below shows the features provided on this board.

Note that the FPGA chip is **10M50DAF484C7G**. Make sure that you specify THIS DEVICE exactly in your design.

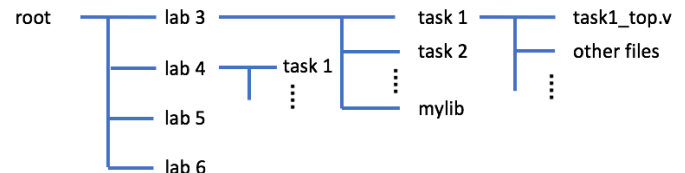


## Task 0: Preparation

In task 1, you will use four slide switches on the right (SW3 to SW0) on the DE10 board as inputs and display the 4-bit binary number as a hexadecimal digit on the right-most 7-segment display (HEX0). In task 0, you will check that everything is working properly.

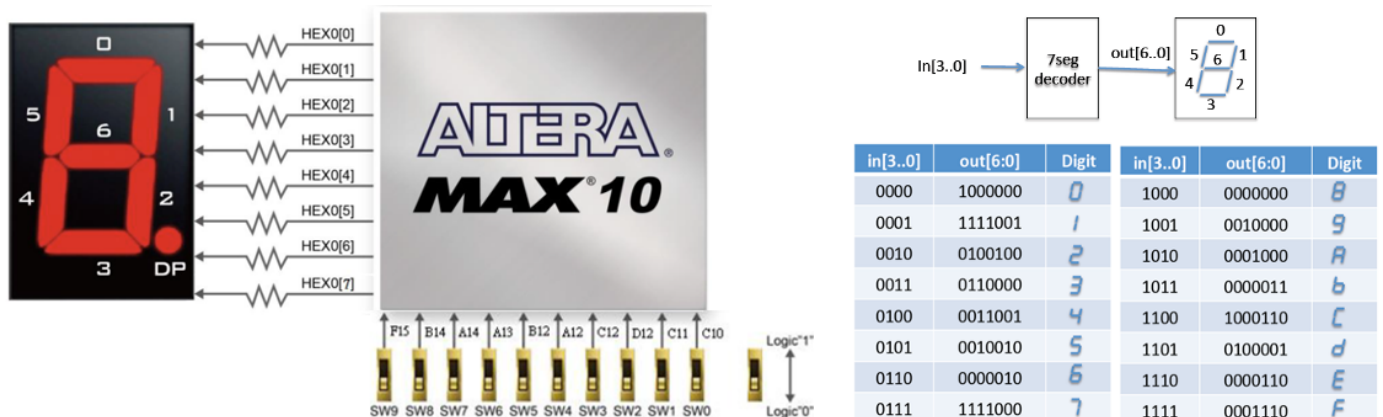
### Step 1: Creating a good directory structure

Before you start carrying out any design for this Lab, it would be very helpful if you first create a directory structure for Lab 3 to 6 on your "h: drive". Shown on the right is a possible directory structure that you may choose to create. Each folder is empty for now, but as you progress through the four Lab Sessions, you will be creating each design in each of the folders.



### Step 2: See what you are aiming for

Go to the course webpage and download a copy of the solution for Exercise 1: "lab3task1\_sol.sof" to your folder for Lab3 (or wherever that is). Make sure that your DE10 board is plugged in and running.



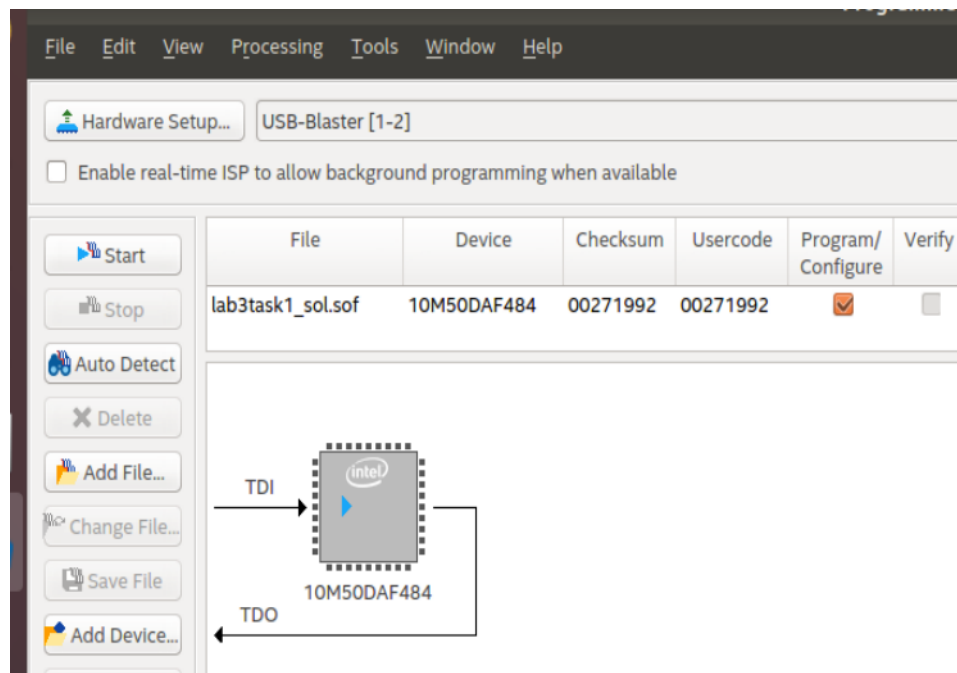
### Step 3: Setting up programming hardware

Run Quartus software on your lab PC. Click command: **Tools > Programmer**. In the popup window, click: **Hardware Setup ....** You should see something like the diagram on the right. Then select: **USB Blaster**. This is to tell Quartus software that you are using the DE10 interface to program (or blast) the FPGA.

If you do not see USB-Blaster under Hardware Setup although the DE10-Lite is plugged in properly, it is most likely that the Device Driver had not been installed or loaded properly. In which case, ask one of the GTAs for help.

#### Step 4: Blasting the FPGA

Next click the **AddFile** button. Navigate to the folder containing the **lab3task1\_sol.sof** file. Select this. You should see a display like this:



Note that the device indicated here is the one on the DE10 board: 10M50DAF484. Click the **Start** button.

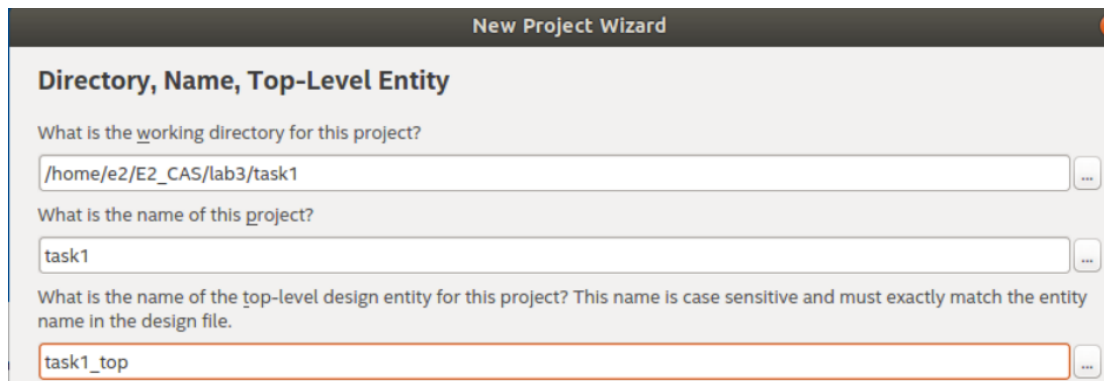
The **lab3task1\_sol.sof** file contains the solution to Task 1. It has the **bit-stream** to configure (or programme/blast) the FPGA Max10 chip. Once the bit-stream is successfully sent to the FPGA chip, the task1 design will take over the function of the chip. You should be able to change the least significant four switches and see a hexadecimal number displayed on rightmost 7-segment display. Now you are ready to create this design from scratch.

The remaining part of Lab 3 will be teaching you how to create YOUR OWN version of this design and blast it onto the FPGA.

## Task 1: The Design Flow – 7 Segment LED Display

### Step 5: Create the project “task1”

- Create in your home directory the folder **E2\_CAS/lab3/task1**.
- Click **file>New Project Wizard**, complete the form. Use **task1** as the project name and **task1\_top** as top-level design name.
- Click **Finish**.



### Step 6: Device Assignment

Click **Assignments -> Device**, and select the Max 10 chip used in DE10, which is: **10M50DAF484C7G**.

Interpretation of the device code: **10M** is Max10 family. **50** is the size of the device of around 50,000 logic elements. **484** is the number of pins. **C7** is the speed grade.

### Step 7: Creating the SystemVerilog specification

- In Quartus, create a design file for the decoder module in SystemVerilog HDL as **hexto7seg.sv** using:

**File > New ....** and select SystemVerilog HDL from the list.

- Type the source file as shown here. Make sure that you pay attention to the syntax of SystemVerilog. Save your file.
- At this stage, you check the syntax of your code by clicking: **Process > Analyze current file**. You should get into a habit of ALWAYS perform this step to make sure that the new SystemVerilog module you created is error free. It will save you a lot of time later.
- Click: **Project > Add Current file to Project** to include this module in your design.

```
1  module hexto7seg  (
2      output logic [6:0] out,    // low-active output
3      input logic [3:0] in      // 4-bit binary input
4  );
5
6      always_comb
7      case (in)
8          4'h0: out = 7'b1000000;
9          4'h1: out = 7'b1111001;
10         4'h2: out = 7'b0100100;    // -- 0 ---
11         4'h3: out = 7'b0110000;    // |      |
12         4'h4: out = 7'b0011001;    // |      |
13         4'h5: out = 7'b0010010;    // -- 6 ---
14         4'h6: out = 7'b0000010;    // |      |
15         4'h7: out = 7'b1111000;    // 4      2
16         4'h8: out = 7'b0000000;    // |      |
17         4'h9: out = 7'b0011000;    // -- 3 ---
18         4'ha: out = 7'b0001000;
19         4'hb: out = 7'b0000011;
20         4'hc: out = 7'b1000110;
21         4'hd: out = 7'b0100001;
22         4'he: out = 7'b0000110;
23         4'hf: out = 7'b0001110;
24         default: out = 7'b0000000; // default all
25     endcase
26 endmodule
```

## Step 8: Create Top-Level Specification in SystemVerilog

- We need to create a top-level (at chip level) design that make sure of the decoder module. Create the file “task1\_top.sv” as shown below.

```
1  //-----
2  // Module name: ex3_top
3  // Function: Top level module for Lab 3 Task 1
4  //      .... to dispaly 4-bit value a 7-seg display
5  // Creator: Peter Cheung
6  // Version: 3.0
7  // Date: 9 Nov 2022
8  //-----
9  module task1_top (
10     input logic [3:0] SW, // declare input/output ports
11     output logic [6:0] HEX0
12 );
13     hexto7seg SEG0 (.out(HEX0), .in(SW[3:0]));
14 endmodule
```

- Specify that this is file as our **top-level design** with:  
**Project > Set as Top-level Entity ....**
- Verify that everything works properly with:  
**Process > Start > Start analysis & elaboration.**

Make sure that there is **no error**.

(Warnings often capture potential errors. However, the Quartus system generates many warnings, and nearly all of which are not important. Once you have gain confidence in the system, you may start ignoring the warning, but never ignore any error.)

You will save a lot of time if you ALWAYS use these two steps: analyze, and analysis & elaboration, and ensure that ALL errors are dealt with (and warning understood).

Note: Every time you create a new entity or module as part of your design, you must include the file in the project.

- Click: **Project > Add Current Files to Project ....**,



## Step 9: Pin assignment – the hard way

You need to associate your design with the **physical pins** of the Max 10 FPGA on the DE10 board. We will now only assign two of 11 pins used in our design.

- Click **Assignment > Pin Planner** and a new window with the chip package diagram. You should also see the top-level input/output ports shown as a list.

Signal Name	Pin Location
HEX0[6]	PIN_C17
HEX0[5]	PIN_D17
HEX0[4]	PIN_E16
HEX0[3]	PIN_C16
HEX0[2]	PIN_C15
HEX0[1]	PIN_E15
HEX0[0]	PIN_C14
SW[3]	PIN_C12
SW[2]	PIN_D12
SW[1]	PIN_C11
SW[0]	PIN_C10

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair
HEX0[6]	Output	PIN_C17		B7_NO	3.0-V LVTTL		12mA (default)	2 (default)	
HEX0[5]	Output				2.5 V (default)		12mA (default)	2 (default)	
HEX0[4]	Output				2.5 V (default)		12mA (default)	2 (default)	
HEX0[3]	Output				2.5 V (default)		12mA (default)	2 (default)	
HEX0[2]	Output				2.5 V (default)		12mA (default)	2 (default)	
HEX0[1]	Output				2.5 V (default)		12mA (default)	2 (default)	
HEX0[0]	Output				2.5 V (default)		12mA (default)	2 (default)	
SW[3]	Input	PIN_C12		B7_NO	3.0-V LVTTL		12mA (default)		

- Click on the field shown and select the appropriate values for Location and I/O standard.
- Close the pin assignment window and click: **File > open...** Enter \*.\* in the file name field and select: **task1.qsf** (qsf = **Q**uartus **S**etting **F**ile). Examine its contents. You should see the effect of the manual pin assignment step as highlighted in RED.
- The first line defines the physical pin location of HEX0[6] is PIN\_C17.
- The second line defines the voltage standard used is 3.0V LVTTL.

```

47 set_global_assignment -name ERROR_CHECK_FREQUENCY_DIVISOR 400
48 set_global_assignment -name VERILOG_FILE task1_top.v
49 set_global_assignment -name VERILOG_FILE hex_to_7seg.v
50 set_global_assignment -name PARTITION_NETLIST_TYPE SOURCE -section_id Top
51 set_global_assignment -name PARTITION_FITTER_PRESERVATION_LEVEL PLACEMENT_AND_ROUTING -section_id Top
52 set_global_assignment -name PARTITION_COLOR 16764057 -section_id Top
53 set_location_assignment PIN_C17 -to HEX0[6]
54 set_instance_assignment -name IO_STANDARD "3.0-V LVTTL" -to HEX0[6]
55 set_global_assignment -name MIN_CORE_JUNCTION_TEMP 0
56 set_global_assignment -name MAX_CORE_JUNCTION_TEMP 85
57 set_location_assignment PIN_C12 -to SW[3]
58 set_instance_assignment -name IO_STANDARD "3.0-V LVTTL" -to SW[3]
59 set_instance_assignment -name PARTITION_HIERARCHY root_partition -to | -section_id Top

```

## Step 10: Pin Assignment – the easy way

- Manual pin assignment is tedious and prone to errors. A much better way to perform pin assignment is to **insert a text file** with the necessary information directly into the **.qsf** file.
- Delete the four lines highlight above which was created through the manual pin assignment in Step 9.

- Download from the course webpage: **pin\_assignment.txt** to the task1 folder.
- Click: Edit > Insert File ... and insert **pin\_assignment.txt** at the end of the file.

ALL the pins used on the DE10 are assigned here. However, unused pins are ignored.

### Step 11: Compile the design & Programming the FPGA

- Click Process > Start Compilation. This will perform all the steps of compilation, placement, routing, fitting etc. and produce a bit-stream file (.sof) ready to blast onto the FPGA.
- Examine the Compilation Report and you should see a Flow Summary similar to the one shown here.

Table of Contents	Flow Summary																																
<ul style="list-style-type: none"> <li>Flow Summary</li> <li>Flow Settings</li> <li>Flow Non-Default Global Settings</li> <li>Flow Elapsed Time</li> <li>Flow OS Summary</li> <li>Flow Log</li> <li>Analysis &amp; Synthesis</li> <li>Fitter</li> <li>Flow Messages</li> <li>Flow Suppressed Messages</li> <li>Assembler</li> <li>Timing Analyzer</li> </ul>	<div>&lt;&lt;Filter&gt;&gt;</div> <table> <tr> <td>Flow Status</td><td>Successful - Sat Oct 31 20:53:08 2020</td></tr> <tr> <td>Quartus Prime Version</td><td>20.1.0 Build 711 06/05/2020 SJ Lite Edition</td></tr> <tr> <td>Revision Name</td><td>task1_top</td></tr> <tr> <td>Top-level Entity Name</td><td>task1_top</td></tr> <tr> <td>Family</td><td>MAX 10</td></tr> <tr> <td>Device</td><td>10M50DAF484C7G</td></tr> <tr> <td>Timing Models</td><td>Final</td></tr> <tr> <td>Total logic elements</td><td>8 / 49,760 ( &lt; 1 % )</td></tr> <tr> <td>Total registers</td><td>0</td></tr> <tr> <td>Total pins</td><td>11 / 360 ( 3 % )</td></tr> <tr> <td>Total virtual pins</td><td>0</td></tr> <tr> <td>Total memory bits</td><td>0 / 1,677,312 ( 0 % )</td></tr> <tr> <td>Embedded Multiplier 9-bit elements</td><td>0 / 288 ( 0 % )</td></tr> <tr> <td>Total PLLs</td><td>0 / 4 ( 0 % )</td></tr> <tr> <td>UFM blocks</td><td>0 / 1 ( 0 % )</td></tr> <tr> <td>ADC blocks</td><td>0 / 2 ( 0 % )</td></tr> </table>	Flow Status	Successful - Sat Oct 31 20:53:08 2020	Quartus Prime Version	20.1.0 Build 711 06/05/2020 SJ Lite Edition	Revision Name	task1_top	Top-level Entity Name	task1_top	Family	MAX 10	Device	10M50DAF484C7G	Timing Models	Final	Total logic elements	8 / 49,760 ( < 1 % )	Total registers	0	Total pins	11 / 360 ( 3 % )	Total virtual pins	0	Total memory bits	0 / 1,677,312 ( 0 % )	Embedded Multiplier 9-bit elements	0 / 288 ( 0 % )	Total PLLs	0 / 4 ( 0 % )	UFM blocks	0 / 1 ( 0 % )	ADC blocks	0 / 2 ( 0 % )
Flow Status	Successful - Sat Oct 31 20:53:08 2020																																
Quartus Prime Version	20.1.0 Build 711 06/05/2020 SJ Lite Edition																																
Revision Name	task1_top																																
Top-level Entity Name	task1_top																																
Family	MAX 10																																
Device	10M50DAF484C7G																																
Timing Models	Final																																
Total logic elements	8 / 49,760 ( < 1 % )																																
Total registers	0																																
Total pins	11 / 360 ( 3 % )																																
Total virtual pins	0																																
Total memory bits	0 / 1,677,312 ( 0 % )																																
Embedded Multiplier 9-bit elements	0 / 288 ( 0 % )																																
Total PLLs	0 / 4 ( 0 % )																																
UFM blocks	0 / 1 ( 0 % )																																
ADC blocks	0 / 2 ( 0 % )																																

- This correctly shows that the design used 8 logic elements (out of nearly 50,000) and 11 pins.
- Programme the DE10 with YOUR design with the file: task1\_top.sof. (See Task 0 if you have forgotten how to do this.) Test you design.

**Congratulations! You have now completed your design from beginning to the end.**

### Put verified modules in mylib

For the rest of this module, you will design and verify various SystemVerilog modules which you will reuse. You should copy **hexto7seg.sv** (and others in the future) to the “**mylib**” folder and include them in your new design as necessary.

**Note:** When you perform a compilation, there may be a popup window informing you that some “Chain\_x.cdf” file has been modified, and ask if you wish to save it. Just click NO.



## Task 2: Explore Netlist Viewer and Timing Analyzer

### Step 1: Viewing the design

Quartus Prime provides a graphical view of the synthesized design. Exploring this provides you with some insight into how the SystemVerilog HDL code is turned into actual FPGA hardware.

- Click **Tools > Netlist Views > RTL Viewer**

This should appear on your screen:

- Push down into this block and investigate what is being displayed and how it relates to the decode logic.

RTL Viewer only shows the abstract Boolean description of the design, not the physical implementation on the FPGA.

- Click **Tools > Netlist Views > Technology Map Viewer (post mapping)**

Explain what you find and link this back to the Compilation Report.

### Step 2: Timing Analyzer

Click **Tools > Timing Analyzer**

A Timing Analyzer window will appear.

Now click **Report datasheet**. The Timing Analyzer tool will provide datasheet type table showing the propagation delays of all the paths in your design

Make a copy of the results in your logbook. Consider these worst-case delays at different temperature and explain what you found.

### Step 3: Test yourself

Create your own design in task2 folder (top-level file is **task2\_top.sv**) to display all 10-bit sliding switches as hexadecimal on three of the 7-segment LED displays.



	Input Port	Output Port
1	SW[0]	HEX0[0]
2	SW[0]	HEX0[1]
3	SW[0]	HEX0[2]
4	SW[0]	HEX0[3]
5	SW[0]	HEX0[4]
6	SW[0]	HEX0[5]
7	SW[0]	HEX0[6]
8	SW[1]	HEX0[0]
9	SW[1]	HEX0[1]
10	SW[1]	HEX0[2]
11	SW[1]	HEX0[3]
12	SW[1]	HEX0[4]
13	SW[1]	HEX0[5]
14	SW[1]	HEX0[6]
15	SW[2]	HEX0[0]
16	SW[2]	HEX0[1]
17	SW[2]	HEX0[2]
18	SW[2]	HEX0[3]
19	SW[2]	HEX0[4]
20	SW[2]	HEX0[5]
21	SW[2]	HEX0[6]
22	SW[3]	HEX0[0]
23	SW[3]	HEX0[1]
24	SW[3]	HEX0[2]
25	SW[3]	HEX0[3]
26	SW[3]	HEX0[4]
27	SW[3]	HEX0[5]
28	SW[3]	HEX0[6]