# ECSE-325
# Digital Systems

**Lab #2** – ***DE1-SoC board Programming and Display*** **Winter 2023**

## Introduction .

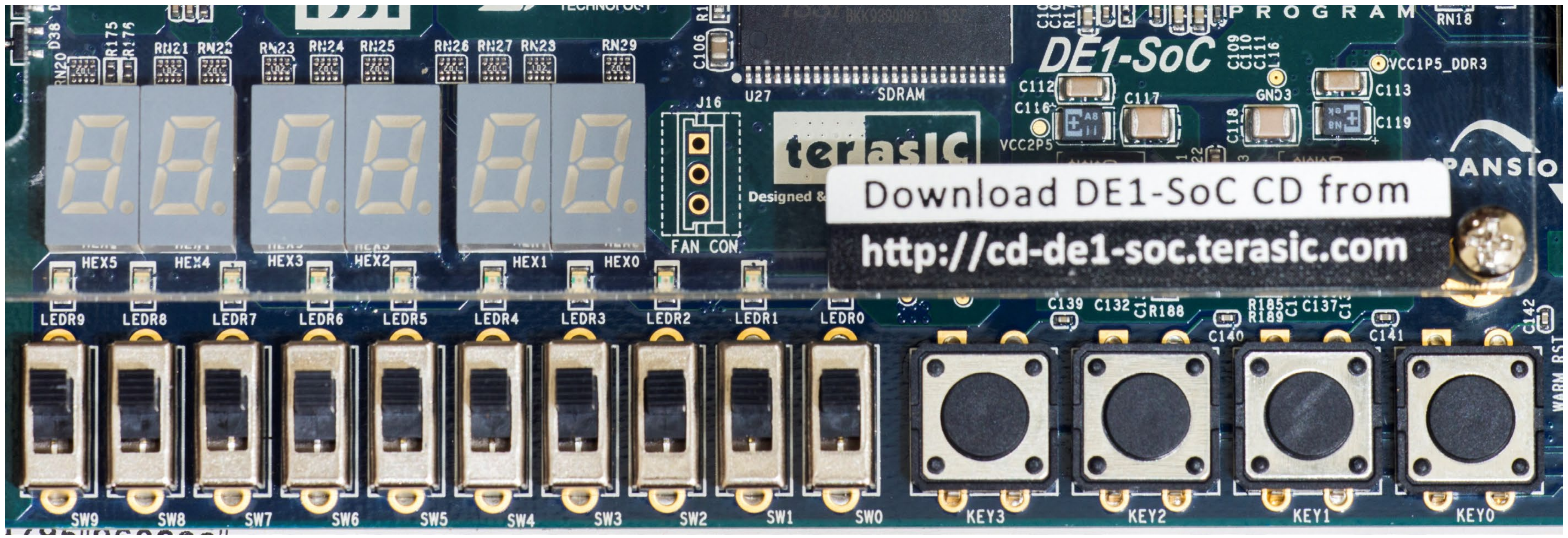
In this lab you will learn how to download designs to the DE1-SoC board. You will learn how to connect various peripheral elements, such as switches and 7-segment LED displays on the board to the Cyclone V FPGA chip on the board.

# Display of Decimal Values on the DE1-SoC Development Board.

A 7-segment LED display has 7 individual light-emitting segments, as shown in the picture below. By turning on different segments at any one time we can obtain different characters or numbers. There are six of these attached to the FPGA on the DE1-SoC board. You will use 4 of these to display the decimal value output of your ARCCOS circuit.

We will need two circuits to do this:

- 10-bit binary to 4-digit BCD (binary-coded-decimal) converter
- 7-segment LED decoder driver, which converts the 4-bit BCD value to the value of the 7 segments on/off state needed to display that digit



## 10-bit Binary to BCD Converter .

For your project, you will need to display various binary values. But binary is hard for most people to read. So you will need to display your values in base-10 digits.

A common method for doing binary to BCD conversion is called the “double-dabble” algorithm.

You can read the Wikipedia article on this:
https://en.wikipedia.org/wiki/Double_dabble

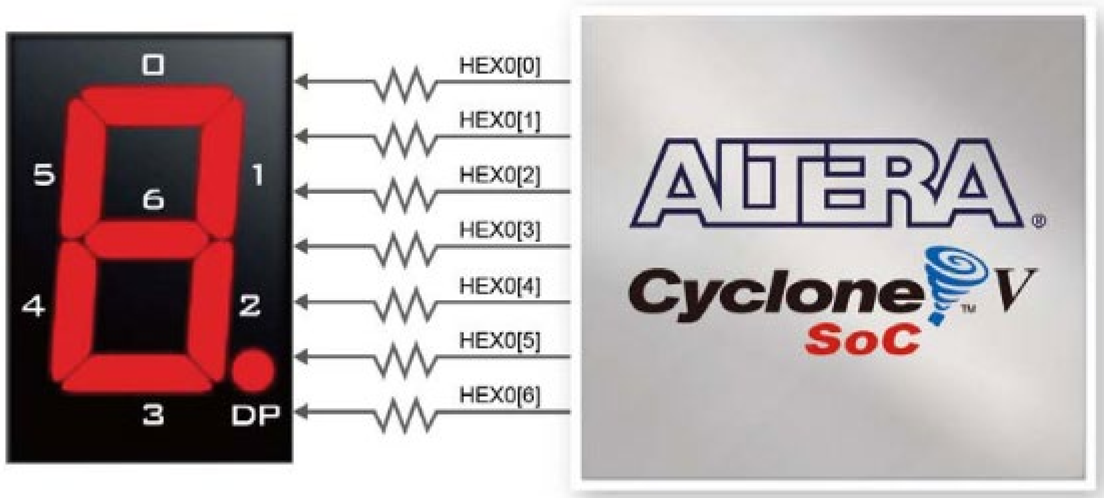
You can use the Verilog implementation shown on the Wikipedia page, or you can write your own VHDL code. The Verilog code is included in the lab assignment as bin2bcd.v

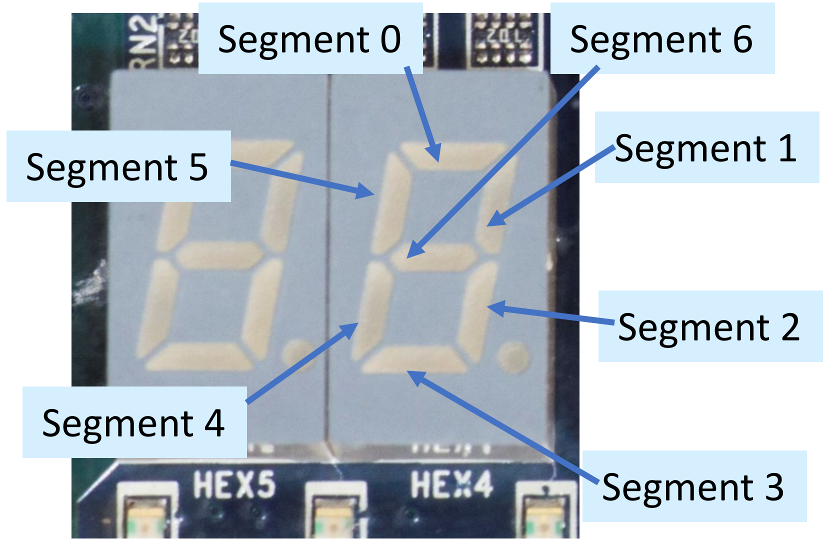
You can instantiate Verilog modules in your VHDL code just as you would VHDL components.

In the next part of the lab, you will design the circuit that will be used to drive the 7-segment LEDs on the DE1-SoC board.

It takes in a 4-bit BCD code representing the 10 digits between 0 and 9 and generates the 7-segment display associated with the input code, as shown on the next page.

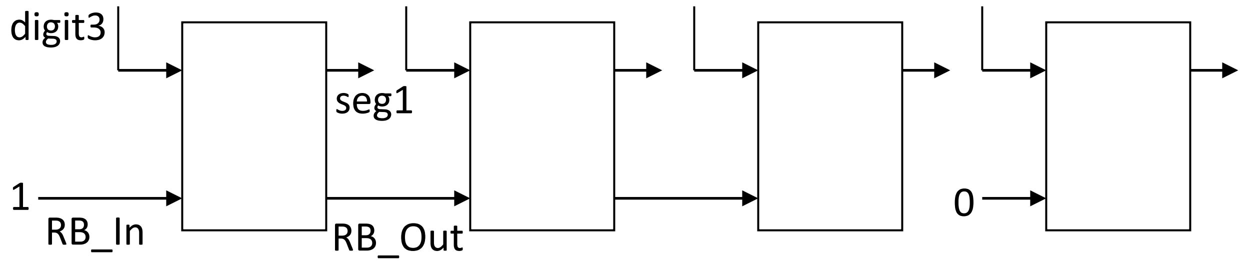
*Note: The outputs should be made **active-low**. This is convenient, as many LED displays, including the ones on the DE1-SoC board, turn on when their segment inputs are driven low.*





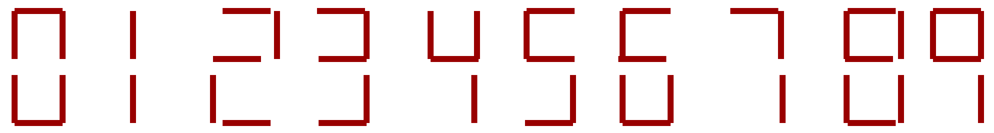
Your circuit should have ***ripple-blanking*** capability. Ripple blanking is the turning off of leading zeroes in a multi-digit display. For example, suppose we had a 4 digit display, with one decimal point. Thus, we could display numbers such as **241.2** and **788.6**. But what about displaying numbers with value less than 100? If we didn't blank the leading zeroes our display would look like **009.5** or **083.4**. This looks ugly and unprofessional, so we would rather display **9.5** and **83.4** in these cases, where the leading zeroes have been suppressed.

The ripple blanking output should be connected to the ripple-blanking input of the next display decoder to the right. The ripple-blanking input of the left-most LED decoder should be connected to '1'. The rightmost LED decoder, and any decoders right of the decimal point should never be blanked, so their ripple blanking inputs should be connected to '0'. In the case described above, the display circuit would look like:



If RB_In = 1 and digiti = 0 then blank display and set RB_Out = 1, else set RB_Out = 0

Segment patterns and input values:



Value: 0 1 2 3 4 5 6 7 8 9

Values 10 through 15 should result in a blank output (no segments on).

When the ripple blanking input is high, and the input value is 0, all segments should be off, and the ripple blanking output set high. Otherwise, the ripple blanking output should be low.

To implement the 7-segment LED decoder, write a VHDL description using ***a single process block with a case statement***. This circuit should be completely combinational (no clock).

Use the following entity declaration, replacing the NN in gNN_7_segment_decoder with your group's number (e.g. g08).

```
entity gNN_7_segment_decoder is
 port ( value               : in std_logic_vector(3 downto 0);
        RB_In               : in std_logic;
        RB_Out              : out std_logic;
        segments            : out std_logic_vector(6 downto 0));
end gNN_7_segment_decoder;
```

## Testing of the binary-BCD and 7-Segment LED Decoder on the DE1-SoC Board .

You will now test the 7-segment decoder circuit you designed earlier. Create, using VHDL, a circuit that connects 4 instances of the LED decoder circuit to the binary-BCD converter circuit. This circuit should also connect the output of the ARCCOS circuit designed in lab #1 to the inputs of the binary-BCD converter. [include the 7-segment LED decoder, binary-to-BCD converter, and the ARCCOS circuit as components]
You should set this VHDL entity as the "top-level" entity on Quartus.

You will then test this combined circuit on the DE1-SoC board, using 8 of the 10 slide switches on the board to define the 8-bit input to the ARCCOS circuit.

First, compile the VHDL for the combined circuit in the Quartus software. This will determine the port signals for the top-level entity, which we can then connect to various peripherals on the development board.

Once you have compiled the combined circuit, it is time to map it onto the target hardware, in this case the Cyclone V chip on the Altera DE1-SoC board. Please read over the DE1-SoC user's manual, which can be found in the lab#2 assignment information on myCourses.

Since you will now be working with an actual device, you have to be concerned with which FPGA device package pins the various inputs and outputs of the project are connected.

In particular, you will want to connect the LED segment outputs from each instance of the ***gNN_7_segment_decoder*** circuit to the corresponding segments of one of the four 7-segment LED displays on the DE1-SoC board. You will just use 4 of the 6 7-segment LEDs on the board.

**The mapping of the DE1-SoC board's 7-segment LEDs' segments to the pins on the Cyclone FPGA device is listed in Table 3.9 on page 27 of the DE1-SoC Board Users Manual.**

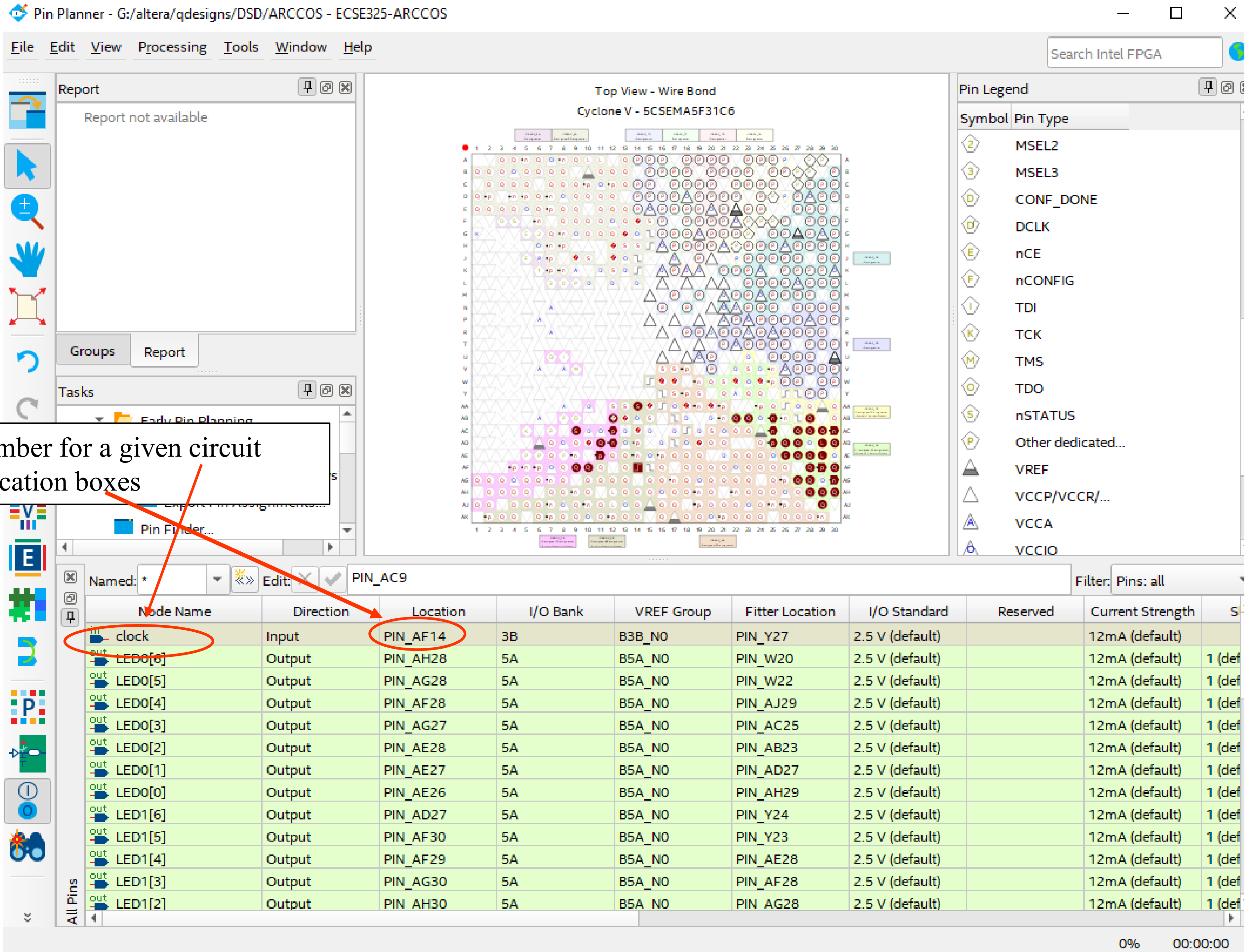
You will also connect, for testing purposes, 8 of the 10 ***slide switches*** on the DE1-SoC board to the input of the combined circuit.

The mapping of the slide switches to the FPGA pins is given in Table 3.6, and that of the pushbuttons is given in Table 3.7, on page 25 of the DE1-SoC user's manual.

Finally, we will use one of the 50MHz clock signals generated on the board as our clock. The available clock signals are listed in Table 3.5 on page 22, along with their FPGA pin connections. Use CLOCK_50, which is connected to FPGA PIN_AF14.

You can tell the compiler of your choices for pin assignments for your inputs and outputs in the ***Assignment Editor***, which can be opened by choosing the ***Pins*** item in the ***Assignments*** menu, as shown in the screenshot on the next page.

Enter the pin number for a given circuit node into the Location boxes



Pin Planner - G:/altera/qdesigns/DSD/ARCCOS - ECSE325-ARCCOS

File Edit View Processing Tools Window Help

Search Intel FPGA

Report

Report not available

Groups Report

Tasks

Early Pin Planning

Export Pin Assignments...

Pin Finder...

Top View - Wire Bond
Cyclone V - 5CSEMA5F31C6

Pin Legend

| Symbol | Pin Type |
|---|---|
| ② | MSEL2 |
| ③ | MSEL3 |
| Ⓓ | CONF_DONE |
| Ⓓ | DCLK |
| Ⓔ | nCE |
| Ⓕ | nCONFIG |
| Ⓘ | TDI |
| Ⓚ | TCK |
| Ⓜ | TMS |
| Ⓞ | TDO |
| Ⓢ | nSTATUS |
| Ⓟ | Other dedicated... |
| ▲ | VREF |
| △ | VCCP/VCCR/... |
| △ | VCCA |
| △ | VCCIO |

Named: * Edit: PIN_AC9 Filter: Pins: all

| Node Name | Direction | Location | I/O Bank | VREF Group | Fitter Location | I/O Standard | Reserved | Current Strength | S |
|---|---|---|---|---|---|---|---|---|---|
| clock | Input | PIN_AF14 | 3B | B3B_N0 | PIN_Y27 | 2.5 V (default) | | 12mA (default) | |
| LED0[6] | Output | PIN_AH28 | 5A | B5A_N0 | PIN_W20 | 2.5 V (default) | | 12mA (default) | 1 (def |
| LED0[5] | Output | PIN_AG28 | 5A | B5A_N0 | PIN_W22 | 2.5 V (default) | | 12mA (default) | 1 (def |
| LED0[4] | Output | PIN_AF28 | 5A | B5A_N0 | PIN_AJ29 | 2.5 V (default) | | 12mA (default) | 1 (def |
| LED0[3] | Output | PIN_AG27 | 5A | B5A_N0 | PIN_AC25 | 2.5 V (default) | | 12mA (default) | 1 (def |
| LED0[2] | Output | PIN_AE28 | 5A | B5A_N0 | PIN_AB23 | 2.5 V (default) | | 12mA (default) | 1 (def |
| LED0[1] | Output | PIN_AE27 | 5A | B5A_N0 | PIN_AD27 | 2.5 V (default) | | 12mA (default) | 1 (def |
| LED0[0] | Output | PIN_AE26 | 5A | B5A_N0 | PIN_AH29 | 2.5 V (default) | | 12mA (default) | 1 (def |
| LED1[6] | Output | PIN_AD27 | 5A | B5A_N0 | PIN_Y24 | 2.5 V (default) | | 12mA (default) | 1 (def |
| LED1[5] | Output | PIN_AF30 | 5A | B5A_N0 | PIN_Y23 | 2.5 V (default) | | 12mA (default) | 1 (def |
| LED1[4] | Output | PIN_AF29 | 5A | B5A_N0 | PIN_AE28 | 2.5 V (default) | | 12mA (default) | 1 (def |
| LED1[3] | Output | PIN_AG30 | 5A | B5A_N0 | PIN_AF28 | 2.5 V (default) | | 12mA (default) | 1 (def |
| LED1[2] | Output | PIN_AH30 | 5A | B5A_N0 | PIN_AG28 | 2.5 V (default) | | 12mA (default) | 1 (def |

All Pins

0% 00:00:00

Double-check your entered pin assignments, as it is easy to make mistakes.

Once you have assigned all the inputs and outputs of your circuit to appropriate device pins, ***re-compile*** your design.

**You can check that the pins have been assigned correctly by looking at the floorplan on the pin planner (zoom in) and verifying that the right pins have been used.**

Your design is now ready to be downloaded to the target hardware. Read section 3.2 of the DE1 user's manual for information on configuring (programming) the Cyclone V FPGA on the board (pages 14 through 18). ***You will be using the JTAG mode to configure the device.***

**Take the DE1-SoC board out of the kit box and connect the USB cable to the computer's USB port and to the USB connector on the Altera board. Then connect the power supply and press the red button to turn on the board. You should (might not) see a test pattern displayed on the LEDs.**

**Carry out steps *1 through 6* on pages 14-18 of the DE1-SoC User Manual to download your design to the DE1-SoC board.**

Connect the USB cable from the computer to the board at the USB-Blaster II port

Then connect the power supply to the Power DC jack

Press the red Power ON/OFF button to turn on the board.

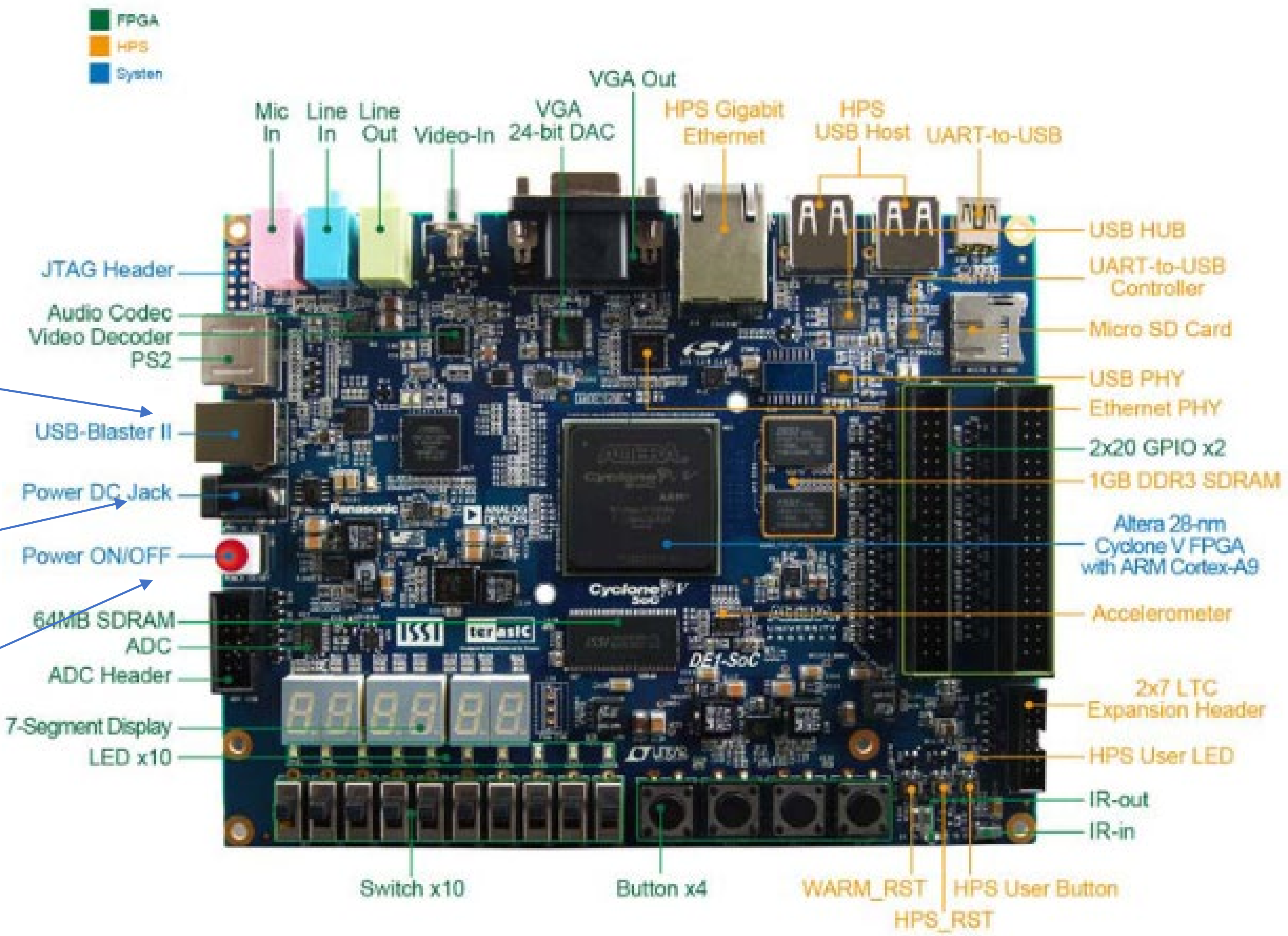


**Figure 2-1 DE1-SoC development board (top view)**

If the Auto Detect button in the programmer is greyed out, click on Hardware Setup and select the DE-SoC entry in the Hardware pane. Press “Close”.
The Auto Detect button should now be available.

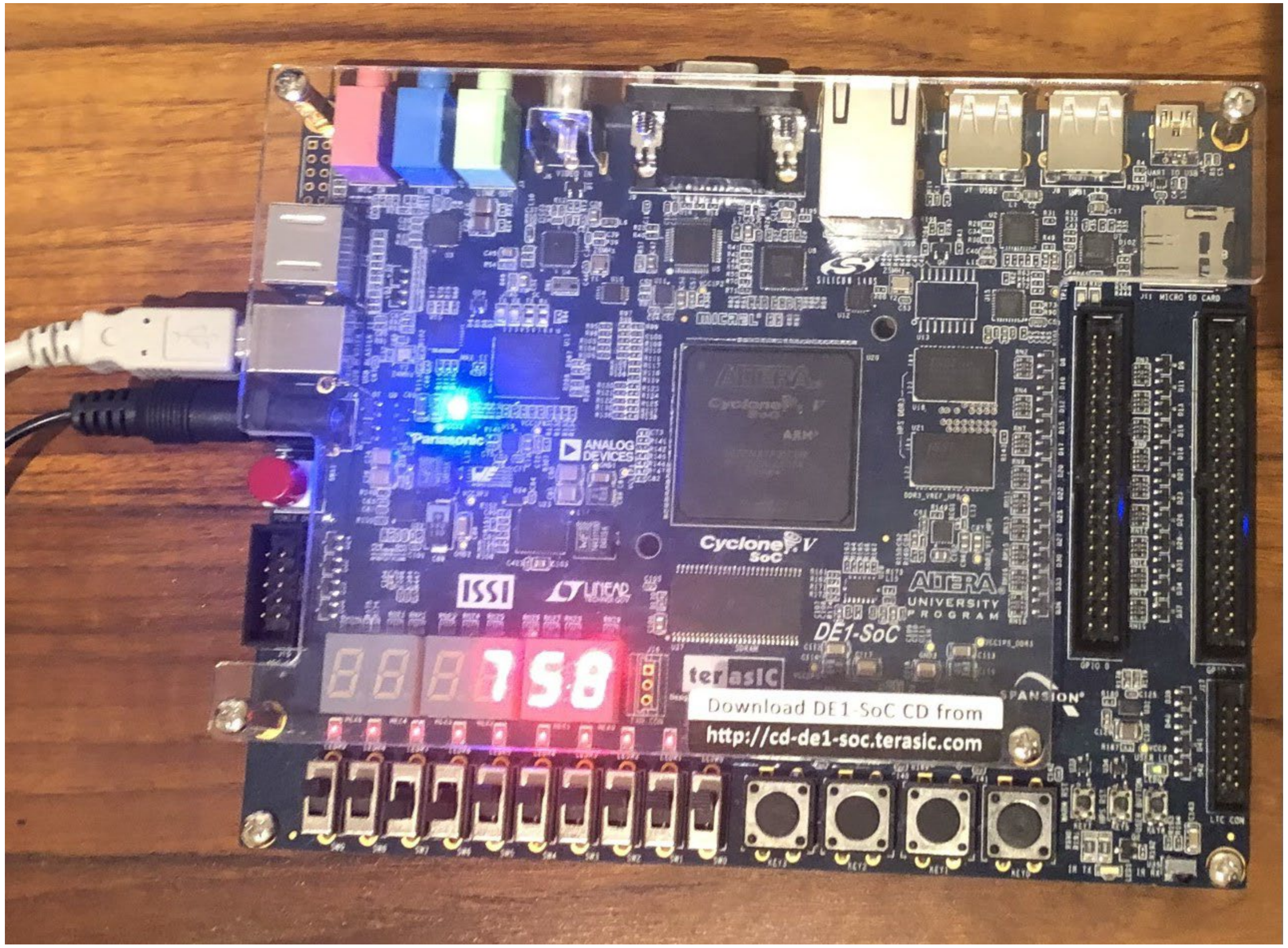
When the instructions say to select the .sof file, note that this file might be in the “output_files” directory.

Test your system by setting the switches to input different binary coded values, and observing whether the 7-segment displays show the expected values (go through all 256 different input patterns).

If your LED digits look strange, it could be due to one of the following issues:

- Pin assignments are incorrect
- 7-segment driver outputs are not active low
- The order of the segment outputs is wrong

If all goes well, you should see something like the screenshot below, which shows *arccos(63/256) ~ 75.8* degrees



## Writeup the Lab Report .

Write up a short report describing your arccos test circuit that you designed in this lab. This report should be submitted in pdf format.

The report must include the following items:

- A header listing the group number, the names and student numbers of each group member, and a title, giving the name (e.g. ***gNN_ARCCOS_test***) of the circuit.
- A description of the circuit's function, listing the inputs and outputs.
- The VHDL description of the circuits designed in the lab.
- A listing of the displayed output for all 256 different input patterns. Compare the obtained results with pre-computed values.
- A photograph of your board displaying one of the test values.
- A screenshot of the “flow summary” from the compilation report.
- A screenshot of the “Fmax Summary” for the “Slow 1100mV 85C Model” from the compilation report.

**The report is due on March 31, at 11:59 PM.**

## Submit the Lab Report to myCourses .

The lab report, and all associated design files (*the design files are the vhdl (.vhd) files*) must be submitted, as an assignment to the myCourses site. Only one submission need be made per group (both students will receive the same grade).

**Combine all of the files that you are submitting into one *zip* file and name the zip file gNN_LAB_1.zip (where NN is your group number).**