Plant Watering Suggestion System

Josie S. Lyon

CPE 526

VLSI Using Hardware Description Languages, Modeling, and Synthesis

April 23, 2021

Table of Contents

1. Executive Summary

The advent of the COVID-19 pandemic has generated a noticeable growth in houseplant acquisition. With little to do beyond staring at a wall, even those with the brownest of thumbs have decided to give domestic botany a swirl. Unfortunately, these brave new endeavors have resulted in the deaths of likely thousands of succulents, monsteras, and peace lilies alike.

The biggest question that new and even experienced plant owners ask upon acquiring a plant is, "How much do I water it?" The answer can be very different depending on the plant. Over and underwatering is a primary, heartbreaking cause of death for houseplants. This project seeks to address this problem by providing the lay plant person a Plant Watering Suggestion System.

This Plant Watering Suggestion System provides the user with an easy-to-use water suggestion station based on plant type. A dual purpose of this project is to provide current and future students of VHDL an interesting sample project that uses VHDL combined with an accessible development board (the Terasic DE Series) and affordable sensors to create a functioning utility. This project configures analog input from the sensor, the 7-segment display, the switch inputs, and LED outputs. This report should serve as an instruction manual for those who may want to recreate the project.

2. Introduction

2.1. Contact Information

Table 1 – Contact Information

Name	Email	Phone
Josie Lyon	Jsl0033@uah.edu	615-546-9434

2.2. Requirements

i. The plant watering suggestion system (referred to as “the system” for the reminder of these requirements) shall allow the user to input three different plant type settings, one for dry, medium, and moist soil preferences.

ii. The system shall display the selected soil moisture preference.

iii. The system shall suggest whether a plant should be watered according to the selected soil moisture preference.

iv. The system shall display the plant watering suggestion in the form of a “yes” or “no”.

v. The source files for the system shall be written in VHDL.

vi. The control logic for the system shall be implemented in an FPGA.

vii. A physical test bench shall be created to test the system’s suggestion validity in a variety of different plant types and soil moisture levels.

2.3. Hardware & Materials Overview

2.3.1. Terasic DE10-Lite Development Board

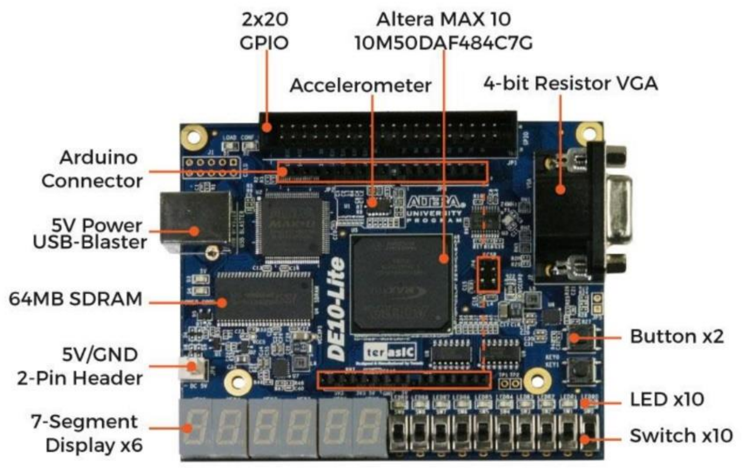


Figure 1: DE10-Lite Development Board Component Layout

Table 2: DE10-Lite Components Used in System Configuration
Altera MAX 10 FPGA 10M50DAF484C7G
5V Power USB-Blaster
7-Segment Display x6
LED x10
Switch x10
A/D Converter (MAX 10 FPGA ADC1, shares a port with Arduino connector)

The Terasic DE10-Lite board came with a lot of support, including a detailed manual with diagrams, component descriptions, FPGA pin tables, and demo projects in Verilog. A link to the user manual for the DE10-Lite can be found in the Appendix. The only challenging piece that was easily configured was the configuration of the ADC input from the moisture sensor. This required the creation of an Altera ADC IP module that is then instantiated in the design. Fortunately, the DE10-Lite user's manual provides the demo, "ADC Measurement", written in Verilog that contains an already configured Altera ADC IP module. This module was acquired and absorbed into the system design, and the demo was converted to VHDL which served as the "blinky lights" foundation of the design.

2.3.2. Keyestudio Moisture Sensor

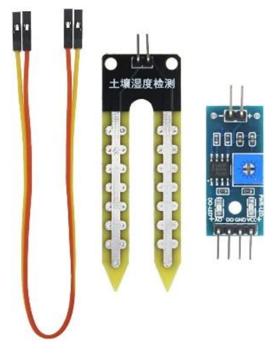


Figure 2: Keyestudio Moisture Sensor

Table 3: Keyestudio Moisture Sensor Specifications
Power Supply Voltage 3.3V or 5.0V
Working Current: ≤ 20mA
Output Voltage: 0-2.3V (When the sensor is totally immersed in water, the voltage will be 2.3V) the higher humidity, the higher the output voltage
Sensor Type: Analog/Digital output

The Keyestudio Moisture Sensor was initially difficult to find support for, as it was ordered from Amazon, misspelled by the seller, and the product labeling is written in Chinese. Fortunately, a wiki page was found for a similar Keyestudio sensor (see Appendix) and the team had direct access to a colleague who double majored in physics and Japanese in their undergraduate degree who was able to provide a rough translation. The characters in the center figure above reads “soil moisture meter”, whereas the righthand figure has two labeled LEDs, with the left LED says “power prompt” and the right LED says “switch prompt”. And behold, the “power prompt” LED lights up whenever the sensor is connected to power, and the “switch prompt” lights up when the sensor is place in water or moist soil, i.e., when a voltage signal is being generated.

The theory behind the sensor is based on Ohm’s Law:

Voltage = Current x Resistance, or

$$V = IR$$

Essentially, current flows through the two metal prongs of the sensor, and the sensor reads the current changes between the two prongs. Since the voltage is a known constant (3.3V or 5.0V), the resistance can easily be derived:

$$R = 3.3V / \Delta I$$

Water is an electrically conductive medium. So, if the sensors report a high resistance, then the medium must not be very conductive, which in turn means that the soil must not have a lot of water. Likewise, if the sensors report a low resistance, that means that electricity can flow easily, which in turn means that the soil must be very moist.

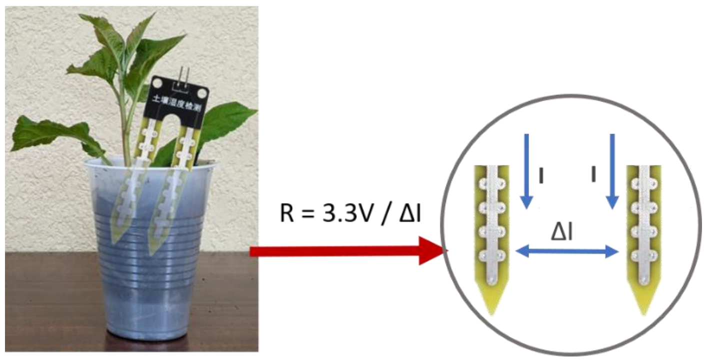


Figure 3: The Theory Behind the Keyestudio Sensor

2.3.3. Other Materials

Other materials include a Type A Male to Type B Male USB Cable, 1 breadboard, 5 male/female connectors, 1 male/male connectors, 2 female/female connectors. Of course, it should be noted that this circuit was created using the materials available, and that multiple possible connector configuration are possible to achieve the same set up.

2.4. Bill of Materials

Table 4: Bill of Materials

Item	Cost
Terasic DE10-Lite Development Board w/ MAX 10 FPGA	$95.00
Keyestudio Moisture Sensor	$7.99
Simple Breadboard Circuit	$1.50

3. Design

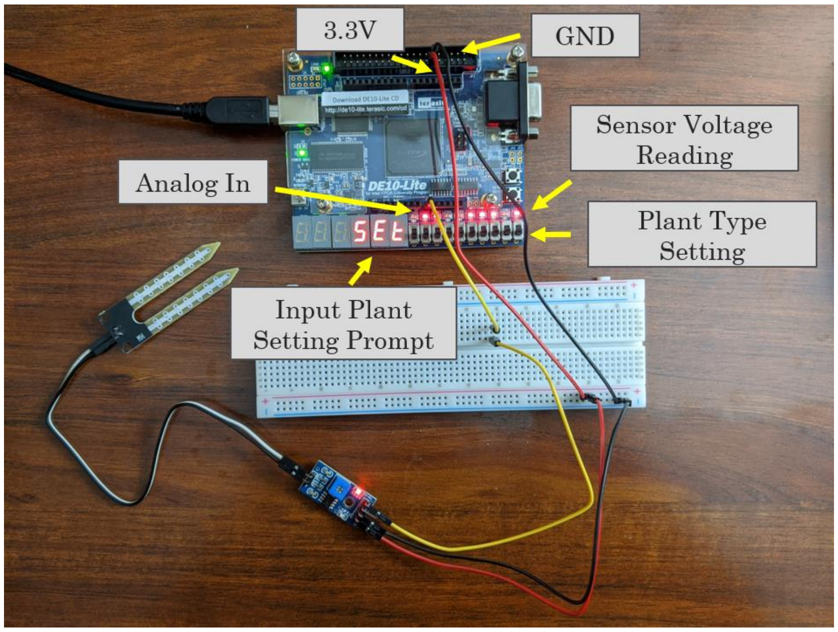


Figure 4: Annotated Design Overview

The design is broken down into six distinct pieces (five entities and 1 procedure):

- Top Level entity
- Plant Type Setting entity
- State Machine entity
- ADC Sensor Input entity

- Watering Suggestion entity
- Alphanumeric to 7-Segment Display Converter procedure

3.1. Top Level (Design Overview)

The following RTL schematic was generated using the Quartus 18.1 Netlist Viewer toolkit. The four entities detailed above can be seen, and the entire view can be considered the top level schematic. It details two inputs (the clock and three switches) and seven outputs (the six 7-segment displays and ten LEDs).

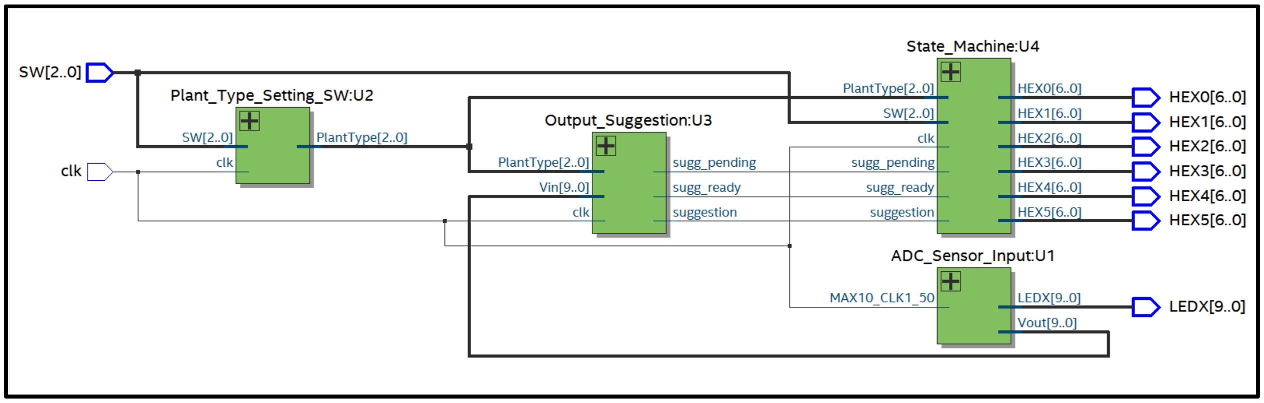


Figure 5: Design RTL Schematic

3.2. Plant Type Setting

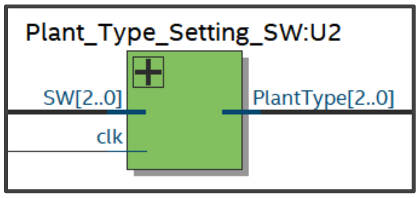


Figure 6: Plant Type Setting Entity Schematic

The system provides three different settings for users to input using the first three switches (righthand side) in the DE10-Lite board, SW0, SW1, and SW2. Only one switch is in the "on" (up, logic high) position at a time. A SW0 = '1' configuration corresponds to a plant that prefers dry soil (dry), a SW1 = '1' configuration corresponds to a plant that prefers medium moisture soil (avg), and a SW2 = '1' configuration corresponds to a plant that prefers moist soil (wet).

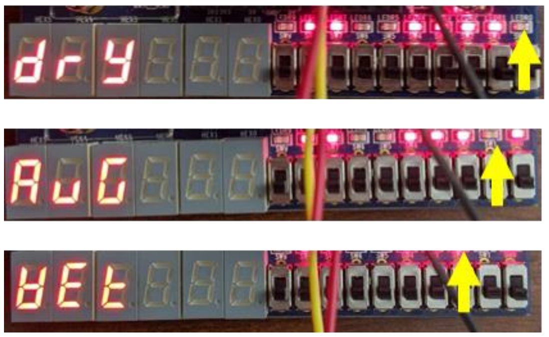


Figure 7: Plant Type Setting Switch & 7-Segment Display Configuration

3.3. State Machine

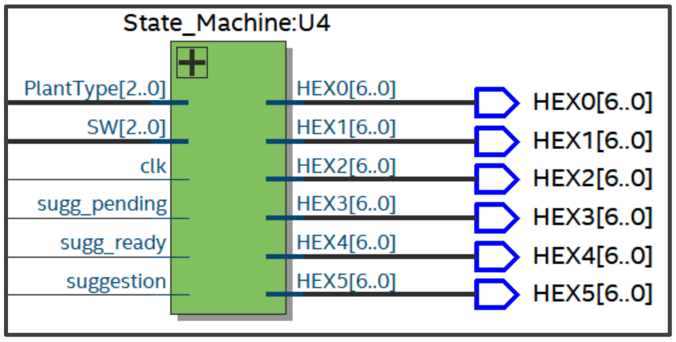


Figure 8: State Machine Entity Schematic

A finite state machine is used to drive the watering suggestion system. This simple state machine essentially takes the outputs of every other entity in the design, works them through the state machine, and ultimately outputs the watering suggestion of “yes” or “no” alongside the plant setting in the 7-segment display. The following diagram and table describe each state, its output, and its respective transitions in the state machine.

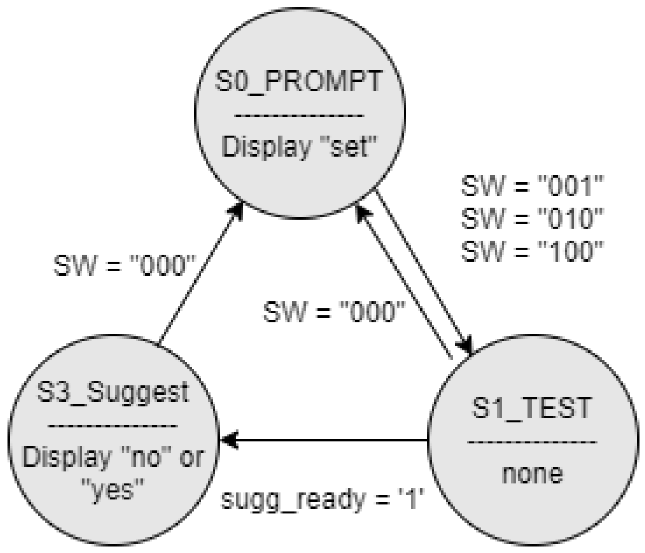


Figure 9: Finite State Machine Diagram

Table 5: State Machine for Watering Suggestion System

State	Function	Output	Next State	Transition Case
S0_PROMPT	This serves as the reset state. When all three switches are down, the user is prompted to input the plant type using the switches.	The 7-segment display displays “set”, then “dry”, “avg”, or “wet” when the plant type is set.	S1_Test	The transition occurs on the rising edge of the clock when exactly one of the three switches has been set.
S1_TEST	This state is where the user places the sensors into the soil of a plant, and the model calculates the soil moisture and suggestion.	None.	S2_SUGGEST	The transition occurs on the rising edge of the clock when the Watering Suggestion module outputs a “suggestion ready” signal.
S2_SUGGEST	This state provides the plant watering suggestion.	The 7-segment display displays “yes” or “no” according to the calculation suggestion.	S0_PROMPT	The transition occurs on the rising edge of the clock when no single switch is set.

3.4. ADC Sensor Input

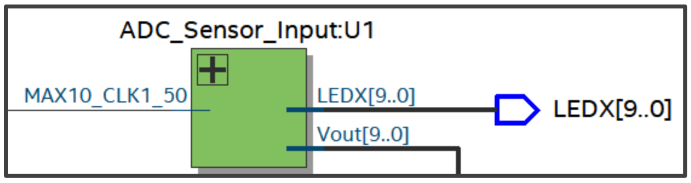
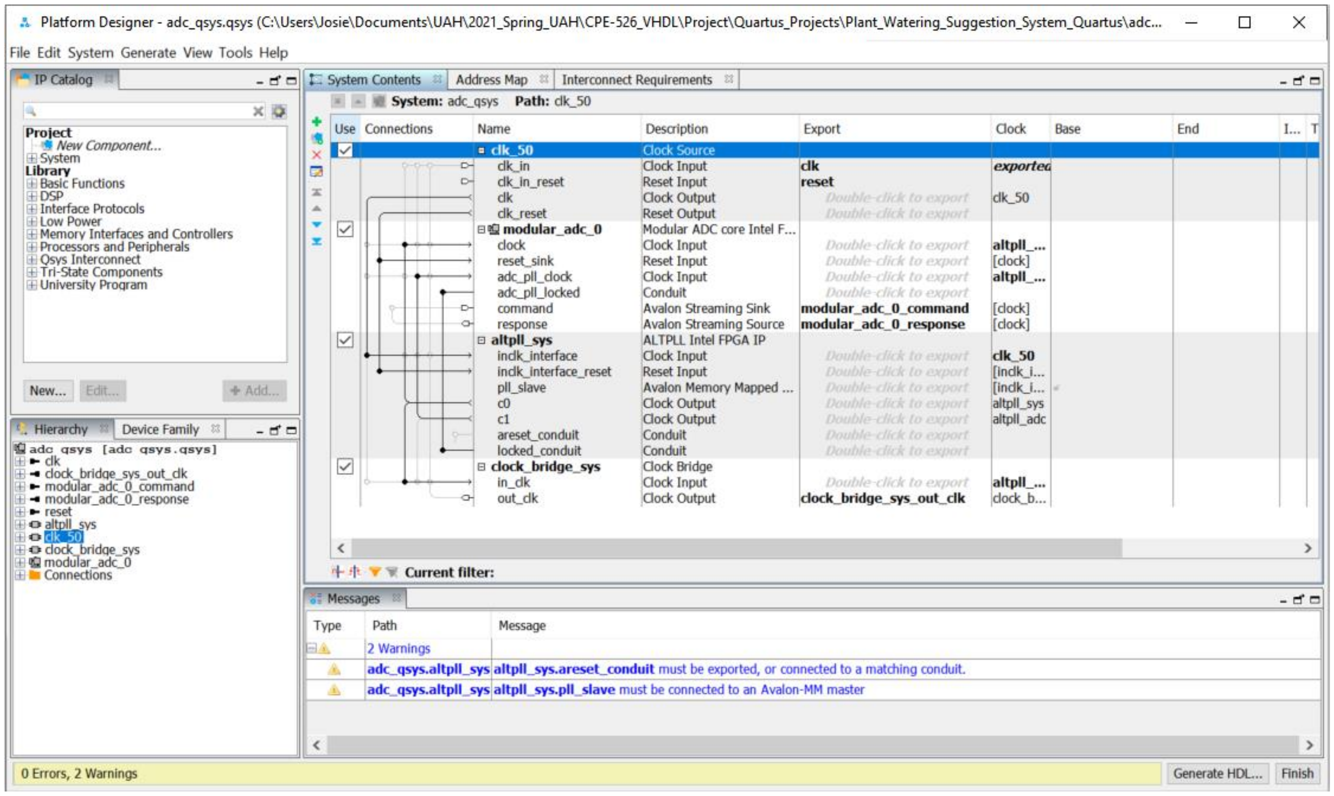


Figure 10: State Machine Entity Schematic

The sensor outputs an analog signal that corresponds to the soil moisture level. However, as mentioned in Section 2.3.1, the Altera ADC IP module had to be configured and then instantiated in the model. The Altera ADC IP taken from the Terasic DE10-Lite provided demo, "ADC Measurement" was specifically created using the Quartus Prime tool, Qsys, which is now called Platform Designer in newer version of Quartus Prime. Fortunately, all that was need was to take the two IP module files in the demo, adc_qsys.qsys and adc_qsys.qip, and add them to the Quartus project.


Platform Designer - adc_qsys.qsys (C:\Users\Josie\Documents\UAH\2021_Spring_UAH\CPE-526_VHDL\Project\Quartus_Projects\Plant_Watering_Suggestion_System_Quartus\adc...

IP Catalog
- Project
  - New Component...
  - System
- Library
  - Basic Functions
  - DSP
  - Interface Protocols
  - Low Power
  - Memory Interfaces and Controllers
  - Processors and Peripherals
  - Qsys Interconnect
  - Tri-State Components
  - University Program

Hierarchy
- adc_qsys [adc_qsys.qsys]
  - clk
  - clock_bridge_sys_out_clk
  - modular_adc_0_command
  - modular_adc_0_response
  - reset
  - altpll_sys
  - clk_50
  - clock_bridge_sys
  - modular_adc_0
  - Connections

System Contents — System: adc_qsys Path: clk_50

Use	Name	Description	Export	Clock
✓	clk_50	Clock Source		
	clk_in	Clock Input	clk	exported
	clk_in_reset	Reset Input	reset	
	clk	Clock Output	Double-click to export	clk_50
	clk_reset	Reset Output	Double-click to export	
✓	modular_adc_0	Modular ADC core Intel F...		
	clock	Clock Input	Double-click to export	altpll_...
	reset_sink	Reset Input	Double-click to export	[clock]
	adc_pll_clock	Clock Input	Double-click to export	altpll_...
	adc_pll_locked	Conduit	Double-click to export	
	command	Avalon Streaming Sink	modular_adc_0_command	[clock]
	response	Avalon Streaming Source	modular_adc_0_response	[clock]
✓	altpll_sys	ALTPLL Intel FPGA IP		
	inclk_interface	Clock Input	Double-click to export	clk_50
	inclk_interface_reset	Reset Input	Double-click to export	[inclk_i...
	pll_slave	Avalon Memory Mapped ...	Double-click to export	[inclk_i...
	c0	Clock Output	Double-click to export	altpll_sys
	c1	Clock Output	Double-click to export	altpll_adc
	areset_conduit	Conduit	Double-click to export	
	locked_conduit	Conduit	Double-click to export	
✓	clock_bridge_sys	Clock Bridge		
	in_clk	Clock Input	Double-click to export	altpll_...
	out_clk	Clock Output	clock_bridge_sys_out_clk	clock_b...

Current filter:

Messages

Type	Path	Message
⚠	2 Warnings	
⚠	adc_qsys.altpll_sys	altpll_sys.areset_conduit must be exported, or connected to a matching conduit.
⚠	adc_qsys.altpll_sys	altpll_sys.pll_slave must be connected to an Avalon-MM master

0 Errors, 2 Warnings

Figure 11: Altera ADC IP Module Configuration Quartus Platform Designer (adc_qsys.qsys)

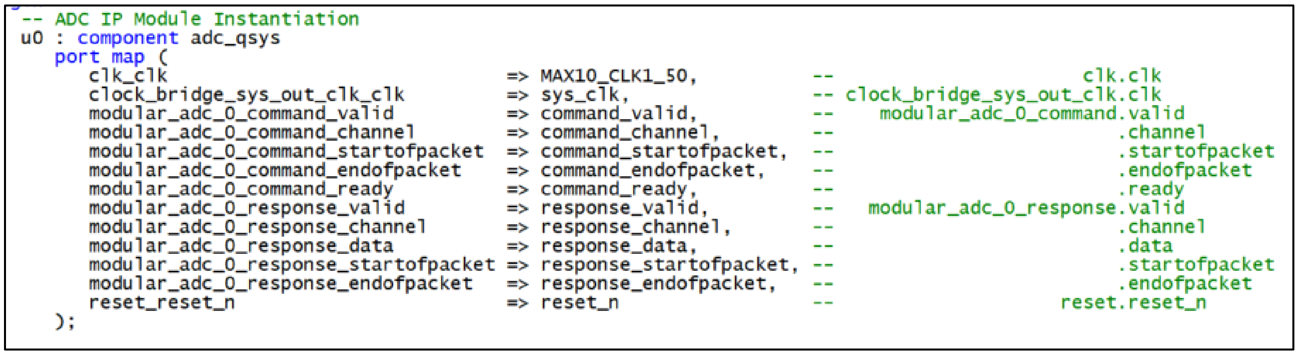

```vhdl
-- ADC IP Module Instantiation
u0 : component adc_qsys
    port map (
        clk_clk                              => MAX10_CLK1_50,           --                   clk.clk
        clock_bridge_sys_out_clk_clk         => sys_clk,                 -- clock_bridge_sys_out_clk.clk
        modular_adc_0_command_valid          => command_valid,           --    modular_adc_0_command.valid
        modular_adc_0_command_channel        => command_channel,         --                        .channel
        modular_adc_0_command_startofpacket  => command_startofpacket,   --                        .startofpacket
        modular_adc_0_command_endofpacket    => command_endofpacket,     --                        .endofpacket
        modular_adc_0_command_ready          => command_ready,           --                        .ready
        modular_adc_0_response_valid         => response_valid,          --   modular_adc_0_response.valid
        modular_adc_0_response_channel       => response_channel,        --                        .channel
        modular_adc_0_response_data          => response_data,           --                        .data
        modular_adc_0_response_startofpacket => response_startofpacket,  --                        .startofpacket
        modular_adc_0_response_endofpacket   => response_endofpacket,    --                        .endofpacket
        reset_reset_n                        => reset_n                  --                  reset.reset_n
    );
```

Figure 12: ADC IP Module Instantiation

The ADC IP module is then instantiated using a VHDL block generated by the Platform Designer tool in the entity that processes the analog input. Most of this entity is derived from the DE10-Lite provided demo, “ADC Measurement”, translated into VHDL. The process drives a 10-bit voltage reading output, and the ten LEDs light up according to that voltage reading. The user can then confirm that the sensor is indeed picking up different soil moisture readings when he or she sees the LEDs light up in a beautiful dancing display when the sensor is placed in different mediums.

3.5. Watering Suggestion

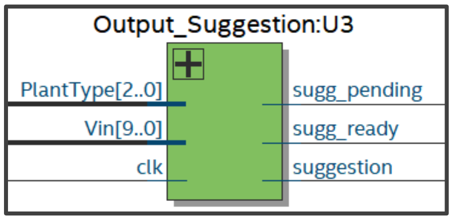


Figure 13: Watering Suggestion Entity Schematic

The watering suggestion entity inputs the voltage reading from the sensor along with the plant type indicated by the user, classifies the amount of moisture into five moisture categories, and outputs the suggestion. The five ranges are calculated by considered the maximum and minimum voltage readings. The voltage readings are reflected in the LED output, which was converted from binary to decimal by the team to determine the maximum and minimum voltage values:

When the sensor is immersed in water,	voltage ≈ 168*.
When the sensor is not in immersed in anything,	voltage ≈ 413.

*This value will vary depending on immersion.

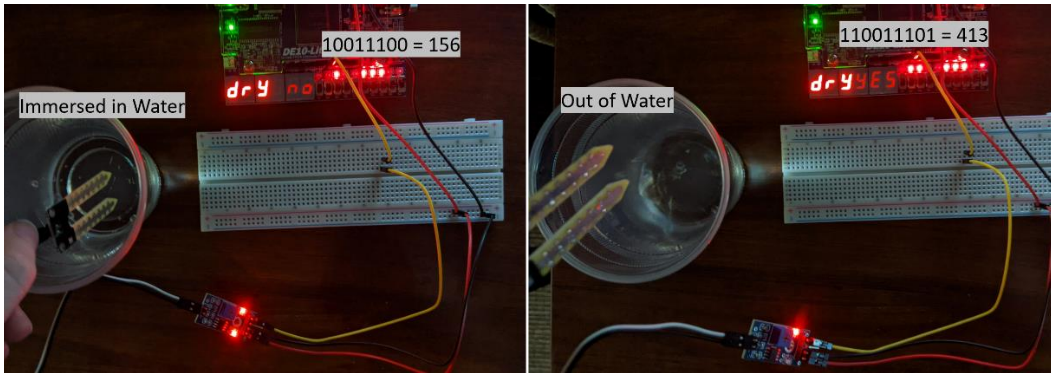


Figure 14: (Left) Sensor Fully Immersed in Water and (Right) Sensor out of Water LED Voltage Readings.

The five soil classification ranges are:

(413 – 168) / 5 ≈ 49

Very Dry	= 430 to 381
Dry	= 380 to 331
Medium Dry	= 330 to 281
Moist	= 280 to 231
Very Moist	= 230 to 168

The "Very Moist" range was left intentionally large to reflect that it is typically better to not over water a plant. When in doubt, don't water it!

The following table details the watering suggestion according to plant type and voltage reading.

Table #6: Watering Suggestion According to Plant Type and Soil Classification

Plant Type	Very Dry	Dry	Medium	Moist	Very Moist
Dry	yes	no	no	no	no
Avg	yes	yes	no	no	no
Wet	yes	yes	yes	no	no

3.6. Alphanumeric to 7-Segment Display Converter

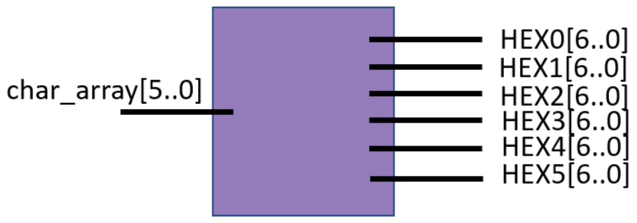


Figure 15: 7-Segment Alphanumeric Converter Procedure Schematic

The alphanumeric converter was created to allow the modeler to write numerous different strings to be displayed on the 7-segment display. The procedure takes in an array of characters (type defined in the same package as the procedure), and drives the input signal through a giant case statement that covers each letter of the alphabet, A-Z, and digit, 0-9. Important factors to remember are (1) each segment lights up on a logic low, not high, and (2) the segment numbers are read from right to left. For example, the letter "A" is represented by a std_logic_vector "0001000", which corresponds positionally to "6543210", and an '0' is a lit segment, and a '1' is a dark segment.

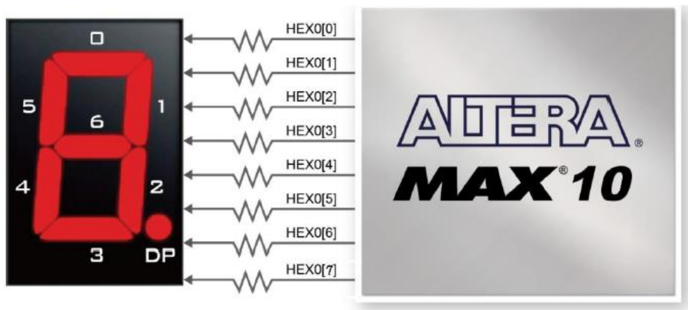


Figure 16: DE10-Lite 7-Segment Display

4. Verification

The verification method was a physical testbench that provided a variety of different sample plant soils. The first soil is dry, the second soil is medium dry, and the third soil is very moist. These soils were tested multiple times as the soil (mulch that varied from dense chips to fine grains) absorbed the water. The following table is a sample of the test bench results, which were repeated several times for verification.

Table #7: Sample Testbench & Results

Plant Type	Test Subject	Water Suggestion?	Expected Result?
DRY	Glass of Water	NO	Yes.
AVG	Glass of Water	NO	Yes.
WET	Glass of Water	NO	Yes.
DRY	Dry Soil	NO	Yes.
AVG	Dry Soil	YES	Either.
WET	Dry Soil	YES	Yes.
DRY	Semi-Moist Soil	NO	Yes.
AVG	Semi-Moist Soil	YES	Either.
WET	Semi-Moist Soil	YES	Either.
DRY	Wet Soil	NO	Yes.
AVG	Wet Soil	NO	Yes.
WET	Wet Soil	NO/YES	Yes.

A demonstration of the functioning plant watering suggestion system and the physical verification process can be found at the following link:

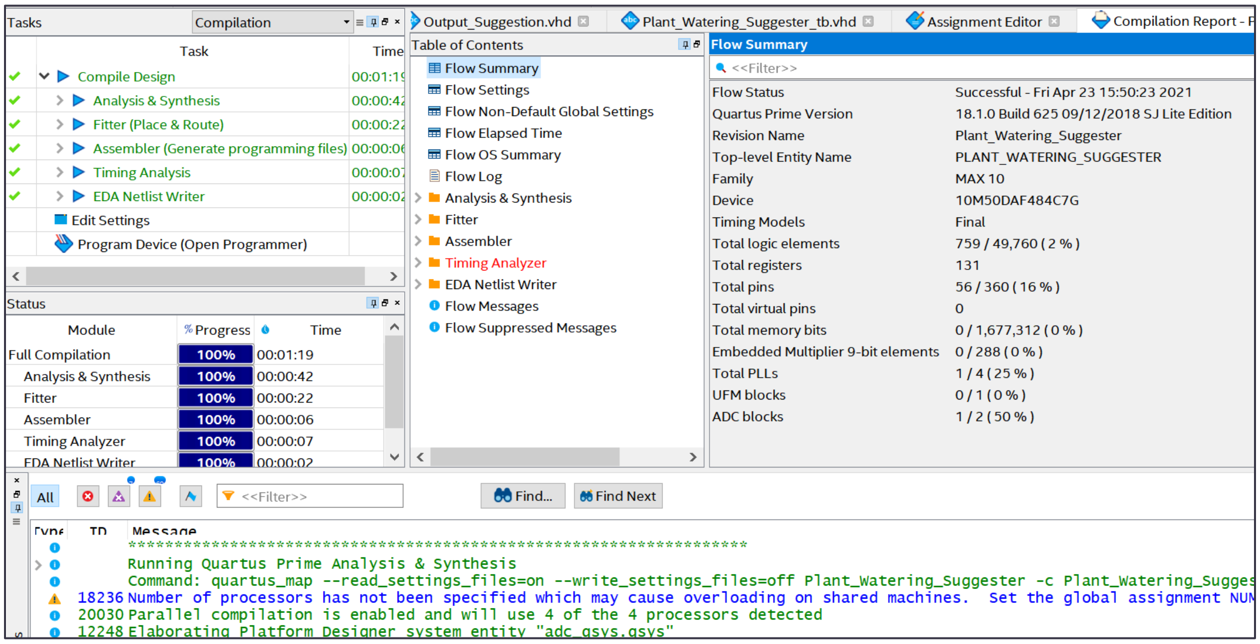
- https://tinyurl.com/yvkwcwbj

5. Synthesis

The design was synthesized using Intel's Quartus Prime Lite 18.1 programmable logic device design software. There are three main components of synthesizing the design: compilation, pin assignment, and programming the FPGA.

5.1. Design Compilation

The following screen displays when the design is successfully compiled. Take careful notice of the hardware utilization, with 2% of the total logic elements used, 16% of the total pins used, 25% of total PLLs used, and 50% of the ADC blocks used (which is expected, as the MAX 10 FPGA has two ADCs, with only one being accessible in the DE10-Lite board!).



Tasks — Compilation

	Task	Time
✓	Compile Design	00:01:19
✓	Analysis & Synthesis	00:00:42
✓	Fitter (Place & Route)	00:00:22
✓	Assembler (Generate programming files)	00:00:06
✓	Timing Analysis	00:00:07
✓	EDA Netlist Writer	00:00:02
	Edit Settings	
	Program Device (Open Programmer)	

Status

Module	Progress	Time
Full Compilation	100%	00:01:19
Analysis & Synthesis	100%	00:00:42
Fitter	100%	00:00:22
Assembler	100%	00:00:06
Timing Analyzer	100%	00:00:07
EDA Netlist Writer	100%	00:00:02

Output_Suggestion.vhd | Plant_Watering_Suggester_tb.vhd | Assignment Editor | Compilation Report

Table of Contents: Flow Summary, Flow Settings, Flow Non-Default Global Settings, Flow Elapsed Time, Flow OS Summary, Flow Log, Analysis & Synthesis, Fitter, Assembler, Timing Analyzer, EDA Netlist Writer, Flow Messages, Flow Suppressed Messages

Flow Summary

Flow Status	Successful - Fri Apr 23 15:50:23 2021
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	Plant_Watering_Suggester
Top-level Entity Name	PLANT_WATERING_SUGGESTER
Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	759 / 49,760 (2 %)
Total registers	131
Total pins	56 / 360 (16 %)
Total virtual pins	0
Total memory bits	0 / 1,677,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 288 (0 %)
Total PLLs	1 / 4 (25 %)
UFM blocks	0 / 1 (0 %)
ADC blocks	1 / 2 (50 %)

```
Type  ID    Message
            *******************************************************************************
            Running Quartus Prime Analysis & Synthesis
            Command: quartus_map --read_settings_files=on --write_settings_files=off Plant_Watering_Suggester -c Plant_Watering_Sugges
      18236 Number of processors has not been specified which may cause overloading on shared machines.  Set the global assignment NUM
      20030 Parallel compilation is enabled and will use 4 of the 4 processors detected
      12248 Elaborating Platform Designer system entity "adc_qsys.qsys"
```

Figure 17: Quartus Compilation Report

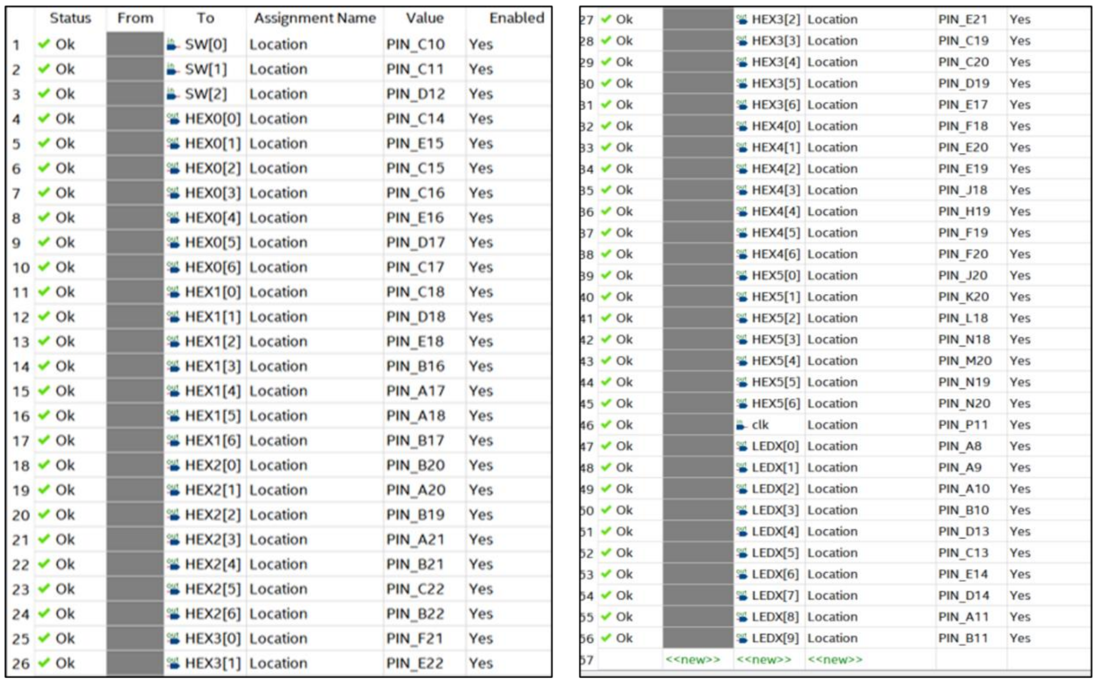
The steps to compiling a design in Quartus are as follows:

1. In the Project Navigator view, view Files, and right click the top-level entity and “Set as Top-Level Entity”.
2. Go to the top bar of icons and click the plain blue triangle to start compilation (alternatively, go to the Processing tab, and click “Start Compilation”.

3. Wait for a few minutes for compilation to be complete. If no errors are reported in the console, then the design has successfully compiled!

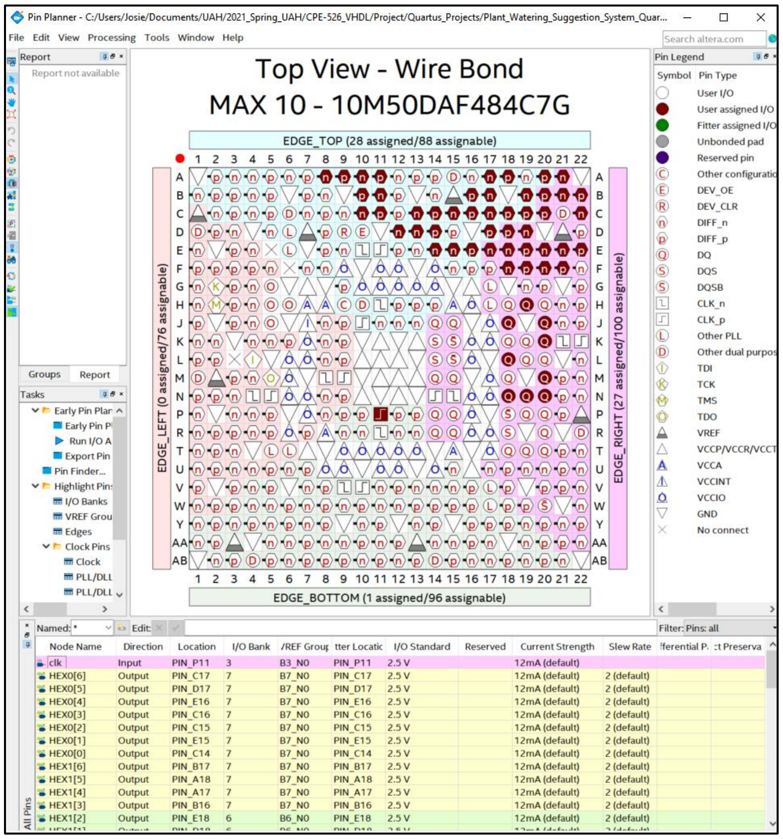
5.2. Pin Assignments

The next step is to assign the inputs and outputs of the top-level design to the correct pins. The pin assignments for each component are listed in the DE10-Lite user's manual and are detailed in the Appendix in Section 7.2. Quartus Prime's tool Assignment Editor and Pin Planner allow the user to assign pins with ease and confirms whether or not the pin assignment is valid. Both these utilities can be found be navigating to the Assignments tab.



	Status	From	To	Assignment Name	Value	Enabled
1	✓ Ok		SW[0]	Location	PIN_C10	Yes
2	✓ Ok		SW[1]	Location	PIN_C11	Yes
3	✓ Ok		SW[2]	Location	PIN_D12	Yes
4	✓ Ok		HEX0[0]	Location	PIN_C14	Yes
5	✓ Ok		HEX0[1]	Location	PIN_E15	Yes
6	✓ Ok		HEX0[2]	Location	PIN_C15	Yes
7	✓ Ok		HEX0[3]	Location	PIN_C16	Yes
8	✓ Ok		HEX0[4]	Location	PIN_E16	Yes
9	✓ Ok		HEX0[5]	Location	PIN_D17	Yes
10	✓ Ok		HEX0[6]	Location	PIN_C17	Yes
11	✓ Ok		HEX1[0]	Location	PIN_C18	Yes
12	✓ Ok		HEX1[1]	Location	PIN_D18	Yes
13	✓ Ok		HEX1[2]	Location	PIN_E18	Yes
14	✓ Ok		HEX1[3]	Location	PIN_B16	Yes
15	✓ Ok		HEX1[4]	Location	PIN_A17	Yes
16	✓ Ok		HEX1[5]	Location	PIN_A18	Yes
17	✓ Ok		HEX1[6]	Location	PIN_B17	Yes
18	✓ Ok		HEX2[0]	Location	PIN_B20	Yes
19	✓ Ok		HEX2[1]	Location	PIN_A20	Yes
20	✓ Ok		HEX2[2]	Location	PIN_B19	Yes
21	✓ Ok		HEX2[3]	Location	PIN_A21	Yes
22	✓ Ok		HEX2[4]	Location	PIN_B21	Yes
23	✓ Ok		HEX2[5]	Location	PIN_C22	Yes
24	✓ Ok		HEX2[6]	Location	PIN_B22	Yes
25	✓ Ok		HEX3[0]	Location	PIN_F21	Yes
26	✓ Ok		HEX3[1]	Location	PIN_E22	Yes
27	✓ Ok		HEX3[2]	Location	PIN_E21	Yes
28	✓ Ok		HEX3[3]	Location	PIN_C19	Yes
29	✓ Ok		HEX3[4]	Location	PIN_C20	Yes
30	✓ Ok		HEX3[5]	Location	PIN_D19	Yes
31	✓ Ok		HEX3[6]	Location	PIN_E17	Yes
32	✓ Ok		HEX4[0]	Location	PIN_F18	Yes
33	✓ Ok		HEX4[1]	Location	PIN_E20	Yes
34	✓ Ok		HEX4[2]	Location	PIN_E19	Yes
35	✓ Ok		HEX4[3]	Location	PIN_J18	Yes
36	✓ Ok		HEX4[4]	Location	PIN_H19	Yes
37	✓ Ok		HEX4[5]	Location	PIN_F19	Yes
38	✓ Ok		HEX4[6]	Location	PIN_F20	Yes
39	✓ Ok		HEX5[0]	Location	PIN_J20	Yes
40	✓ Ok		HEX5[1]	Location	PIN_K20	Yes
41	✓ Ok		HEX5[2]	Location	PIN_L18	Yes
42	✓ Ok		HEX5[3]	Location	PIN_N18	Yes
43	✓ Ok		HEX5[4]	Location	PIN_M20	Yes
44	✓ Ok		HEX5[5]	Location	PIN_N19	Yes
45	✓ Ok		HEX5[6]	Location	PIN_N20	Yes
46	✓ Ok		clk	Location	PIN_P11	Yes
47	✓ Ok		LEDX[0]	Location	PIN_A8	Yes
48	✓ Ok		LEDX[1]	Location	PIN_A9	Yes
49	✓ Ok		LEDX[2]	Location	PIN_A10	Yes
50	✓ Ok		LEDX[3]	Location	PIN_B10	Yes
51	✓ Ok		LEDX[4]	Location	PIN_D13	Yes
52	✓ Ok		LEDX[5]	Location	PIN_C13	Yes
53	✓ Ok		LEDX[6]	Location	PIN_E14	Yes
54	✓ Ok		LEDX[7]	Location	PIN_D14	Yes
55	✓ Ok		LEDX[8]	Location	PIN_A11	Yes
56	✓ Ok		LEDX[9]	Location	PIN_B11	Yes
57		<<new>>	<<new>>	<<new>>		

Figure 18: MAX10 Pin Assignments in Quartus Assignment Editor


Pin Planner - C:/Users/Josie/Documents/UAH/2021_Spring_UAH/CPE-526_VHDL/Project/Quartus_Projects/Plant_Watering_Suggestion_System_Quar...
File Edit View Processing Tools Window Help
Top View - Wire Bond
MAX 10 - 10M50DAF484C7G
EDGE_TOP (28 assigned/88 assignable)
EDGE_LEFT (0 assigned/76 assignable)
EDGE_RIGHT (27 assigned/100 assignable)
EDGE_BOTTOM (1 assigned/96 assignable)
Pin Legend
Named: *
Filter: Pins: all

Node Name	Direction	Location	I/O Bank	/REF Group	tter Locatic	I/O Standard	Reserved	Current Strength	Slew Rate	fferential P.	:t Preserva
clk	Input	PIN_P11	3	B3_N0	PIN_P11	2.5 V		12mA (default)			
HEX0[6]	Output	PIN_C17	7	B7_N0	PIN_C17	2.5 V		12mA (default)	2 (default)		
HEX0[5]	Output	PIN_D17	7	B7_N0	PIN_D17	2.5 V		12mA (default)	2 (default)		
HEX0[4]	Output	PIN_E16	7	B7_N0	PIN_E16	2.5 V		12mA (default)	2 (default)		
HEX0[3]	Output	PIN_C16	7	B7_N0	PIN_C16	2.5 V		12mA (default)	2 (default)		
HEX0[2]	Output	PIN_C15	7	B7_N0	PIN_C15	2.5 V		12mA (default)	2 (default)		
HEX0[1]	Output	PIN_E15	7	B7_N0	PIN_E15	2.5 V		12mA (default)	2 (default)		
HEX0[0]	Output	PIN_C14	7	B7_N0	PIN_C14	2.5 V		12mA (default)	2 (default)		
HEX1[6]	Output	PIN_B17	7	B7_N0	PIN_B17	2.5 V		12mA (default)	2 (default)		
HEX1[5]	Output	PIN_A18	7	B7_N0	PIN_A18	2.5 V		12mA (default)	2 (default)		
HEX1[4]	Output	PIN_A17	7	B7_N0	PIN_A17	2.5 V		12mA (default)	2 (default)		
HEX1[3]	Output	PIN_B16	7	B7_N0	PIN_B16	2.5 V		12mA (default)	2 (default)		
HEX1[2]	Output	PIN_E18	6	B6_N0	PIN_E18	2.5 V		12mA (default)	2 (default)		

Figure 19: Figure 18: MAX10 Pin Assignments in Quartus Pin Planner

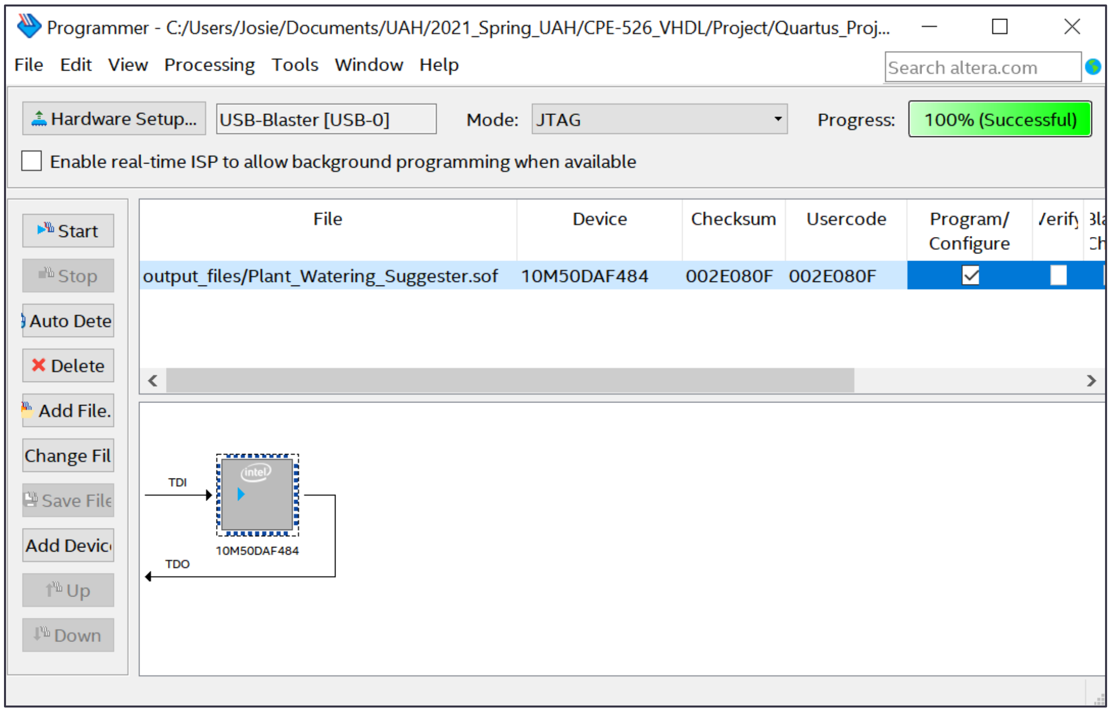
5.3. Programming the FPGA

The FPGA can be programmed in Quartus by using the Programmer tool. This is the most compilated step, as it is likely that upon the first usage of the USB Blaster, a new driver will have to be installed. If the following steps are followed and no USB blaster is detected, then these two links can help navigate the process of updating the computer's drivers:

https://community.intel.com/t5/Programmable-Devices/Windows-10-driver-support-for-USB-Blaster/m-p/55327#M14432

https://www.terasic.com.tw/wiki/Altera_USB_Blaster_Driver_Installation_Instructions

1. Go to the Tools tab and click on “Programmer”.
2. If the upper left box says “No Hardware”, click the “Hardware Setup” button to open up a pop-up window.
3. The USB Blaster should appear in the list of “Available hardware items” box.
4. If it does not appear, navigate to the links above to update the driver. Also check that the board is plugged in, etc.
5. If it does appear, click on it to add it, and exit the pop-up window.
6. Now USB Blaster should be viewed at the top of the Programmer window.
7. Click “Add File” and navigate to the .sof file that was generated by the successful compilation (typically in a folder named “output”).
8. Check the box for “Program/Configuration”.
9. The Progress bar on the upper right should fill up to be completely green, stating that the board logic has been successfully programmed.


Programmer - C:/Users/Josie/Documents/UAH/2021_Spring_UAH/CPE-526_VHDL/Project/Quartus_Proj...

USB-Blaster [USB-0] Mode: JTAG Progress: 100% (Successful)

☐ Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	/erify
output_files/Plant_Watering_Suggester.sof	10M50DAF484	002E080F	002E080F	✓	☐

Figure 20: Quartus Programmer Configuration

This completes the synthesis process.

6. Conclusion

In conclusion, the plant watering suggestion system functions! This can be considered a beta version, as it satisfies the requirements listed in Section 2.2 and demonstrates the capability to intelligently make water suggestions to the lay plant user. This is the first step towards mitigating the accelerated plant deathrate that the last year of inexperienced domestic botanists have incurred.

6.1. Lessons Learned

Four important lessons (among many, many others) have been derived from this development experience.

Lesson #1: In addition to synthesizing early and often, also simulate early and often so one can debug a wonky synthesized model, as well as pass the model on to other people to help with model verification.

Lesson #2: Clocking is IMPORTANT. This is not C++, so one should anticipate synthesized models to have more unpredictable behavior because they exist in the real-life physical realm.

Lesson #3: Configuring analog output took significantly more time than expected. This lesson is among many other indicators to start such a project as early as possible to account for possible unexpected learning curves and setbacks.

Lesson #4: Do not plug power into ground and ground into power. Especially do not do this when one is traveling for work 300 miles away from all the spare parts and extra equipment. Hilarity will not ensure.

6.2. Future Development

- Create a more robust physical testbench, including different types of soil, plants, and moisture levels.

- Use test results to fine tune and optimize watering suggestion calculation.
- Optimization the 7-Segment display (this involves improving clocking, perhaps by incorporating a clock divider).
- Complete simulation model to help debug wonky synthesis output and port model for other people to test.

7. Appendices

7.1. Data Sheets & Reference Links

DE10-Lite User Manual

https://www.manualslib.com/manual/1429115/Terasic-De10-Lite.html

Ks0049 Keyestudio Soil Humidity Sensor Wiki

https://wiki.keyestudio.com/Ks0049_keyestudio_Soil_Humidity_Sensor

Quartus Prime Lite Edition & Device Installation Center

https://fpgasoftware.intel.com/?edition=lite

Terasic DE10-Lite ADC Measurement Demo (download "CD")

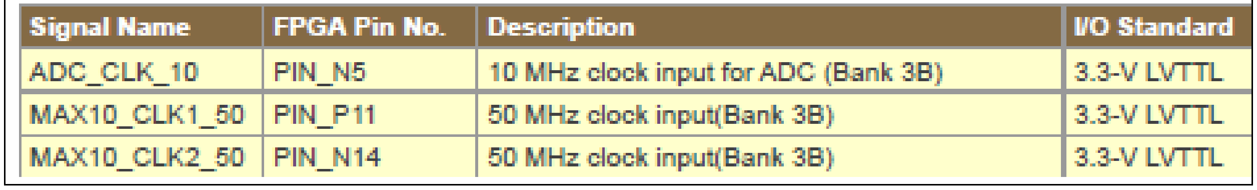
https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=234&No=1021&PartNo=4

Project GitHub Link

https://github.com/lyonthefrog/VHDL-DE10-Plant-Watering-Suggestion-System.git

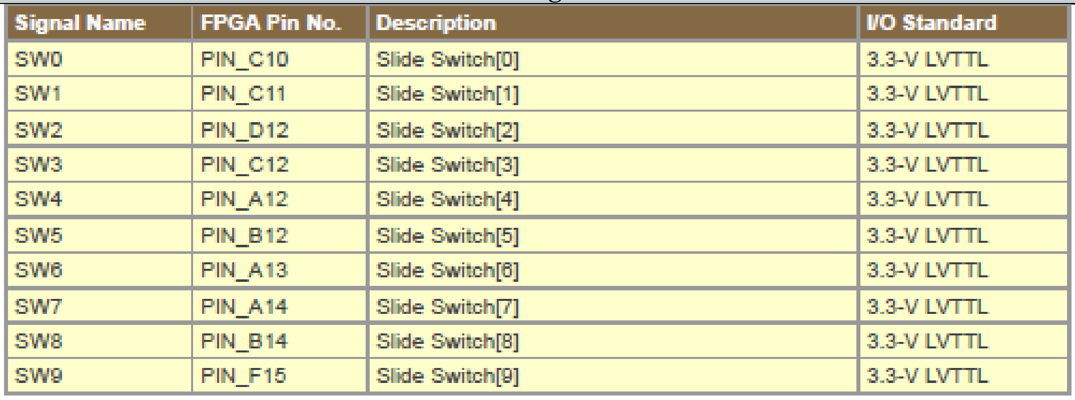
7.2. Pin Assignments

Table 8: DE10-Lite Pin Assignments of Clock Inputs



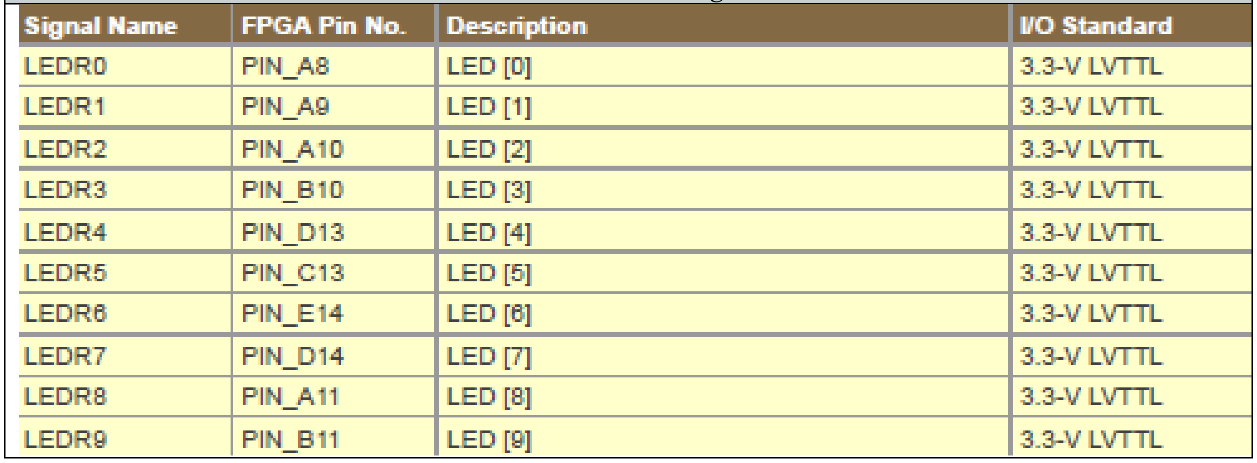
Signal Name	FPGA Pin No.	Description	I/O Standard
ADC_CLK_10	PIN_N5	10 MHz clock input for ADC (Bank 3B)	3.3-V LVTTL
MAX10_CLK1_50	PIN_P11	50 MHz clock input(Bank 3B)	3.3-V LVTTL
MAX10_CLK2_50	PIN_N14	50 MHz clock input(Bank 3B)	3.3-V LVTTL

Table 9: DE10-Lite Pin Assignments of Slide Switches



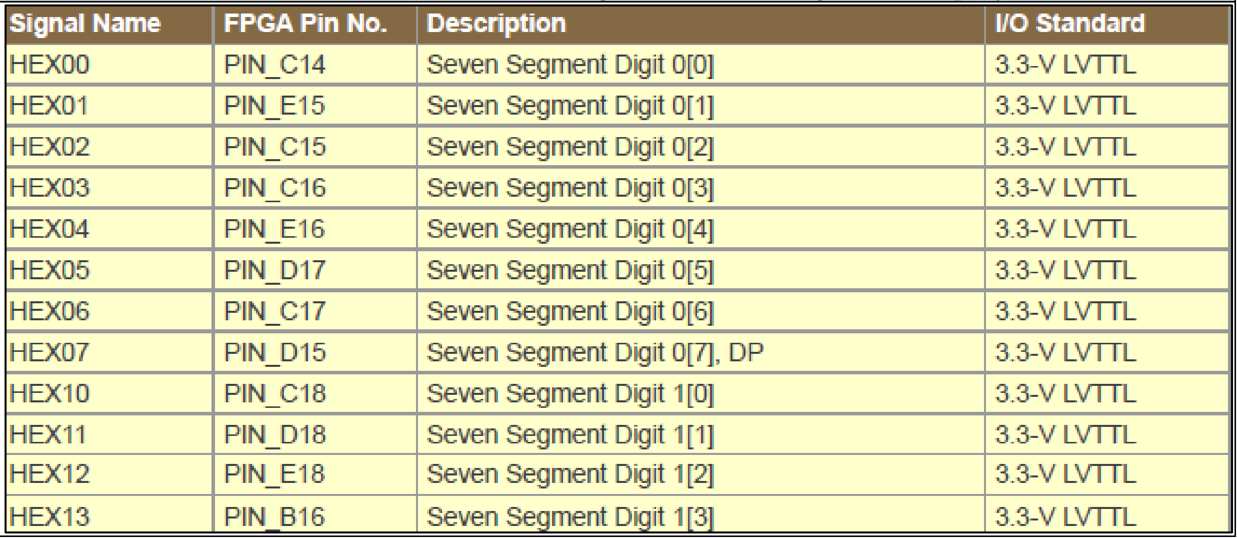
Signal Name	FPGA Pin No.	Description	I/O Standard
SW0	PIN_C10	Slide Switch[0]	3.3-V LVTTL
SW1	PIN_C11	Slide Switch[1]	3.3-V LVTTL
SW2	PIN_D12	Slide Switch[2]	3.3-V LVTTL
SW3	PIN_C12	Slide Switch[3]	3.3-V LVTTL
SW4	PIN_A12	Slide Switch[4]	3.3-V LVTTL
SW5	PIN_B12	Slide Switch[5]	3.3-V LVTTL
SW6	PIN_A13	Slide Switch[6]	3.3-V LVTTL
SW7	PIN_A14	Slide Switch[7]	3.3-V LVTTL
SW8	PIN_B14	Slide Switch[8]	3.3-V LVTTL
SW9	PIN_F15	Slide Switch[9]	3.3-V LVTTL

Table 10: DE10-Lite Pin Assignments of LEDs



Signal Name	FPGA Pin No.	Description	I/O Standard
LEDR0	PIN_A8	LED [0]	3.3-V LVTTL
LEDR1	PIN_A9	LED [1]	3.3-V LVTTL
LEDR2	PIN_A10	LED [2]	3.3-V LVTTL
LEDR3	PIN_B10	LED [3]	3.3-V LVTTL
LEDR4	PIN_D13	LED [4]	3.3-V LVTTL
LEDR5	PIN_C13	LED [5]	3.3-V LVTTL
LEDR6	PIN_E14	LED [6]	3.3-V LVTTL
LEDR7	PIN_D14	LED [7]	3.3-V LVTTL
LEDR8	PIN_A11	LED [8]	3.3-V LVTTL
LEDR9	PIN_B11	LED [9]	3.3-V LVTTL

Table 11: DE10-Lite Pin Assignments of 7-Segment Displays



Signal Name	FPGA Pin No.	Description	I/O Standard
HEX00	PIN_C14	Seven Segment Digit 0[0]	3.3-V LVTTL
HEX01	PIN_E15	Seven Segment Digit 0[1]	3.3-V LVTTL
HEX02	PIN_C15	Seven Segment Digit 0[2]	3.3-V LVTTL
HEX03	PIN_C16	Seven Segment Digit 0[3]	3.3-V LVTTL
HEX04	PIN_E16	Seven Segment Digit 0[4]	3.3-V LVTTL
HEX05	PIN_D17	Seven Segment Digit 0[5]	3.3-V LVTTL
HEX06	PIN_C17	Seven Segment Digit 0[6]	3.3-V LVTTL
HEX07	PIN_D15	Seven Segment Digit 0[7], DP	3.3-V LVTTL
HEX10	PIN_C18	Seven Segment Digit 1[0]	3.3-V LVTTL
HEX11	PIN_D18	Seven Segment Digit 1[1]	3.3-V LVTTL
HEX12	PIN_E18	Seven Segment Digit 1[2]	3.3-V LVTTL
HEX13	PIN_B16	Seven Segment Digit 1[3]	3.3-V LVTTL