Combinational Logic Design I—A Sprinkler System

You will learn how to translate a word description for a desired functional behavior into a precise specification using ABEL, how to use a few more features of the logic simulator, and to how to interpret the implementation reports.

1.0 Overview

In this experiment you are asked to specify a design using the ABEL Hardware Description Language (HDL). Even though the circuit you will be creating is simple enough to describe using the Schematic Editor, use of ABEL at this stage will help become better prepared for the more complex circuits we'll be designing in future experiments.

2.0 Preparation

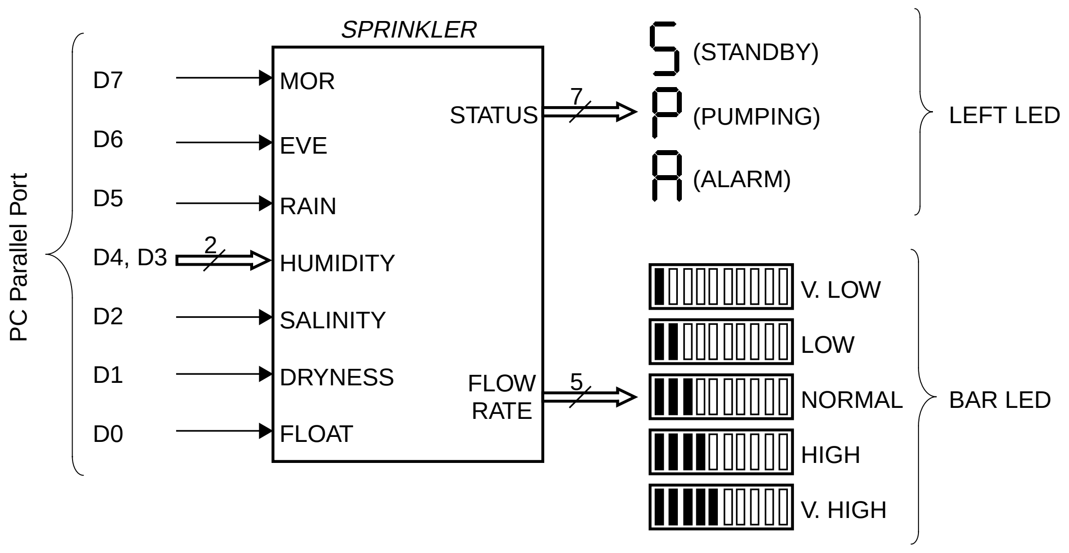
- Read Sec. 4.6 of Wakerly and refer to the online *Xilinx ABEL Reference Manual* available through the Help menu of the Foundation Project Manager.
- Read Sec. 4.3 of Wakerly on combinational circuit synthesis. Even though you do not need to perform logic minimization when describing your design using ABEL, knowing what two-level logic minimization involves will help you understand the output of the Foundation synthesis tools.

3.0 Design Specification

The top-level schematic of the *SPRINKLER* circuit you need to design is shown in Figure 4. It has the following inputs:

- MOR: a 1-bit signal that becomes 1 between 8:00 and 8:30 AM and is 0 otherwise.
- EVE: a 1-bit signal that becomes 1 between 6:00 and 6:30 PM and is 0 otherwise.
- RAIN: a 1-bit signal that is 1 when it is raining and 0 otherwise.

FIGURE 4. Top-level schematic of *SPRINKLER*



- HUMIDITY: a 2-bit signal that indicates the level of humidity in the air according to the following encoding:

00 :	Very humid
01 :	Humid
10 :	Dry
11 :	Very dry

- SALINITY: a 1-bit signal that becomes 0 when the soil salinity level becomes too high and is 1 otherwise.
- DRYNESS: a 1-bit signal that becomes 1 when the soil is too dry and is 0 otherwise.
- FLOAT: a 1-bit signal that becomes 0 when the water level in the reservoir supplying the sprinklers is too low ad is 1 otherwise.

These inputs should be connected to the data bits of the PC parallel port as indicated in Figure 4.

SPRINKLER has three mutually-exclusive operation modes—STANDBY, PUMPING, and ALARM—defined as follows:

- STANDBY: the normal "off" condition; water is not flowing.
- PUMPING: the normal "on" condition; water is flowing.
- ALARM: the abnormal "off" condition; water should be flowing but is not because its level in the reservoir is too low.

These modes should be indicated with the letters S, P, and A on the LEFT LED as shown in Figure 4.

The normal “on” condition should occur twice a day (when either MOR or EVE become asserted) unless it is raining. The “on” condition is also activated whenever the soil salinity *and* dryness levels become too high. When in the PUMPING mode, the water flow rate depends on the air humidity and soil conditions according to:

Soil	Air	Flow Rate
Dry and saline	—	Very High
Neither dry nor saline	Very dry	High
Neither dry nor saline	Dry	Normal
Neither dry nor saline	Humid	Low
Neither dry nor saline	Very humid	Very low

These five flow rates should be indicated using the BAR LED as shown in Figure 4.

4.0 Design Notes and Hints

- Before you start coding your design in ABEL, examine the specifications to make sure you understand them. Pay particular attention to any ambiguities and be prepared to document how you handled them.
- Even though the input/output declarations in an ABEL program can optionally include specific pin numbers on the target chip, it is best to put such device-specific information in the User Constraint File (UCF) of your project. In fact, for our design it is not even possible to specify pin numbers for two of the inputs: you may recall that parallel port bits D6 and D7 correspond to the special mode pins MD0 and MD2 on the Xilinx FPGA.

  The solution to this problem is to create a *macro* that implements the functionality of *SPRINKLER* and to instantiate it, along with instances of MD0, MD2, and other necessary buffers, inverters, and I/O pads, using the Schematic Editor. A macro can be generated by clicking **Create Macro** under the **Project** menu of the HDL editor.
- Keep the name of your ABEL module short: eight or fewer characters! I used *SPRNKLER* (no I) after consulting the Xilinx web Answer Database to decipher the cryptic error message I got from the compiler. This is an MS-DOS limitation.
- To reduce the tedium and minimize errors in creating the UCF (for this and all future experiments), it might be a good idea to create a “universal” UCF that captures all the pin bindings (names and locations) in Table 1 on page 7 of the *EECS Laboratory Overview* handout and save it somewhere (I called mine XESS.ucf). When you’re ready to enter the constraints for a particular project, copy the contents of this universal UCF to your project’s UCF and delete all lines corresponding to pins that you do not use. This can save a lot of typing and reduce the chance of errors. It also means that you should label your design’s I/O nets using the names in this “universal” UCF (i.e. the names in the above-mentioned Table 1.)

5.0 Deliverables

5.1 Pre-Lab

1. Hardcopy of *SPRINKLER*'s schematic, ABEL, and UCF files.
2. Hardcopy of simulation results for the two scenarios shown in the following table:

Scenario A		**Scenario B**	
Signal	**Stimulator**	**Signal**	**Stimulator**
MOR	1	MOR	0
EVE	0	EVE	0
RAIN	0	RAIN	0
DRYNESS	0	FLOAT	1
SALINITY	1	DRYNESS	B0
FLOAT	B0	SALINITY	B1
HUMIDITY0	B1	HUMIDITY0	B2
HUMIDITY1	B2	HUMIDITY1	B3

Display simulation traces showing the waveforms on the STATUS and FLOWRATE outputs for all possible combinations of the applied counter stimulators: 8 for scenario A and 16 for scenario B. To make the simulation results easier to read use the bus display mode to bundle some signals (and show their values in hex) as follows:

Scenario A	**Scenario B**
HUMIDITY1..HUMIDITY0	SALINITY..DRYNESS
STATUS6..STATUS0	HUMIDITY1..HUMIDITY0
FLOWRATE5..FLOWRATE1	STATUS6..STATUS0
	FLOWRATE5..FLOWRATE1

5.2 In-Lab

Generate the .bit file for *SPRINKLER*, download it to the XESS board, and verify that your implementation is working properly. When you are satisfied that you have a correctly-functioning circuit, demonstrate its operation to your lab GSI and have him sign your experiment's cover sheet.

5.3 Post-Lab

Prepare your lab report as described in the *EECS270 Laboratory Overview* handout. Make sure you complete and include all parts of the report including the *Cover Sheet*, the *Design Narrative* section, and the *Design Documentation* section. In the *Post-Lab Questions* section, provide answers to the following questions:

1. Generate the minimum two-level forms of the output equations for your ABEL macro. How many product terms are needed to implement it? For extra credit, locate (and include) the file containing the synthesized equations for the macro and verify that they are indeed minimal.
2. How would you change your design to allow for a manual override feature that enables you to control the water flow rate (from no flow to very high flow) regardless of the other inputs to the system?
3. To increase system reliability, suppose that another pump is added to the reservoir. Can you design a combinational circuit that will alternate between these two pumps whenever the PUMPING mode is indicated (i.e. the first time PUMPING is true use pump 1, the second time use pump 2, the third time use pump1, and so on.)? If your answer is affirmative, provide the equations necessary to do the job; if negative, give a clear justification of why you think it cannot be done.

This page is intentionally blank!