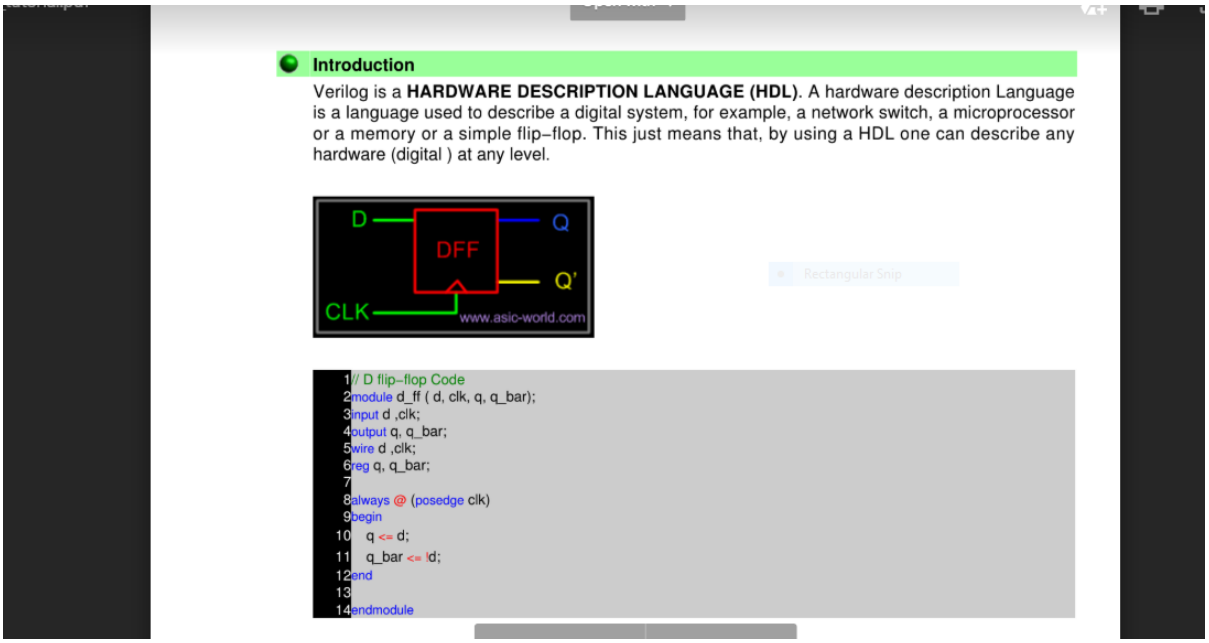
DAILY ASSESSMENT

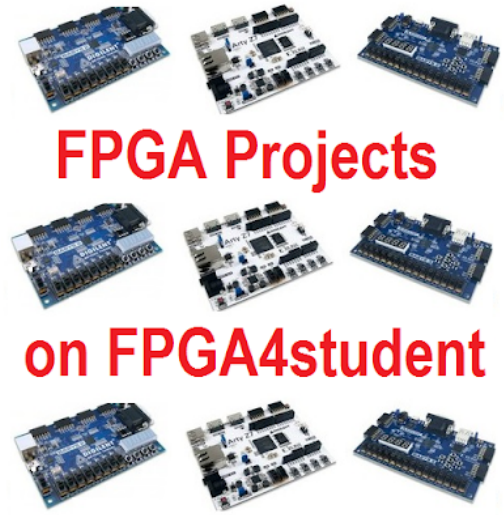
Date:	05-06-2020	Name:	POOJA K S
Course:	DIGITAL DESIGN USING HDL	USN:	4AL17EC070
Topic:	Verilog Tutorials and practice programs & Building/ Demo projects using FPGA	Semester & Section:	6TH SEM & 'B' Section
Github Repository:	pooja-shivanna		

FORENOON SESSION DETAILS


Edit with WPS Office

FPGA Projects

This page presents FPGA projects on fpga4student.com. The first FPGA project helps students understand the basics of FPGAs and how Verilog/ VHDL works on FPGA.



Some of the FPGA projects can be FPGA tutorials such as What is FPGA Programming, image processing on FPGA, matrix multiplication on FPGA Xilinx using Core Generator, Verilog vs VHDL: Explain by Examples and how to load text files or images into FPGA. Many others FPGA projects

Report

Introduction

Verilog is a HARDWARE DESCRIPTION LANGUAGE (HDL). A hardware description Language is a language used to describe a digital system, for example, a network switch, a microprocessor or a memory or a simple flip−flop. This just means that, by using a HDL one can describe any hardware (digital) at any level.

```
// D flip−flop Code
module d_ff ( d, clk, q, q_bar);
input d ,clk;
output q, q_bar;
```

Edit with WPS Office

```
wire d ,clk;

reg q, q_bar;

always @ (posedge clk)

begin

 q <= d;

 q_bar <= !d;

end

endmodule
```

One can describe a simple Flip flop as that in above figure as well as one can describe a complicated designs having 1 million gates. Verilog is one of the HDL languages available in the industry for designing the Hardware. Verilog allows us to design a Digital design at Behavior Level,Register Transfer Level (RTL), Gate level and at switch level. Verilog allows hardware designers to express their designs with behavioral constructs, deterring the details of implementation to a later stage of design in the final design.

Many engineers who want to learn Verilog, most often ask this question, how much time it will take to learn Verilog?, Well my answer to them is "It may not take more then one week, if you happen to know at least one programming language".

Design Styles Verilog like any other hardware description language, permits the designers to design a design in either Bottom-up or Top-down methodology.

Bottom-Up Design

The traditional method of electronic design is bottom-up. Each design is performed at the gate-level using the standard gates (Refer to the Digital Section for more details) With increasing complexity of new designs this approach is nearly impossible to maintain. New systems consist of ASIC or microprocessors with a complexity of thousands of transistors. These traditional bottom-up designs have to give way to new structural, hierarchical design methods. Without these new design practices it would be impossible to handle the new complexity.

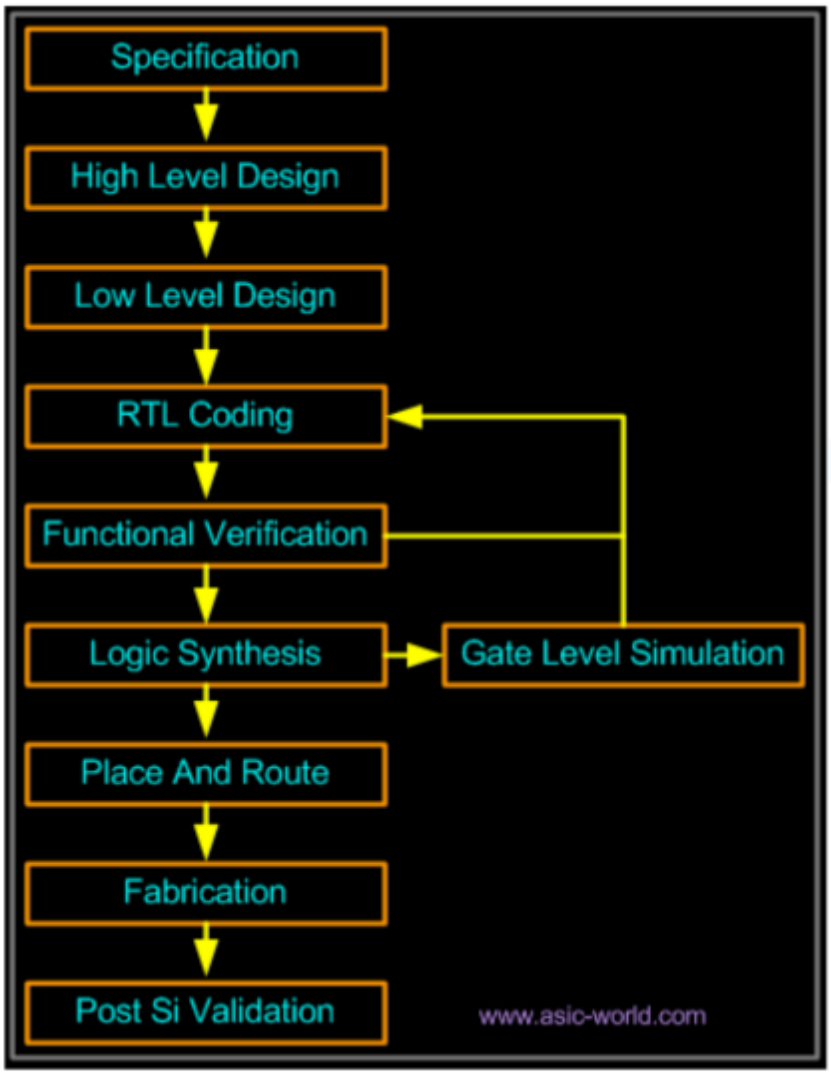
Top-Down Design

The desired design-style of all designers is the top-down design. A real top-down design allows early testing, easy change of different technologies, a structured system design and offers many other advantages. But it is very

Edit with WPS Office

difficult to follow a pure top−down design. Due to this fact most designs are mix of both the methods, implementing some key elements of both design styles.

Figure shows a Top−Down design approach.



Abstraction Levels of Verilog

Verilog supports a design at many different levels of abstraction. Three of them are very important:

• Behavioral level

• Register−Transfer Level

• Gate Level

Behavioral level

This level describes a system by concurrent algorithms (Behavioral). Each algorithm itself is sequential, that means it consists of a set of instructions that

Edit with WPS Office

are executed one after the other. Functions, Tasks and Always blocks are the main elements. There is no regard to the structural realization of the design.

Register–Transfer Level Designs using the Register–Transfer Level specify the characteristics of a circuit by operations and the transfer of data between the registers. An explicit clock is used. RTL design contains exact timing possibility, operations are scheduled to occur at certain times. Modern definition of a RTL

code is "Any code that is synthesizable is called RTL code".

Gate Level

Within the logic level the characteristics of a system are described by logical links and their timing properties. All signals are discrete signals. They can only have definite logical values (`0', `1', `X',`Z`). The usable operations are predefined logic primitives (AND, OR, NOT etc gates). Using gate level modeling might not be a good idea for any level of logic design. Gate level code is generated by tools like synthesis tools and this netlist is used for gate level simulation and for backend.

Data Types

Verilog Language has two primary data types

• Nets – represents structural connections between components.

• Registers – represent variables used to store data.

Every signal has a data type associated with it:

• Explicitly declared with a declaration in your Verilog code.

Implicitly declared with no declaration but used to connect structural building blocks in your code.

•Implicit declaration is always a net type "wire" and is one bit wide.

Types of Nets

Each net type has functionality that is used to model different types of hardware (such as PMOS,NMOS, CMOS, etc)

Edit with WPS Office

Net Data Type	Functionality
wire, tri	Interconnecting wire – no special resolution function
wor, trior	Wired outputs OR together (models ECL)
wand,triand	Wired outputs AND together (models open–collector)
tri0,tri1	Net pulls–down or pulls–up when not driven
supply0,suppy1	Net has a constant logic 0 or logic 1 (supply strength)
trireg	

Register Data Types

Registers store the last value assigned to them until another assignment statement

changes their value.

• Registers represent data storage constructs.

• You can create arrays of the regs called memories.

• register data types are used as variables in procedural blocks.

• A register data type is required if a signal is assigned a value within a procedural block

• Procedural blocks begin with keyword initial and always.

Data Types	Functionality
reg	Unsigned variable
integer	Signed variable – 32 bits
time	Unsigned integer – 64 bits
real	Double precision floating point variable

Some of the FPGA projects can be FPGA tutorials such as What is FPGA Programming, image processing on FPGA, matrix multiplication on FPGA Xilinx using Core Generator, Verilog vs VHDL: Explain by Examples and how to load text files or images into FPGA. Many others FPGA projects provide students with full Verilog/ VHDL source code to practice and run on FPGA boards. Some of them can be used for another bigger FPGA projects.

Task

Edit with WPS Office

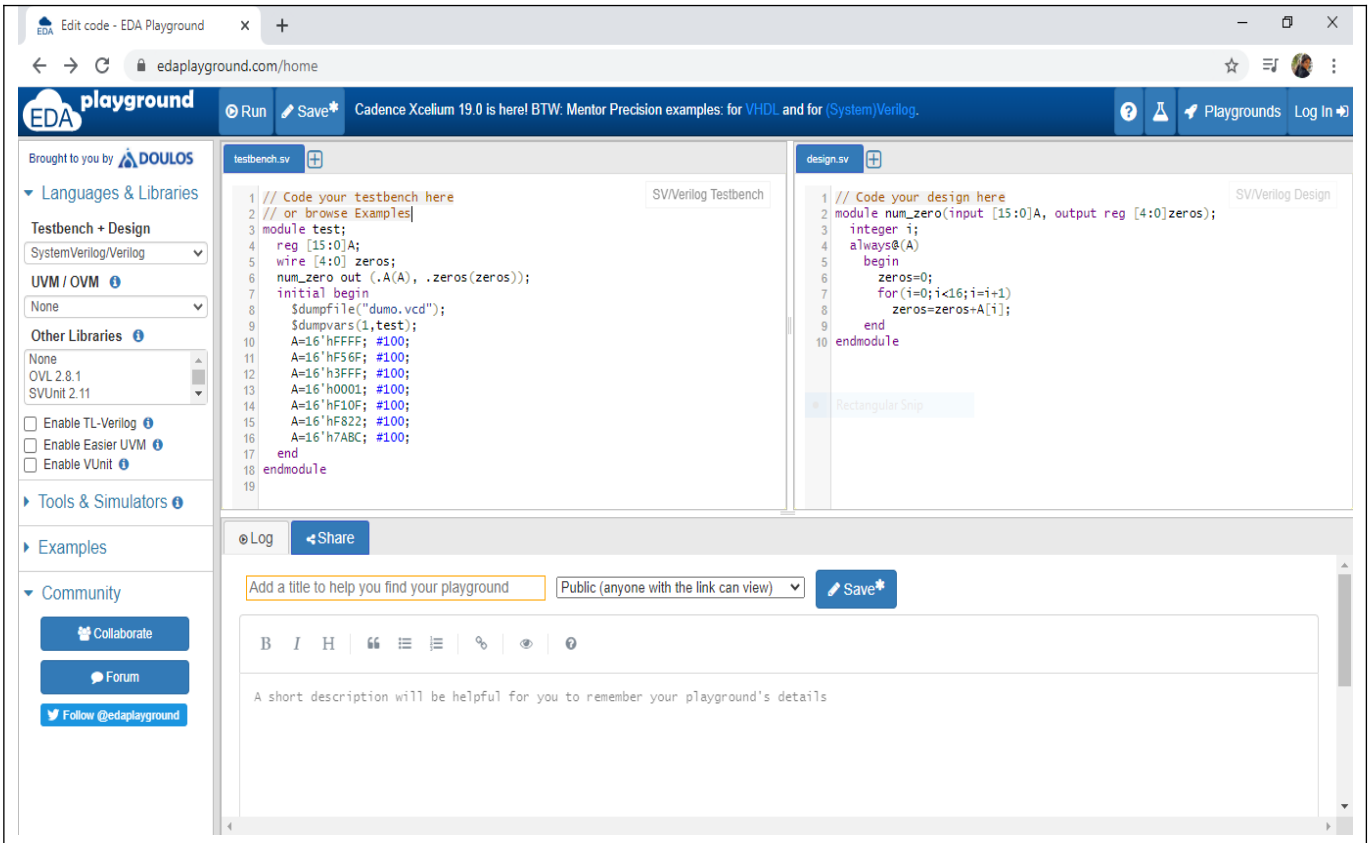
code

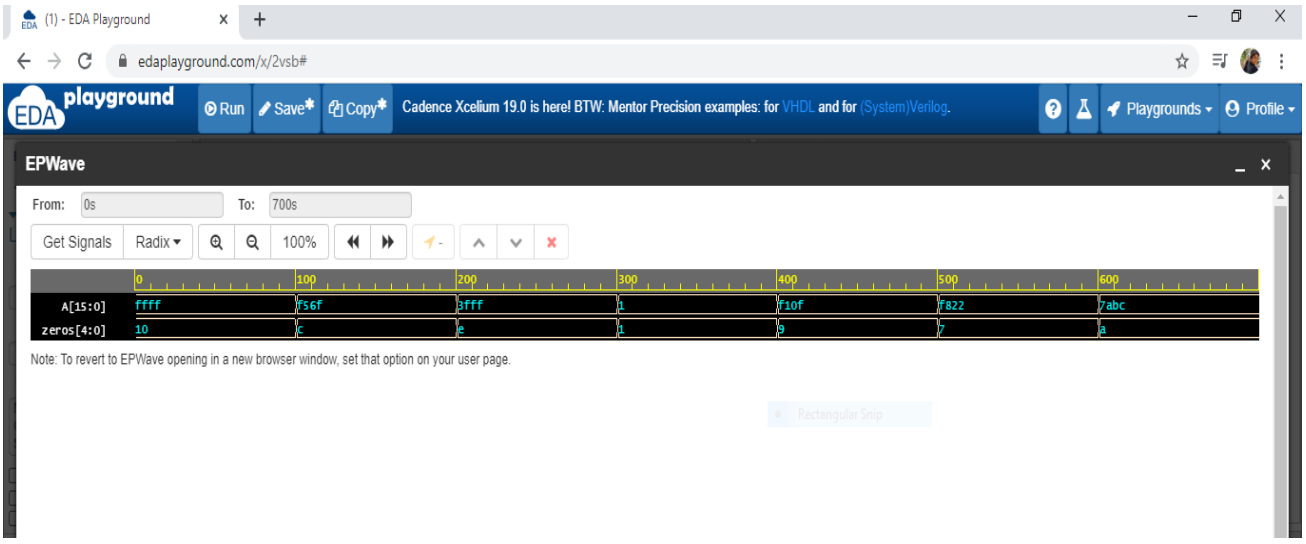
```
module num_zero(input [15:0]A, output reg [4:0]zeros);
  integer i;
  always@(A)
    begin
      zeros=0;
      for(i=0;i<16;i=i+1)
        zeros=zeros+A[i];
    end
endmodule
```

test bench code

```
module test;
  reg [15:0]A;
  wire [4:0] zeros;
  num_zero out (.A(A), .zeros(zeros));
  initial begin
    $dumpfile("dumo.vcd");
    $dumpvars(1,test);
    A=16'hFFFF; #100;
    A=16'hF56F; #100;
    A=16'h3FFF; #100;
    A=16'h0001; #100;
    A=16'hF10F; #100;
    A=16'hF822; #100;
    A=16'h7ABC; #100;
  end
endmodule
```

Edit with WPS Office



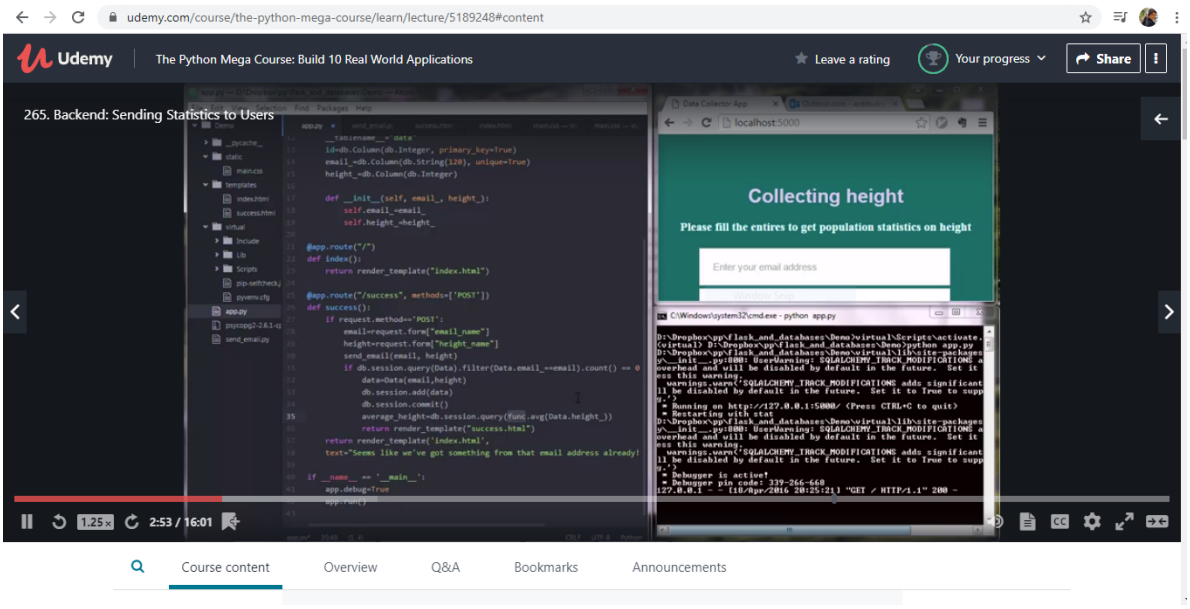


Edit with WPS Office

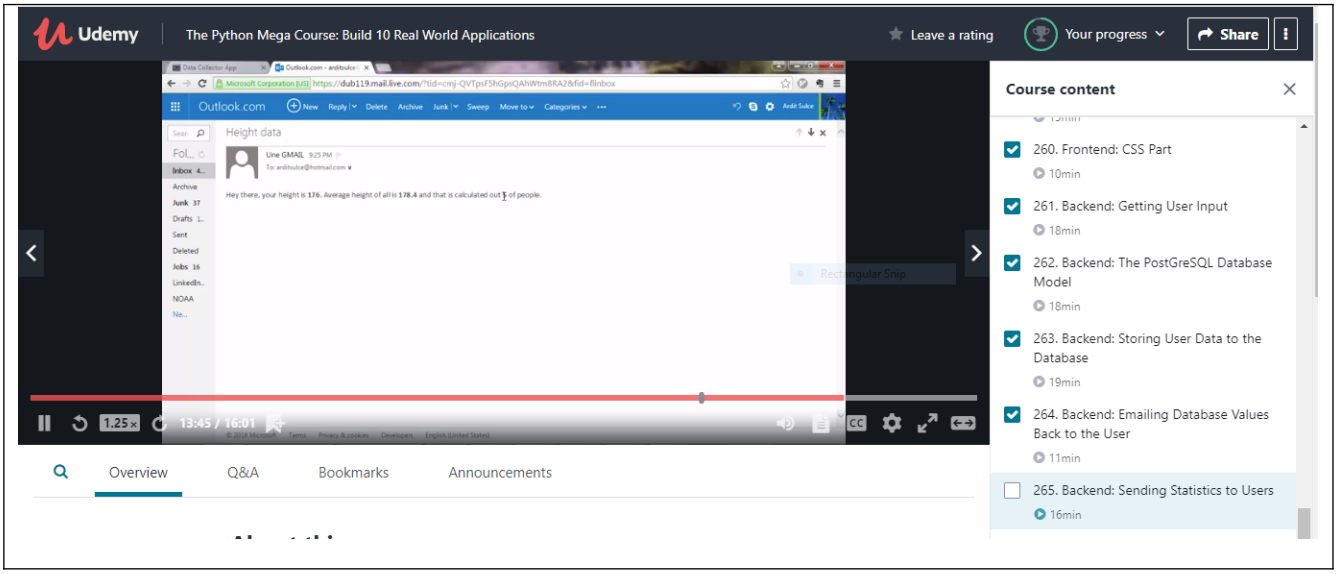
DAILY ASSESSMENT

Date:	05-06-2020	Name:	POOJA K S
Course:	PYTHON	USN:	4AL17EC079
Topic:	Application 9: Build a Data Collector Web App with PostGreSQL and Flask	Semester & Section:	6TH SEM & 'B' Section

AFTERNOON SESSION DETAILS



Edit with WPS Office



Report-

Flask startup and configuration Like most widely used Python libraries, the Flask package is installable from the Python Package Index (PPI). First create a directory to work in (something like flask_todo is a fine directory name) then install the flask package. You'll also want to install flask-sqlalchemy so your Flask application has a simple way to talk to a SQL database.A good way to get moving is to turn the codebase into an installable Python distribution. At the project's root, create setup.py and a directory called todo to hold the source code. The setup.py should look something like this:

```
 requires = [
                'flask',
               'flask-sqlalchemy',
                'psycopg2',
             ]
setup(
        name='flask_todo',
        version='0.0',
        description='A To-Do List built with Flask',
        author='<Your actual name here>',
        author_email='<Your actual e-mail address here>',
```

Edit with WPS Office

```
        keywords='web flask',
        packages=find_packages(),
        include_package_data=True,
        install_requires=requires
    )
```

This way, whenever you want to install or deploy your project, you'll have all the necessary packages in the requires list. You'll also have everything you need to set up and install the package in sitepackages. For more information on how to write an installable Python distribution, check out the docs on setup.py.Within the todo directory containing your source code, create an app.py file and a blank __init__.py file. The __init__.py file allows you to import from todo as if it were an installed package. The app.py file will be the application's root. This is where all the Flask application goodness will go, and you'll create an environment variable that points to that file. If you're using pipenv (like I am), you can locate your virtual environment with pipenv --venv and set up that environment variable in your environment's activate script.

Edit with WPS Office