SnapEDA Take Home Challenge

Goal

The goal of this exercise is to modify a parsing script to fix 3 issues. Follow the instructions below to setup and fix these issues.

Software and setup needed

- Python (https://www.python.org/downloads/)
- Eagle PCB (http://www.autodesk.com/products/eagle/free-download)

Instructions

The script you'll be fixing will parse .bsdl files (a type of descriptive language that describes the properties of a PCB component) and outputs an .lbr file (a file that can be opened with the Eagle PCB software to view the component's CAD model)

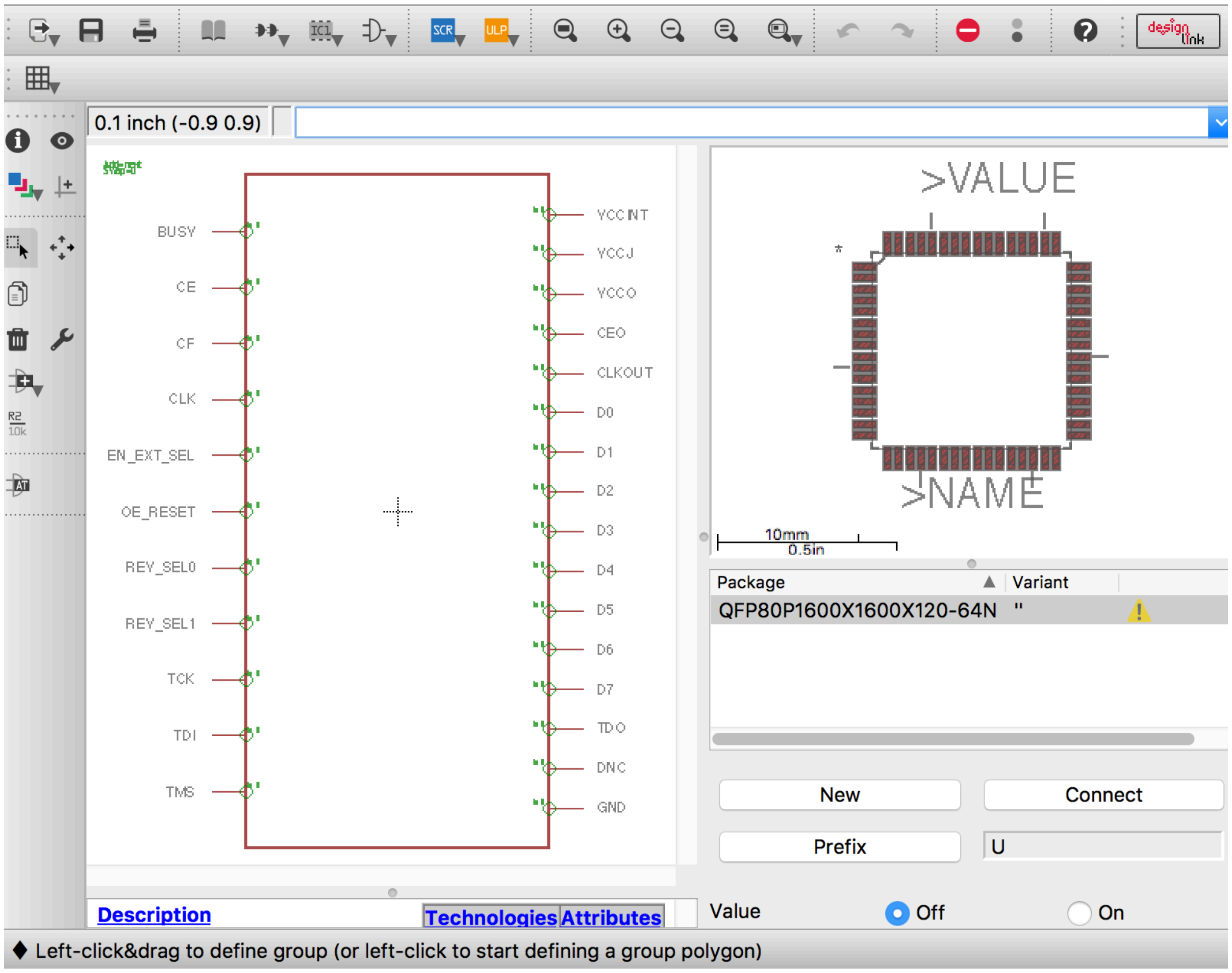
Unzip the BSDL Parser.zip, you'll find the following files:

- **parse_bsdl.py** – the parsing script that can parse a .bsdl files in the 'bsdl_files' folder and outputs the .lbr file
- **create_xml.py** – used by parse_bsdl.py to generate the output
- **TemplateEagleWithPackage.lbr** – used by parse_bsdl.py as a template to generate the output
- **Bsdl_files folder** – contains a 2 .bsdl files used for your testing

Try running the script by going to terminal (or command prompt), navigate to the BSDL Parse folder and enter:

```
python parse_bsdl.py bsdl_files/XCF32P.bsdl
```

This will parse the XCF32P.bsdl and output XCF32P.lbr in the 'bsdl_files' folder. If you have Eagle installed, you can double-click the .lbr file to open it in Eagle (sometimes, you might need to open it twice). You should see XCF32P under the Device column, double-click to open it and you should see the following:



Package	Variant
QFP80P1600X1600X120-64N	''

Prefix: U

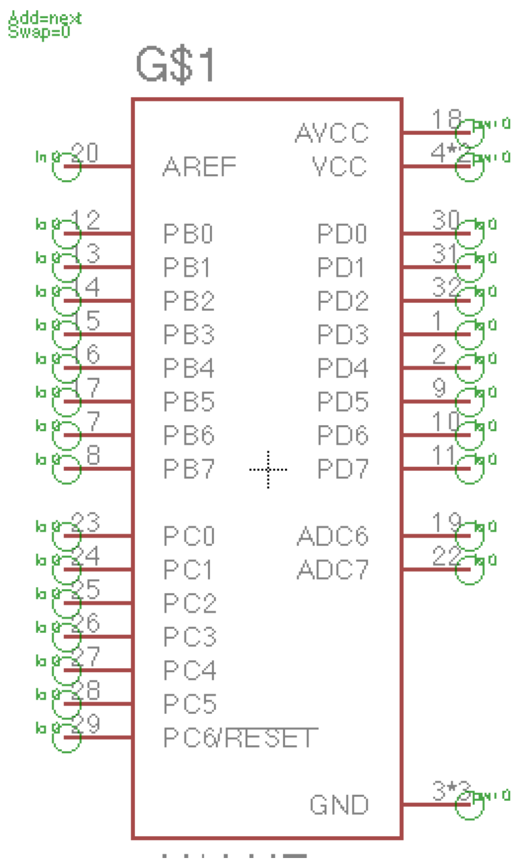
Value [✓ Off] On

Left-click&drag to define group (or left-click to start defining a group polygon)

Modify the parser script to fix the following 3 issues:

1. **Make the pin names appear on the inside of the box.**

   When you open the .lbr file in Eagle, you'll see a box with pins on either side (this is the symbol of the component). The pin names are on the outside of the box, modify the parser so that the pins are on the inside of the box. Here is an example of another symbol with the pin names on the inside of the box:



Note: you can also open the .lbr file using a text-editor to see the markup language describing the file (it uses an XML format)

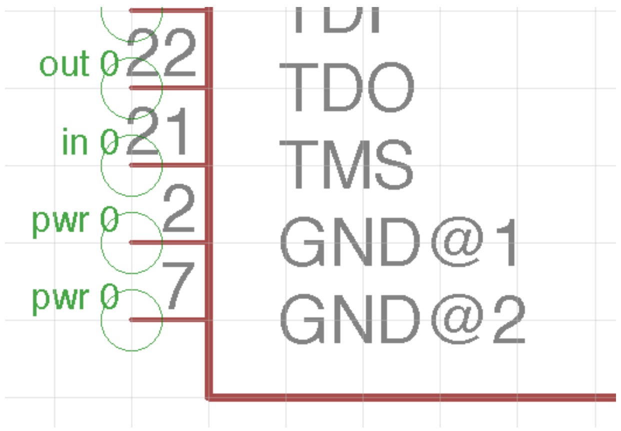
2. **Make the pin type label more descriptive ('in', 'out', 'pwr')**

Currently, you'll see a green circle with 'io' labeled next to each pin. Modify the script so that the label will say:

'in' – for input pins
'out' – for output pins
'io' – for input and output (inout) pins
'pwr' – for VCC and GND pins

If you open the XCF32P.bsdl file in a text editor, you'll see the types of pins described on lines 47 – 80. You can parse this information to generate the correct pin label.

Here is an example of a symbol with correct labels:



3. **Fix the parser to parse multiple pin mappings per line**

   Try parsing the other bsdl file:

   ```
   python parse_bsdl.py bsdl_files/XC2C64A.bsdl
   ```

   You'll get an error. This is because the parser can't handle multiple pin mappings per line (**lines 134-147** of XC2C64A.bsdl). In comparison, the previous bsdl had only 1 mapping per line (**lines 92 – 118** of XCF32P.bsdl)

   Modify the parser so that it can parse pin mapping description in XC2C64A.bsdl file.