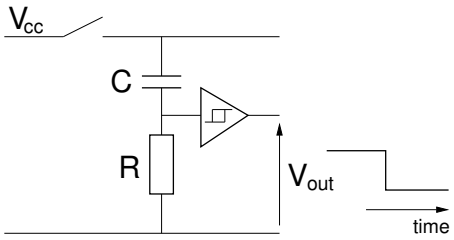
Maintenance

Having been shown the working logic simulator, the client has decided that they would like some modifications made to the system. Note that two of these concern the GUI, so you will need to rapidly redeploy a member of your scanner/parser team onto the GUI team.

• D-type bistables can be set to a predefined state on power-up by connecting their SET or CLEAR inputs to a R-C circuit as follows:



When the power switch is closed, the capacitor is initially uncharged and the voltage at the input to the Schmitt trigger is high. After a while (how long depends on the R-C time constant), the capacitor is sufficiently charged to send the Schmitt trigger output low. If V_{out} is connected to a D-type SET input, Q will be forced high on power-up. Likewise, if V_{out} is connected to a D-type CLEAR input, Q will be forced low on power-up. Implement a new RC device, with the following specification:

`RC`	1 output 0 inputs
Function:	On power-up, output starts high but falls low after n simulation cycles, where n is specified in the definition file.

• One important aspect of software design is *internationalization*. Modify your program so that the GUI supports English and one other language of your choice. You can either select this language via your desktop's **Account Settings** (this will affect all applications), or run the simulator by typing (e.g. for Spanish)

```
LANG=es_ES.utf8 ./logsim.py [filename]
```

which will affect only `logsim`. You will need to read about how wxPython handles internationalization through `wx.GetTranslation` and `wx.Locale`. Do not worry if your team does not have much language expertise. Just pick a language you vaguely recollect from school and use a dictionary. You will not be judged on the quality of your translations! Whichever language you choose, please also get your GUI to display some (arbitrary) non-Latin characters, e.g. Cyrillic, Arabic, Greek, Chinese.

• The client finds the 2D traces a little boring and asks you to implement a 3D display. You find the idea utterly ridiculous, but you are keen to brush up on your OpenGL skills, so you decide to humour the client, who is paying after all! You ask the client to clarify what sort of display they had in mind, and they provide a skeleton implementation in the file `gui_3D.py`, which you can import into the original, supplied code in place of `gui.py`. Add a facility to switch between conventional 2D and over-elaborate 3D signal traces. Do your best to make the 3D traces as useful as possible.

Implement these modifications and retest the system. At the same time, feel free to make any further enhancements of your own.

Andrew Gee
May 2024