z80qtSim is a version of **Z80Simulator** built on the Qt (widgets) framework, with additional features that could help reverse engineer and understand this chip better.

Quick Start

Described is a tested setup. Other setups may or may not work (i.e. not using MSVC 2015 but using other compilers; running on Linux, etc.) However, Qt is a very flexible and cross-platform, robust framework, and it makes a project very likely to "simply work" with different "kits" and platforms.

Install MS Visual Studio 2015.

Install Qt framework with a support for x64 MS Visual Studio 2015.

Tested with a Qt version 5.14

Compile under QtCreator, Release build.

When you run it, it will ask you to "Select chip resource folder". Find this folder under the "external" subdirectory of this git repo. Pick any PNG image in that folder to confirm the folder selection.

Hint: Maximize the main app for the best view

The application loads a number of Z80 images representing all chip layers. It also creates monochrome versions of those images. Then, it loads Z80 netlist and definitions of pads, signals and transistors.

On the top-left corner is a map of all images that are available to look.

Experiment 1

Using the mouse, pan and zoom (using the wheel) onto layer regions.

Press a key (according to the map) to see that particular layer.

Hold ALT key while pressing image selection to overlay more images on top. For example: press "3" to show "metal" traces and then press ALT+5 to overlay "vias" that are connections from metal traces to other layers. This way you can look at layers in an x-ray like manner.

Experiment 2

For this experiment, we will load transistor and signal definition. Click on the "Traces" button (to the bottom of the map. This will load data from the JavaScript resources (*.js) and draw them over the current image. **Bug:** This drawing is permanent at the moment, will be fixed soon.

Once that data is loaded, the sim will show you more information as you move your mouse over the chip features. The layer, transistor number and/or a signal number will be displayed in the map area. **Bug**: There is a large green triangle stretched over the left part of the image. It is a consequence of incompatible JavaScript data; ignore it.

Transistors starts with "t".

Experiment 3

This experiment is of a different nature altogether since it does not currently use the image viewing. The netlist-level chip simulator is built into the app and it can run a simple Z80 test program.

Make sure the Application Log windows is visible and select the menu “Simulation”->”Run”.

The sim will start executing and writing out lines of text containing some of the internal states (flops, traces) directly read from the netlist emulation. The simulation will stop after some time or you can stop it by selecting the menu “Simulation”->”Stop”.

That’s it for now: it is a work in progress with several different pieces slowly getting together.

Notes:

Application settings (windows positions, sizes, …) are stored in the Windows registry at this path:

HKEY_CURRENT_USER\Software\Baltazar Studios, LLC\Z80qtSim

References:

Z80Simulator: https://github.com/gdevic/Z80Simulator

This is a forked project. My additions to it, in the “private” branch, are:

1. Making it to compile with MS Visual Studio 2015 on Windows 10
2. Adding data serialization to the transistor netlist so that the built netlist could be saved and used in z80qtSim.

Visual6502: https://github.com/gdevic/visual6502

This is a forked project. The original runs the website http://www.visual6502.org/JSSim/expert-z80.html where you can actually run Z80.

My addition to it, in the “dev” branch” is:

1. Adding zoom using the mouse button