SUMMARY ON NETLIST SIMULATOR

NGUYEN DOAN DAI

This memo attempt to summarise my implementation of netlist simulator.

1. Compilation and usage

Compilation requires build tool ocamlbuild of version 0.14.0 or higher. The simulator is implemented in OCaml version 5.1.0. The implementation was compiled and tested in Linux Ubuntu 20.04.6 LTS through Window Subsystem for Linux. The source can be found on `https://github.com/enbugging/SystemeNumeriqueENS`.

To compile, use command `ocamlbuild netlist_simulator.byte`.

The execution has basic interface, of the form

`netlist_simulator.byte [-print] [-n nr] netfile.net`,

with

- `-print`: flag to print only the netlist after scheduling;
- `-n nr`: flag to specify `nr`, the number of cycles to be simulated;
- `netfile.net`, the `.net` file containing the description of circuit in netlist. For further information, consult the documentation provided with the package, title `minijazz.pdf`.

2. Functionalities

As required, it can simulate circuits described in netlist language. I tried to implement the interface as close as possible to the provided prebuilt netlist simulator. Registers, RAM, and ROM are maintained using dictionary, so in particular, there is no hard limit for memory except that allowed for the simulator, at the expense of an additional log factor to the complexity. By default, all variables and memory are initialised to `0` or `False`.

This convention does lead to the following behavior. When the scheduler is evoke, for the equation `RAM addr_size word_size read_addr write_enable write_addr write_data`, we only required that `read_addr`, `write_enable`, `write_addr`, i.e. read address, flag to switch from read to write, and write address, be calculated and well-defined.

In addition to basic functionalities, the simulator is also capable of error handling with detailed messages, including detecting if

- there is a combinatorial cycle;
- there is a variable assigned twice;
- there is an unknown variable;
- there is an ill-defined value to be written to RAM;
- there is an operator with bad arguments;
- there is an unknown error making variable not found.

3. Difficulties

The main difficulty at the beginning was my unfamiliarity with OCaml, insofar as I was considering re-implement the simulator in an imperative language, such as Python. Another difficult concerned the behavior of memory and registers, which I found baffling and not well-documented enough. No information concerning the initial values for variables (which are required for the implementation of RAM and registers) is provided, and I chose to make a (somewhat arbitrary) convention.