Icarus Verilog Usage Guide

Icarus Verilog comes with two tools, the compiler, and the program that executes the compiled executable. These are named **iverilog** and **vvp**, respectively.

Icarus Verilog is not strict on naming conventions for file suffixes, they can be listed as .ver, .v, .vlg, .vg etc

To compile your source HDL into an executable, you would use the following:

```
$ iverilog -o <name> <source file(s)>
```

To execute the executable you use vvp:

```
$ vvp <name>
```

It's possible to list all sources, which is often inconvienent for large designs. To make this easier, you can use a command file, which can be any file containing text with your sources seperated by newlines. It is specified with the "-c" flag.

EX: **file_list.txt**

```
$ iverilog -o my_design -c file_list.txt
$ vvp my_design
```

As designs get more complicated, they will contain multiple heirarchical Verilog modules that represent the heirarchy of your design. Typically, there is one module that instantiates other modules but is not instantiated by other modules. Icarus Verilog chooses as roots all of the modules that are not instantiated by other modules. If there is no such module, the compiler will not be able to choose a root, and the designer must use the "-s root" switch to identy the root:

```
$ iverilog -s main -o hello hello.v
```

** The "-s" flag turns off automatic search for other root modules. **

Common Flags

- -c <cmdfile>
- -d <flag>
  - Options:
    - scope
    - eval_tree
    - elaborate
    - synth2
- g <generation flag>
  - specifies the language and extensions during compilation
    - options
      - 1995 (IEEE 1364-1995 standard)
      - 2001 (IEEE 1364-2001 standard)
      - 2001-noconfig (config file support disabled)
        - Removes config file keywords from the language and so helps some programs written to older 2001 compile
      - 2005 (IEEE 1364-2005 standard)
        - Default after v0.9
      - 2009 (IEEE 1800-2009 standard)

        - Includes SystemVerilog
        - SystemVerilog support not present in v0.9 and earlier. Actual SV support ongoing
    - 2012 (IEEE 1800-2012)
        - SystemVerilog included
    - Verilog-ams

How Icarus Verilog Works

Compilation & Elaboration:

Simulation of a design = compiling and executing a program. The verilog sources that represent that simulation model and the test bench are compiled into an executable form and executed by a simulation engine (vvp).

Steps for compilation:

1. Macro Pre-Processing
2. Compilation
3. Elaboration
4. Optimizations (Optional)
5. Code Generation

The macro preprocessing step performs textual substitutions of macros defined with `define statements, and `include statements, and `ifdef and `ifndef

The macrprocessor for Icarus Verilog is internally a separate program that can be accessed independently by using the "-E" flag to "iverilog" command:

$ iverilog -E -o out.v example.v

Command = example.v is preprocessed, and then output into out.v

Normally, the "-E" flag is not used and the preprocessed Verilog is instead sent to the compiler which takes as input preprocessed Verilog and generates an internal parsed form. The parsed form is an internal representation of the source file, and not available to the user.

Steps for Elaboration:

1. Takes the parsed form, chooses the root modules
2. Instantiates those roots. The root instances may contain instances of other modules (which can contain instances of other modules).
3. Creates a heirarchy of module instances that ends with primitive gates and statements

Module vs Module Instances

Module = type, description of the contents of module instances that have its type

Module Instance = identified with a name and the type of instance

Root Modules (special case) – programmer does not give instance names

Elaboration creates a heirarchy of scopes where each module instance creates a new scope within its parent module, with each root starting a heirarchy. The elaborated design is then optimized to reduce it to an optimal but equivalent form.

Optimizations can be:

- Forms of constant propagation and dead code elimination.
- Useless logic is eliminated
- Constant expressions are pre-calculated

** optimized form not accessible to the user **

The optimized code is passed to a code generator that writes the design into an executable form. For simulation, the code generator uses .vvp format (text that can be executed by the simulation engine).

Includes

$ iverilog -I/directory/to/search example.v

To look in directory for library modules when the program instantiates a module that is not otherwise defined:

$ iverilog -y/library/to/search example.v

** can include multiple “-y” flags on the command line **

Advanced Command Files

- Name all the sources that make up a design
- Then compilation can just be iverilog -c example.cf
- Command file = configuration
- Include search paths defined with “+incdir+” tokens
- Directory paths seperated by + not spaces