# OpenChar

## User Manual

---

**Version 0.1**

**July 7, 2026**

https://github.com/Blebowski/OpenChar

# Contents

# 1 Introduction

OpenChar is an open-source VLSI library characterization tool. OpenChar invokes analog simulator to measure characteristics of analog circuits and produces digital views for ASIC VLSI design such as Synopsys Liberty (.lib) and Verilog model (.v).

## 1.1 Development status

OpenChar is in its pre-release / alfa version state, meaning internal APIs as well as user interface may change in the future.

## 1.2 Supported platforms

OpenChar is developed under Ubuntu Linux, and it is currently the only supported platform.

## 1.3 Building OpenChar

OpenChar is built with CMake and C++20 compliant compiler. OpenChar has following dependencies on Ubuntu:

```
build-essential cmake tcl tcl-dev libreadline-dev tcl-tclreadline tk tk-dev
```

## 1.4 Invoking OpenChar

OpenChar is a command line application invoked like so:

```
openchar
    -file <script>     TCL script to execute
    -help              Show this help
```

Upon launching, OpenChar starts a TCL shell and executes a TCL script if provided by the `-file` argument. After executing the TCL script, OpenChar stays in the TCL shell, unless the TCL script contains `exit` command.

OpenChar uses analog simulator NGSPICE by invoking a `ngspice` executable. The `ngspice` executable shall be available in PATH environment variable when launching openchar.

# 2 Features

## 2.1 Supported cell types

- Combinatorial cells
- D-style flip-flops with arbitrary clock polairty and preset/clear polarities

## 2.2 Measured parameters

- Delays and transitions arcs on combinatorial cells
- Internal power on combinatorial cells
- Leakage power for combinatorial and sequential cells
- D flip-flop delay from clock to output
- Setup and Hold on D-flip-flop
- Minimal pulse width on clock pin of D-flip-flop

## 2.3 Automatically recognized functions

- Logic function of combinatorial cells
- Polarity and function of asynchronous pins in sequential cells
- Recognizing if sequential cell is latch or flip-flop
- Clock polarity for sequential cells

# 3 TCL commands

OpenChar has a TCL shell that contains all TCL built-in commands. OpenChar specific commands are described in this section.

All values that are specified in TCL commands have following units:

- **Temperature:** Degree Celsius
- **Voltage:** Volt
- **Time:** NanoSecond
- **Current:** MicroAmp
- **Capacitance:** PicoFarad
- **Resistance:** KiloOhm
- **Power:** NanoWatt
- **Energy:** PicoJoule

## characterize_library

### Syntax

`characterize_library`

### Description

Characterize defined cells into a library.

The cells should be previously defined by `define_cell` command. This commands shall be used only once during single invocation of OpenChar.

## define_cell

### Syntax

```
define_cell
  -area       number              Cell area (in square microns).
  -async      {pin_names}         Asynchronous set/clear pin or pins.
  -clock      pin_name            Clock pin
  -constraint constraint_template Name of constraint template to use.
  -delay      delay_template      Name of delay template to use.
  -footprint  footprint_name      Cell footprint.
  -input      {pin_names}         Input pin or pins.
  -output     {pin_names}         Output pin or pins.
  cell_name                       Name of the cell.
```

### Description

Define a cell to be characterized.

The name of the defined cell must match the .subckt definition in SPICE netlist of the cell. Use `-input`, `-output`, `-clock` and `-async` to define pins of the cell.

OpenChar recognizes type of the cell to characterize according to the following rules:

- If only `-input` and `-output` are set, the cell is combinatorial cell.
- If `-clock` or `-async` inputs are present, the cell is sequential cell.

OpenChar automatically recognizes logic function of combinatorial cells. To set templates to be used when characterizing the cell, use `-delay` for delay template and `-constraint` for constraint template such as setups or hold.

To specify additional data that do not impact characterization, but are written to Liberty file by `write_lib` command, use `-area` and `-footprint` switches.

### Examples

The following example defines an inverter cell and 2-input AND cell:

```
define_cell -input A -output Y -delay del_templ_5x5 INV
define_cell -input A B -output Y -delay del_templ_5x5 AND2
```

The following example defines D Flip-flop with asynchronous reset:

```
define_cell -input D -async RB -output Q -clock CK \
            -delay del_templ_5x5 -constraint constr_templ_3x3 DFF
```

## define_template

### Syntax

```
define_template
  -index_1      {values}      List of values to be used as first index.
  -index_2      {values}      List of values to be used as second index.
  -type         (delay|power) Type of template to be defined.
  template_name               Name of the template.
```

### Description

Defines a template for characterization.

A template is set of numeric values used when building NLDM delay or power tables as well as constraints. A template name shall be unique. Use `-index_1` and `-index_2` to specify set of incrementing numbers for the given template. For `-type delay` and `-type power` template, the `-index_1` corresponds to contrained pin transition and `-index_2` corresponds to output capacitance load. For `-type constraints` template, the `-index_1` corresponds to contrained pin transition and `-index_2` corresponds to related pin transition.

### Examples

The following example defines delay and constraint templates:

```
define_template -index_1 0.01 0.02 0.05 0.1 0.2 \
                -index_2 0.05 0.1 0.3 0.7 1.0 \
                -type delay del_templ_5x5


define_template -index_1 0.01 0.1 0.5 \
                -index_2 0.01 0.1 0.5 \
                -type constraint constr_templ_3x3
```

## help

### Syntax

**`help`**

### Description

Show all available commands.

## read_spice

### Syntax

```
read_spice
  -corner       corner          Corner in the model to be used (e.g.  tt,ss,ff).
  -type         (netlist|model) Type of the file provided.
  spice_files                   Name of the supply voltage net.
```

### Description

Read SPICE netlist(s) or model deck(s).

The spice files are included in the simulated analog testbench based on `-type` argument set:

- With `.lib` SPICE statement for `-type model`
- With `.include` SPICE statement for `-type netlist`

## report_app_vars

### Syntax

`report_app_vars`

### Description

Report application variables and their values.

See section 4 for description of all OpenChar variables.

## set_gnd

### Syntax

```
set_gnd
  net_name       Name of the ground net
  voltage_value  Ground voltage value
```

### Description

Define ground voltage and ground net name.

## set_operating_condition

### Syntax

```
set_operating_condition
  -name         string Name of the operating condition.
  -supply_name string Supply name set by 'set_vdd' command.
  -temp         float  Operating temperature.
  -voltage      float  Operating voltage.
```

### Description

Define operating conditions (supply voltage, temperature, name).

The operating conditions are used for characterization and written to the `.lib` file when `write_lib` is invoked.

If no operating conditions were defined, OpenChar uses default operating condition with folowing properties:

- **Name:** DEFAULT_OPCOND
- **Power net:** VDD 1.2 V
- **Ground net:** GND 0 V
- **Temperature:** 25 C

## set_vdd

### Syntax

```
set_vdd
  net_name       Name of the supply voltage net
  voltage_value  Supply voltage value
```

### Description

Define supply voltage and supply voltage net name.

## write_library

### Syntax

```
write_library
  liberty_file string Output library file name.
```

### Description

Write characterized library to a liberty file.

The command shall be executed only after `characterize_library` was executed first.

## write_verilog

### Syntax

```
write_verilog
  output_file string Output verilog file name.
```

### Description

Create verilog model file for the cells in the library

# 4 TCL variables

## delay_in_fall

**Type**: DOUBLE
**Default value**: 0.500000
**Description**:

The % threshold for input falling signal from which delays are measured. Default is 50 %.

## delay_in_rise

**Type**: DOUBLE
**Default value**: 0.500000
**Description**:

The % threshold for input rising signal from which delays are measured. Default is 50 %.

## delay_out_fall

**Type**: DOUBLE
**Default value**: 0.500000
**Description**:

The % threshold for output falling signal to which delays are measured. Default is 50 %.

## delay_out_rise

**Type**: DOUBLE
**Default value**: 0.500000
**Description**:

The % threshold for output rising signal to which delays are measured. Default is 50 %.

## max_threads

**Type**: INT
**Default value**: 20
**Description**:

Maximal number of concurrent running simulations during characterization.

## run_directory

**Type**: STRING
**Default value**: /WORK/characterizer/build/doc/manual
**Description**:

Folder where all simulation data are stored during characterization.

### sim_timestep

**Type**: DOUBLE
**Default value**: 0.001000
**Description**:

Analog simulator timestep.

### slew_lower_fall

**Type**: DOUBLE
**Default value**: 0.200000
**Description**:

The % threshold on output waveform to measure transition to. Default is 20 %.

### slew_lower_rise

**Type**: DOUBLE
**Default value**: 0.200000
**Description**:

The % threshold on output waveform to measure transition from. Default is 20 %.

### slew_upper_fall

**Type**: DOUBLE
**Default value**: 0.800000
**Description**:

The % threshold on output waveform to measure transition from. Default is 80 %.

### slew_upper_rise

**Type**: DOUBLE
**Default value**: 0.800000
**Description**:

The % threshold on output waveform to measure transition to. Default is 80 %.