Wren6991 / asciiwave Public

Turn WaveDrom timing diagrams into ASCII art

BSD-3-Clause license

85 stars 4 forks

Star Watch

Code Issues Pull requests Actions Projects Wiki Security Insig

master

Wren6991 Add licence ... on Mar 14, 2020 13

View code

Readme.md

asciiwave: WaveDrom to ASCII art

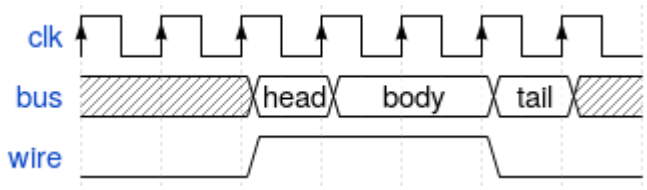
This utility reads WaveDrom JSON files like this:

```
{ signal: [
  { name: "clk",  wave: "P......" },
  { name: "bus",  wave: "x.==.=x", data: ["head", "body", "tail", "data"] },
  { name: "wire", wave: "0.1..0." }
]}
```

And produces ASCII art like this:

```
$ ./asciiwave example/step3.json
      ┌─┐  ┌─┐  ┌─┐  ┌─┐  ┌─┐  ┌─┐  ┌─┐  ┌─
clk : ┘ └──┘ └──┘ └──┘ └──┘ └──┘ └──┘ └──┘ └─
      xxxxxxxxxxxx/    \/          \/    \xxxxxx
bus : xxxxxxxxxxxx\head/\   body   /\tail/xxxxxx
      ┐           ┌──────────────┐          ┌─
wire: └───────────┘              └──────────┘
```

WaveDrom would usually render a PNG or SVG like the below:



However, PNGs can not be pasted into comments in your HDL project!

asciiwave requires the `json5` library from PyPI, as a lot of WaveJSON samples floating around on the internet rely on non-vanilla-JSON features like unquoted keys, single-quoted strings and trailing commas. The `jsonschema` library is also required, for input validation. These can be obtained via:

```
$ pip3 install json5 jsonschema
```

asciiwave features a watch mode (`-w`), which will continously poll a file on disk, and redraw whenever the file changes. This can be used interactively alongside a text editor.

```
$ ./asciiwave --watch example/step4.json

            ┌──┐  ┌──┐  ┌──┐  ┌──┐  ┌──┐  ┌──┐  ┌──┐┆┌──┐  ┌──┐  ┌──┐  ┌──
clk         : ┘  └──┘  └──┘  └──┘  └──┘  └──┘  └──┘┆┘  └──┘  └──┘  └──┘  └──
              xxxxxxxxxxxx/    \/    \/    \xxxxxx┆/         \xxxxxx
Data        : xxxxxxxxxxxx\head/\body/\tail/xxxxxx┆\   data   /xxxxxx
              ┐           ┌─────────────────┐     ┆┌──────────┐     ┌─────
Request     : └───────────┘                 └─────┆┘          └─────┘

              ┌──────────────────────────────────┐┆┐     ┌──────────
Acknowledge: ┘                                  ┆└─────┘

Watching file example/step4.json
Ctrl-C to exit
```

There are simple command-line options for formatting:

```
$ ./asciiwave --hscale=4 --graphics=tall example/step4.json
              ┌────┐    ┌────┐    ┌────┐    ┌────┐    ┌────┐    ┌────┐    ┆┌────┐
┌────┐    ┌────┐
              │    │    │    │    │    │    │    │    │    │    │    │    ┆│    │
│    │    │    │
clk         : ┘    └────┘    └────┘    └────┘    └────┘    └────┘    └────┆┘
└────┘    └────┘    └────
              xxxxxxxxxxxxxxxxxxx/        \/        \/        \xxxxxxxxxx┆/
```

```
\xxxxxxxxxx
             xxxxxxxxxxxxxxxxxxx   head      body      tail   xxxxxxxxxx┆
data        xxxxxxxxxx
Data       : xxxxxxxxxxxxxxxxxxx\       /\        /\       /xxxxxxxxxx┆\
/xxxxxxxxxx
             ┐                   ┌──────────────────────────┐
┆┌──────────────────┐
             │                   │                          │        ┆│
│
Request    : └───────────────────┘                          └────────┆┘
└─────────

             ┌──────────────────────────────────────────────────────────┆┐
┌──────────────────
             │                                                          ┆│
│
Acknowledge: ┘
┆└──────────┘

$ ./asciiwave --hscale=1 --graphics=tiny example/step4.json
clk        : ┌┐_┌┐_┌┐_┌┐_┌┐_┌┐_┆┌┐_┌┐_┌┐_
Data       : xxxxxxxx<he><bo><ta>xxxx┆< data >xxxx
Request    : ┐_______┌──────────┐___┆┌────────┐___

Acknowledge: ┌──────────────────────┆┐___┌────────
```

WaveJSON Subset

asciiwave does not implement the full gamut of WaveJSON features. It supports:

- `wave` commands: `1hHu 0lLd pPnN =2345 zx |`
- The `hscale` config property: the width of each time unit is `hscale * 2 + 2` characters. This is overridden by the `--hscale` command line parameter.
- The `period` signal property: this can be a floating point number. The width of each wave time unit is multiplied by `period` and rounded down.
- The `phase` signal property: this can be a floating point number. The signal is advanced (positive) or retarded (negative) by this number of periods.
- The `data` signal property: either an array of strings, or a single string containing whitespace-separated values.

Graphics

asciiwave defines its graphics like this:

```
graphics_default = [
  "0+1-rfxz< >|UuDd",
  "  ┌─┐┌┐x_/ \┆┌┈  ",
  "  ┘ └┘└x \ /┆  └┈"
]
```

The first line is a key which maps asciiwave's internal representation of wire state to columns of the graphics; the following lines contain the actual graphics. These can be modified if you can't use the Unicode box drawing characters, or have found better-looking characters.

The height is not fixed at 2 lines; any positive number of lines will do. However, the width of each wire state is limited to one column, to simplify rendering (this will be fixed)

Releases

No releases published

Packages

No packages published

Languages

- **Python** 100.0%