Writing GTKWave documents, with style

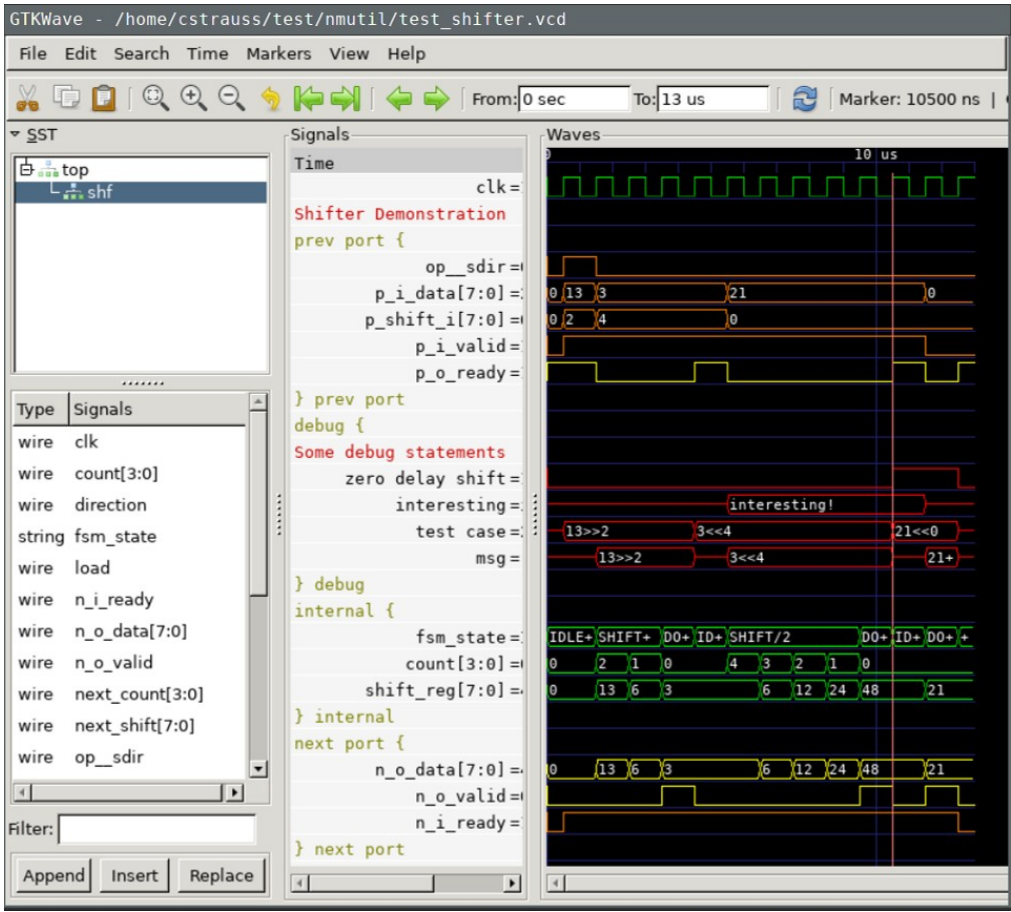
A Python-based CSS-like mini language for generating GTKWave documents

FOSDEM 2022

Cesar Strauss

What is GTKWave?

- GTKWave is a nice waveform viewer
- Useful for digital design
- Has many formatting features
- Use the GUI to create and save the layout (*.gtkw)



Issues with GTKWave's native layout format

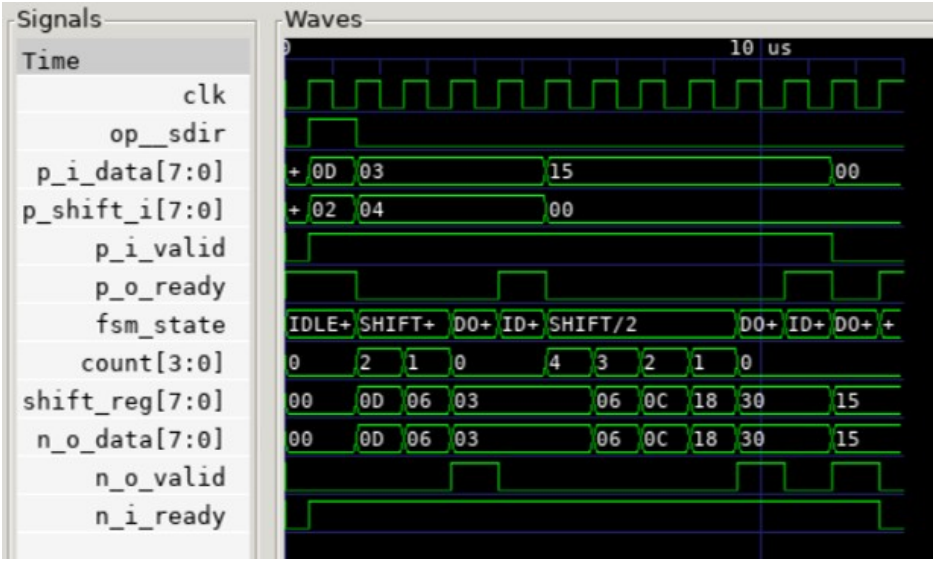
- Inconvenient to share and put on Version Control
- ASCII based, but cryptic
  - Not meant to be editable by hand
  - Magic numeric constants
  - Need to use the GUI (can be tedious)
- Includes GUI settings

Mini-language to generate GTKWave documents

- Python based
  - Useful for Python-based HDLs and simulators
  - Everything can be parametrized
  - Leverages API from pyVCD module for writing in the proper format
- Style can be separated from content
  - Inspired from HTML+CSS

Simple trace list

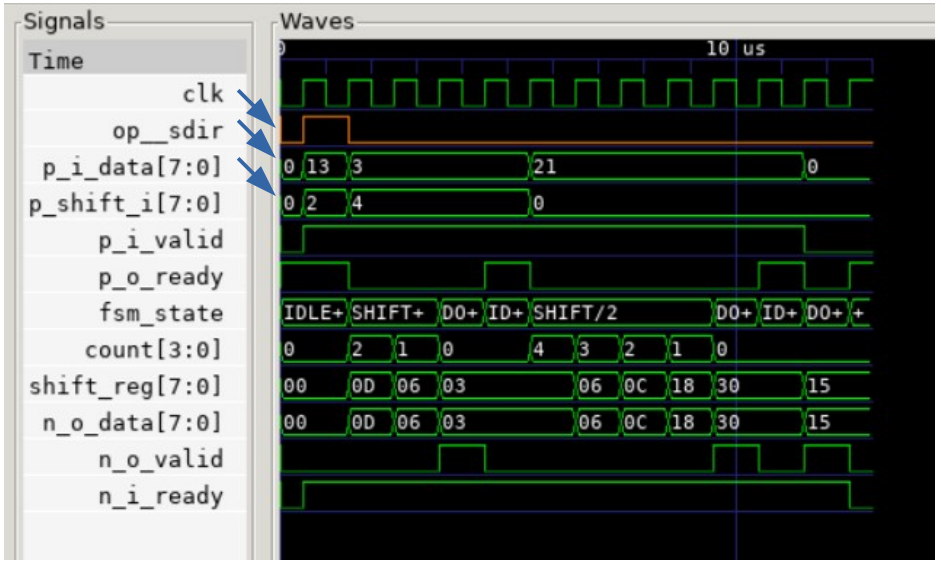
```
from nmutil.gtkw import write_gtkw
traces = [
        'clk',
        # prev port
        'op__sdir', 'p_i_data[7:0]',
        'p_shift_i[7:0]', 'p_i_valid', 'p_o_ready',
        # internal signals
        'fsm_state', 'count[3:0]', 'shift_reg[7:0]',
        # next port
        'n_o_data[7:0]', 'n_o_valid', 'n_i_ready'
]
write_gtkw('simple.gtkw', 'test_shifter.vcd', traces, module='top.shf', clk_period=1e-6)
```



Sane zoom level by default!

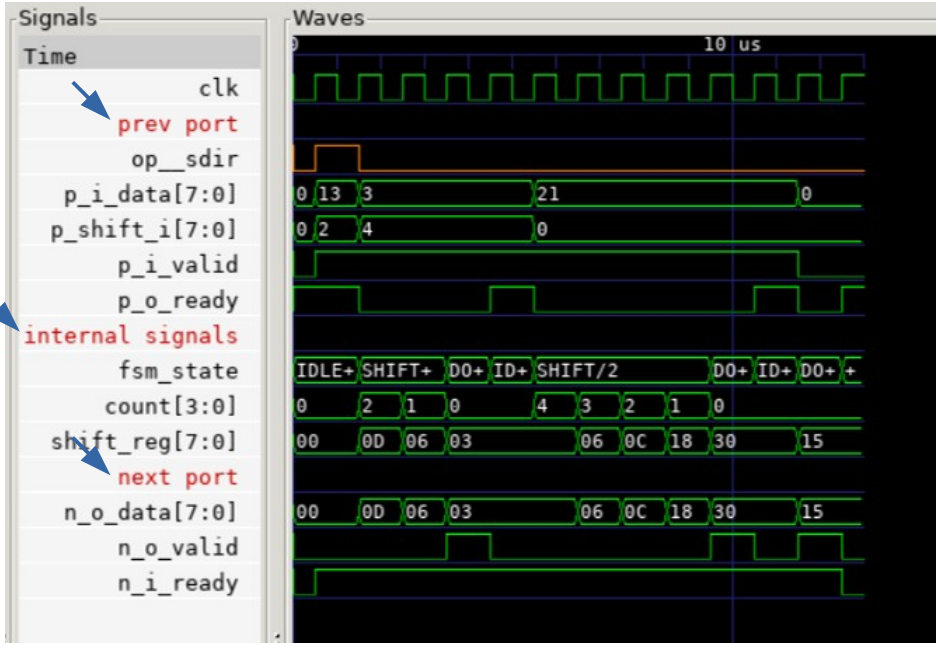
Adding inline attributes

```
traces = [
        'clk',
        # prev port
        ('op__sdir', {'color': 'orange'}),
        ('p_i_data[7:0]', {'base': 'dec'}),
        ('p_shift_i[7:0]', {'base': 'dec'}),
        'p_i_valid', 'p_o_ready',
        # internal signals
        'fsm_state', 'count[3:0]', 'shift_reg[7:0]',
        # next port
        'n_o_data[7:0]', 'n_o_valid', 'n_i_ready'
]
```



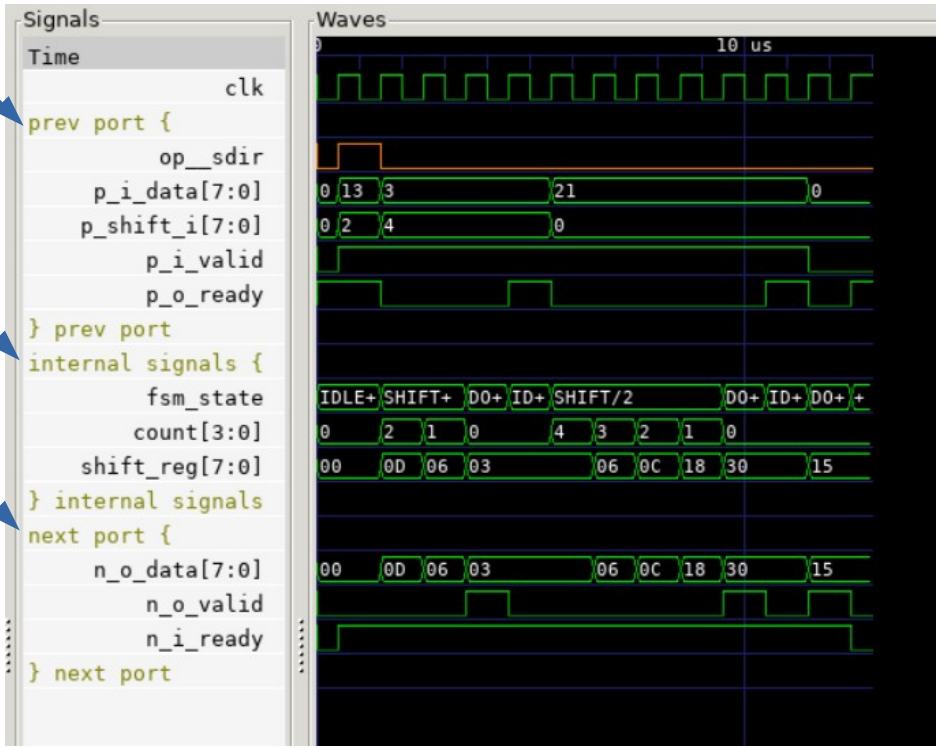
Comments in the signal pane

```
traces = [
        'clk',
        ({'comment': 'prev port'}),
        ('op__sdir', {'color': 'orange'}),
        ('p_i_data[7:0]', {'base': 'dec'}),
        ('p_shift_i[7:0]', {'base': 'dec'}),
        'p_i_valid', 'p_o_ready',
        ({'comment': 'internal signals'}),
        'fsm_state', 'count[3:0]', 'shift_reg[7:0]',
        ({'comment': 'next port'}),
        'n_o_data[7:0]', 'n_o_valid', 'n_i_ready'
]
```



Collapsible groups

```
traces = [
    'clk',
    ('prev port', [
            ('op__sdir', {'color': 'orange'}),
            ('p_i_data[7:0]', {'base': 'dec'}),
            ('p_shift_i[7:0]', {'base': 'dec'}),
            'p_i_valid', 'p_o_ready'
    ]),
    ('internal signals', [
            'fsm_state', 'count[3:0]', 'shift_reg[7:0]'
    ]),
    ('next port', [
            'n_o_data[7:0]', 'n_o_valid', 'n_i_ready'
    ])
]
```



Using styles

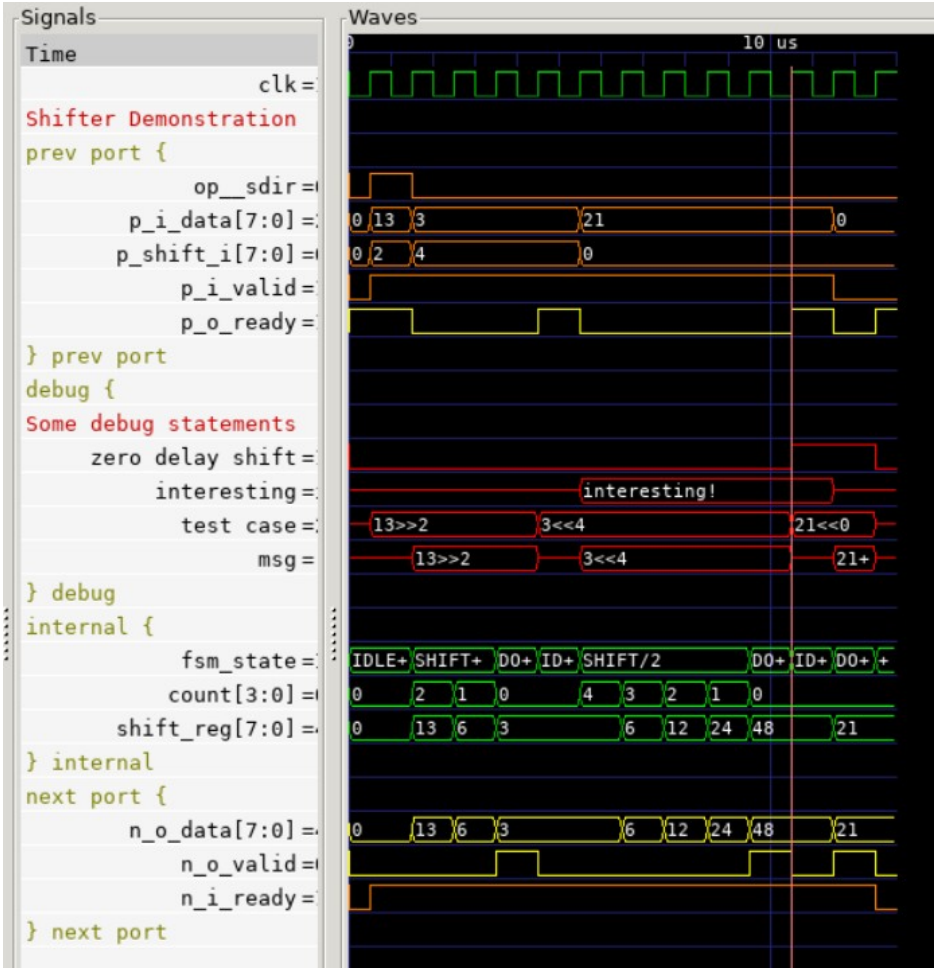
```
gtkwave_style = {
    # Root selector. Gives default attributes for every signal.
    '': {'base': 'dec'},
    # color the traces, according to class
    # class names are not hard-coded, they are just strings
    'in': {'color': 'orange'},
    'out': {'color': 'yellow'},
    # signals in the debug group have a common color and module path
    'debug': {'module': 'top', 'color': 'red'},
    # display a different string replacing the signal name
    'test_case': {'display': 'test case'},
}
```

Using styles

```
gtkwave_desc = [
    'clk',
    {'comment': 'Shifter Demonstration'},
    ('prev port', [
        ('op__sdir', 'in'),
        ('p_i_data[7:0]', 'in'),
        ('p_shift_i[7:0]', 'in'),
        ('p_i_valid', 'in'),
        ('p_o_ready', 'out'),
    ]),
    ('debug', [
        {'comment': 'Some debug statements'},
        ('zero', {'display': 'zero delay shift'}),
        'interesting',
        'test_case',
        'msg',
    ]),
    ('internal', [
        'fsm_state',
        'count[3:0]',
        'shift_reg[7:0]',
    ]),
    ('next port', [
        ('n_o_data[7:0]', 'out'),
        ('n_o_valid', 'out'),
        ('n_i_ready', 'in'),
    ]),
```

Using styles

```
write_gtkw("test_shifter.gtkw", "test_shifter.vcd",
           gtkwave_desc,   gtkwave_style,
           module="top.shf", loc=__file__,
           marker=10500000)
```



Conclusion

- Hoping more people find the idea useful
- Looking for feedback and use cases
- Currently resides on libresoc-nmutil package, under the libre-soc umbrella
  - Can breakout into its own package, maybe

Thanks!