Icarus Verilog & Friends

CPE480, Fall 2019

Hank Dietz

http://aggregate.org/hankd/

UK UNIVERSITY OF KENTUCKY

References

- **gEDA** project (Icarus Verilog is a part of this)
  `http://wiki.geda-project.org/`
- The Icarus Verilog wiki
  `http://iverilog.wikia.com/`
- GTKWave
  `http://gtkwave.sourceforge.net/`
- Yosys
  `http://www.clifford.at/yosys/`

Icarus Verilog Basics

- Verilog source code is a text file
  - Edit it with any text editor (I like emacs)
  - File name ends in `.v` `.vl` `.ver` `.vlg`
- Icarus Verilog is a compiler called `iverilog`
  - Compiler output can be `vvp` `fpga` `vhdl`
  - A `vvp` file is executed by `vvp`, which is the Icarus Verilog runtime engine
- Simulation output is text, trace data

Hello, World

- Make `hello.v` contain:

```
module helloworld;
   initial
      $display("Hello, World!");
endmodule
```

- Compile by:

```
iverilog -o hello hello.v
```

- Simulate (execute) by:

```
vvp hello
```

$ Text Output Basics

- Verilog provides **lots** of $-named tasks for generating formatted output
- Your test bench can work either of two ways:
  - Stimulus only; just generate trace output that some other tool (or human) will examine
  - Complete checker; generate output that identifies any incorrect values
- Ideally, your Verilog test bench should do both; show all values, but note which are bad

Trace Waveform Basics

- `gtkwave` is a visualization tool for trace files, such as a value change dump (VCD)
- In your `.v` file, specify:
  - `$dumpfile("`*file*`.vcd")`
  - `$dumpvars(`*level*`,` *vars_or_modules*`)`
    where *level* means 0→all or 1→listed only
- *file*`.vcd` is created when `vvp` is run
- Visualize using: `gtkwave` *file*`.vcd`

Trace Example

- Make `trace.v` contain:

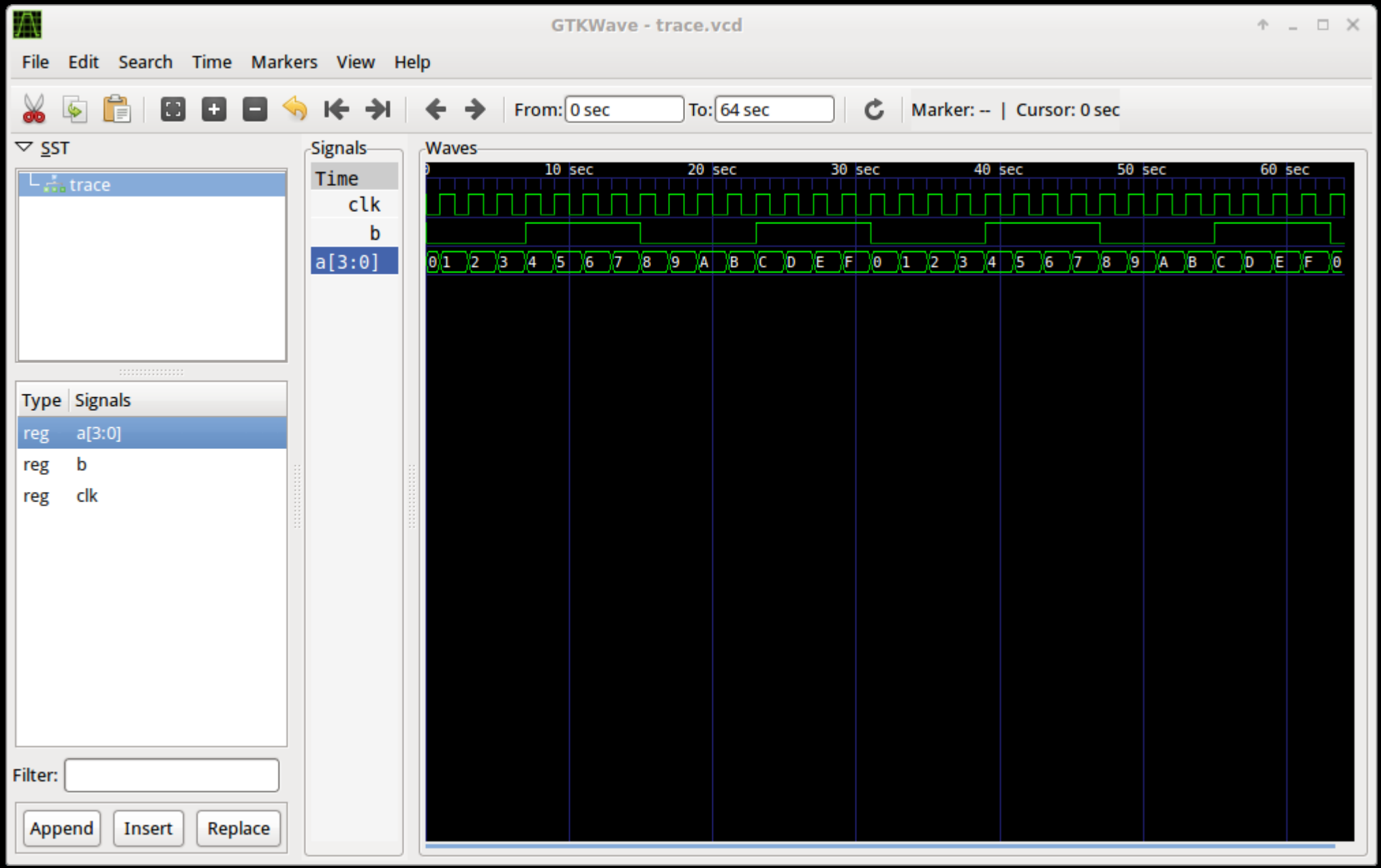
```
module trace;
reg clk=0; reg[3:0]a=4'd0; reg b=0;
always #1 clk=~clk;
always @(posedge clk) begin
  a=a+1; b=a[2]; end
initial begin
  $dumpfile("trace.vcd");
  $dumpvars(0, trace); // all vars
end endmodule
```

Trace Example (continued)

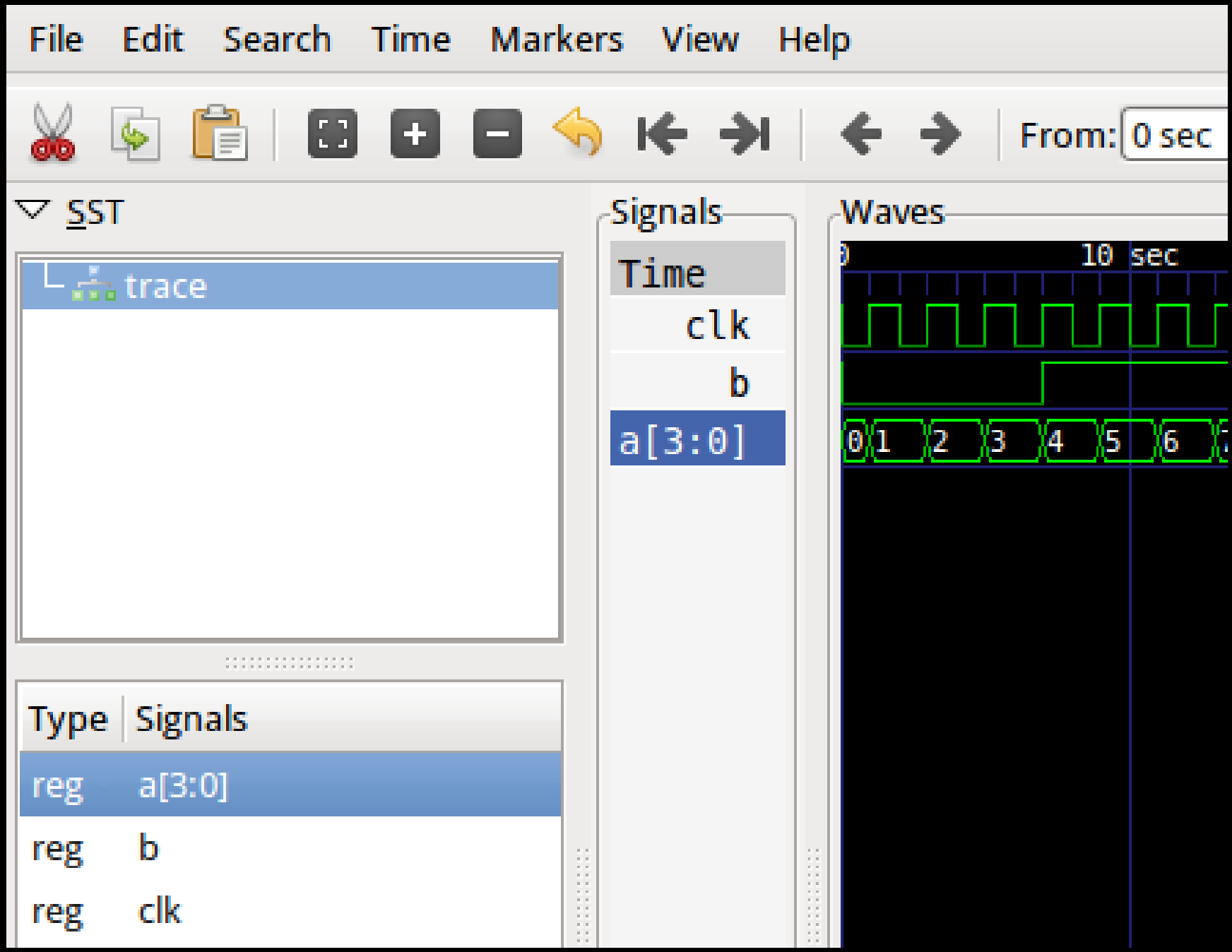
- Compile by:
  `iverilog -o trace trace.v`
- Simulate (execute) by:
  `vvp trace`
  Giving output:

```
VCD info: dumpfile trace.vcd opened for output.
```

- Examine trace with `gtkwave`:
  `gtkwave trace.vcd`



UNIVERSITY OF KENTUCKY


File
Edit
Search
Time
Markers
View
Help
From:
0 sec
SST
trace
Type
Signals
reg
a[3:0]
reg
b
reg
clk
Signals
Time
clk
b
a[3:0]
Waves
10 sec
0
1
2
3
4
5
6

Trace Example Using CGI

- The CGI version doesn't allow specifying a File name for `$dumpfile`
- Example would fail – it times out!
  You must force it to `$finish`

Trace Example Using CGI

```
module trace;
reg clk=0; reg[3:0]a=4'd0; reg b=0;
always #1 clk=~clk;
always @(posedge clk) begin
   a=a+1; b=a[2]; end
initial begin
   $dumpfile; // for CGI
   $dumpvars(0, trace); // all vars
   #100 $finish; // make CGI stop
end endmodule
```

Trace Example CGI Execution

- The complete trace example is pre-loaded into our Icarus Verilog CGI at:

  `http://aggregate.org/EE480/trace.html`

Using Covered On A Trace

- Covered is a tool that "tests tests"
- It needs both Verilog source and VCD files and generates a CCD file from them for a module:

```
covered score -t module -v source.v -o ccdfile -vcd vcdfile
```

- To get the actual coverage report:

```
covered report -d v -o ccdfile
```

- There are various other command-line options:
  http://covered.sourceforge.net/user/chapter.using.html

Circuit Schematic Basics

- Lots of free graphics editors can be used…
  e.g., `gschem Xcircuit KiCad xfig`
- `yosys` is the Yosys Open SYnthesis Suite
  - It sort-of can do ASIC synthesis…
  - It sort-of makes schematics of Verilog code
  - `http://www.clifford.at/yosys/`
- `graphviz` does the real work for `yosys`
  - Great at automatic layout of graphs…
  - `http://www.graphviz.org/`

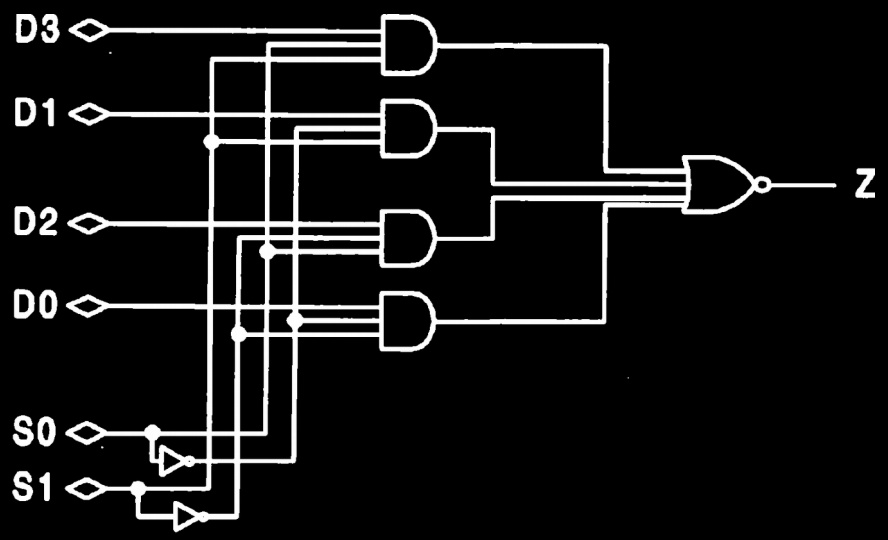
Yosys Schematic Basics

- Start `yosys`
- Issue `> read_verilog` *file.v*
- Issue `> show` *module_name*

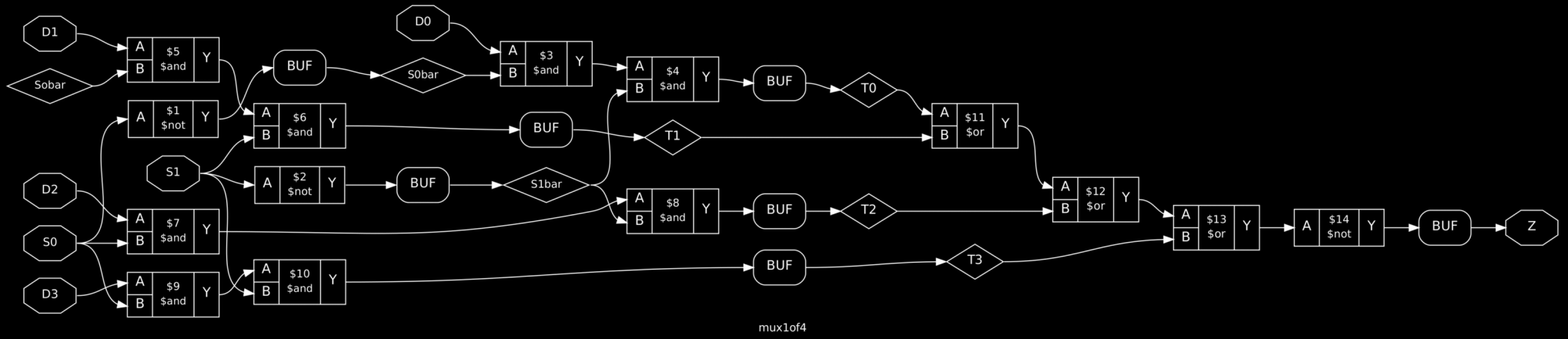
- Resulting graphviz output isn't pretty, but it is a structurally correct representation… useful for debugging, if less so as documentation

Gate Level 1-of-4 Mux

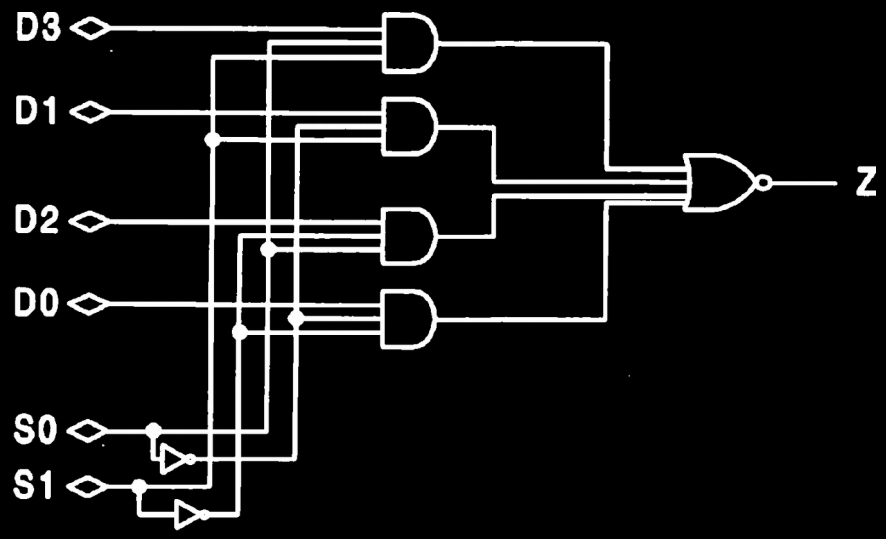
```
module mux1of4(Z,D0,D1,D2,D3,S0,S1);
output Z; input D0,D1,D2,D3,S0,S1;
wire T0,T1,T2,T3;
not(S0bar,S0),(S1bar,S1);
and(T0,D0,S0bar,S1bar),
    (T1,D1,Sobar,S1),
    (T2,D2,S0,S1bar),
    (T3,D3,S0,S1);
nor(Z,T0,T1,T2,T3);
endmodule
```

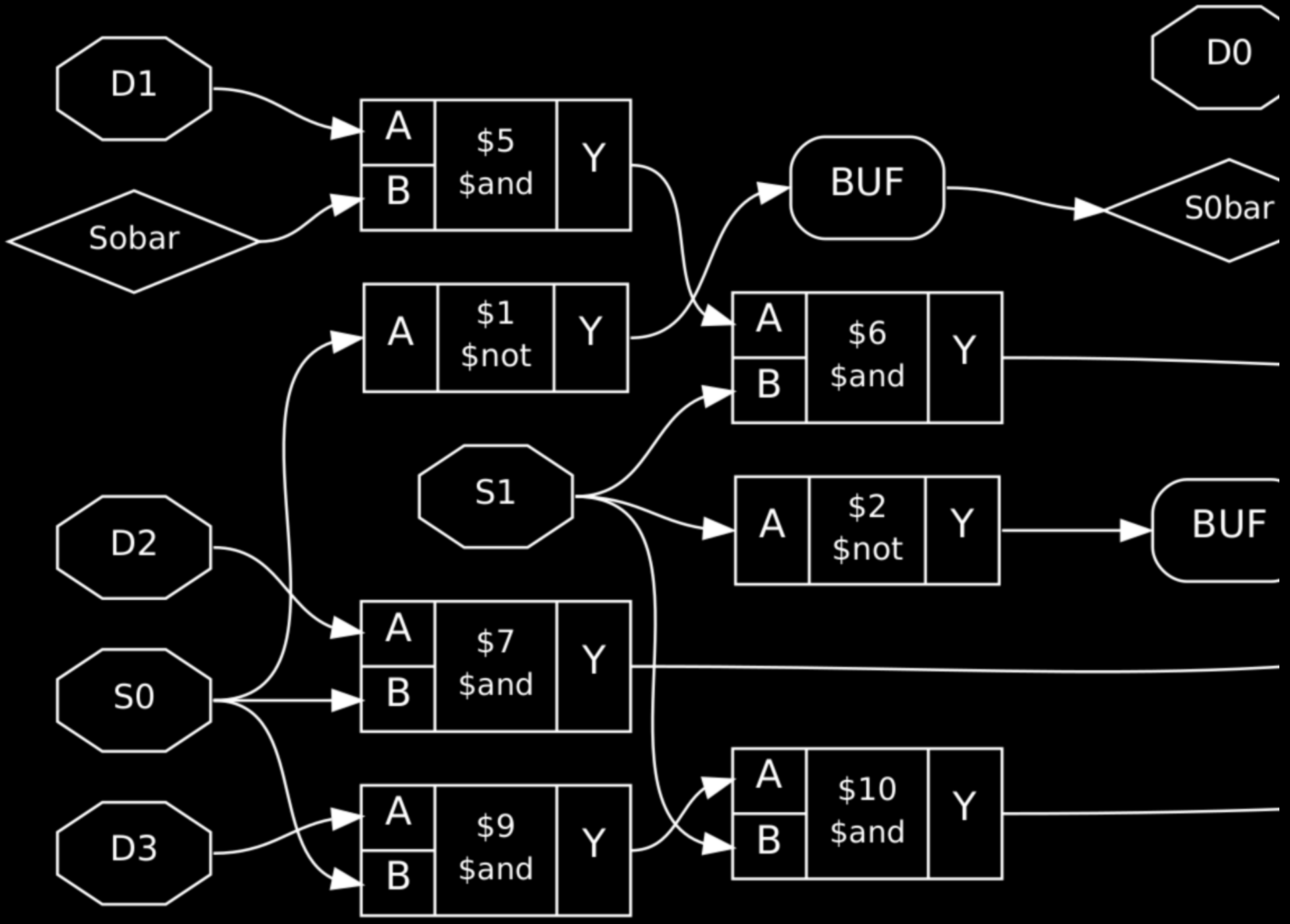


Yosys Schematic Basics



mux1of4



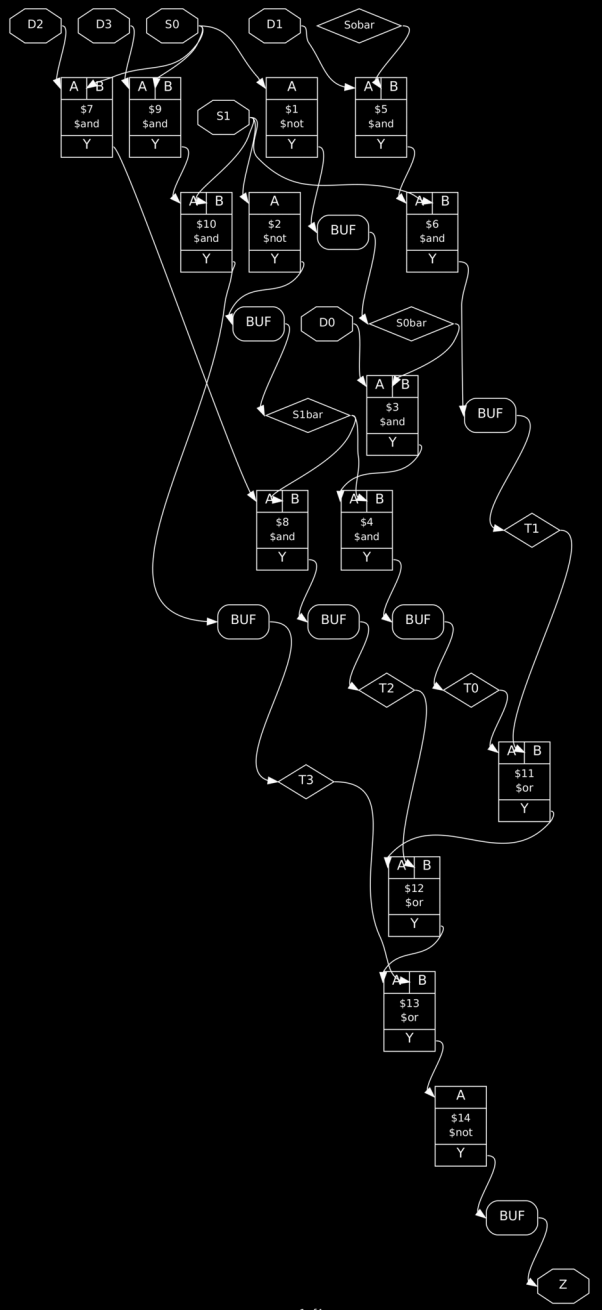


Yosys Dot File Can Be Edited

```
osdigraph "mux1of4" {
label="mux1of4";
rankdir="LR";
remincross=true;
n8 [ shape=diamond, label="T3", color="black", fontcolor="black" ];
n11 [ shape=diamond, label="T2", color="black", fontcolor="black" ];
n14 [ shape=diamond, label="Sobar", color="black", fontcolor="black" ];
n15 [ shape=diamond, label="T1", color="black", fontcolor="black" ];
...
x0 [shape=box, style=rounded, label="BUF"];
x1 [shape=box, style=rounded, label="BUF"];
...
c40:p31:e -> c39:p30:w [color="black", label=""];
x4:e:e -> n11:w [color="black", label=""];
n11:e -> c35:p33:w [color="black", label=""];
...
}
```

Edited

```
osdigraph "mux1of4" {
label="mux1of4";
rankdir="TB";
remincross=true;
n8 [ shape=diamond, label="T3", color="black", fon
n11 [ shape=diamond, label="T2", color="black", fo
n14 [ shape=diamond, label="Sobar", color="black",
n15 [ shape=diamond, label="T1", color="black", fo
...
x0 [shape=box, style=rounded, label="BUF"];
x1 [shape=box, style=rounded, label="BUF"];
...
c40:p31:e -> c39:p30:w [color="black", label=""];
x4:e:e -> n11:w [color="black", label=""];
n11:e -> c35:p33:w [color="black", label=""];
...
}
```



mux1of4