EDA-Tools - Verilog Gate-Level Studio for hardware engineers.

EDA-Tools is a user-friendly HW engineering studio comprising of an array of tools specially designed for performing reports and manipulation on Verilog gate-level netlists.

- EDA-Tools assume a valid Verilog Gate-Level file as input, Bus notation is not supported.
- The current release of EDA-Tools is a 'generic' engineering tool – it operates without the need of loading a vendor library of cells.
- EDA-Tools Studio is designed to operate as a quick prototyping tool (before the sign-off tool). You can perform the manipulations and reports on 'semi-finished' designs and defer the decision on the technology and vendor - the tools are smart enough to differentiate between instances of modules and library gates without a loaded library.

Currently the following tools are available (**NOTE: Bus notation is currently not supported**):

Manipulations – Tools that perform manipulations on the loaded netlist and save the results into a new netlist.

- Remove Buffers
- Replace Library Gate
- Uppercase Library Gate Ports

Reports – Tools that generate many helpful reports on the netlist.

- Count Library Gates
- List Library Gates
- List Physical Paths

EDA-Tools operation mode is unique – once a netlist is read it remains loaded in memory.

- If you perform a manipulation on a loaded netlist and choose to save the resulting netlist into a new file you will have 2 netlists in memory (the source netlist and the netlist after the manipulation).
- You can run any tool on any of the loaded netlists.
- The tool-selection dialog box shows the currently loaded netlists in the upper combo-box.

Remove Buffers

- You supply the buffer definition details - the manipulation will remove the buffer from the netlist, and will do all the necessary rewiring.
- The format of the buffer definition is :
  Buffer_Name <SPACE> **Input_port_Name** <SPACE> **Output_port_Name**
- Only non-pass-through buffers are removed.

Remove Buffers Example:

Source Netlist:

```
module top (clkin, rst_in, out);
input clkin , rst_in  ;
output out ;
    x_inv inst001 ( .i ( outwire ), .o ( clkout ) );
    x_buf inst002 ( .i ( clkout ), .o ( out ) );
    x_buf inst002 ( .i ( clkin ), .o ( outwire ) );
endmodule
```

Netlist After Remove Buffers Manipulation:

```
module top ( clkin , rst_in , out );
input clkin , rst_in ;
output out ;
    x_inv inst001 ( .i ( clkin ) , .o ( out ) );
endmodule
```

Replace Library Gates

- You supply the details of the old gate, the new gate, and specify how ports should be replaced.
- The definitions of the new and old gates must comply with the following format:
  Gate_Name <SPACE> **Input1_Name** <SPACE>… **Output1_ Name**
- The manipulation will replace the old gate with the new gate, and will do all the rewiring.

Replace Library Gates Example:

Source Netlist:

```
module top ( clkin , rst_in , out );
input clkin , rst_in ;
output out ;
    x_inv inst001 ( .i ( clkin ) , .o ( out ) );
endmodule
```

Netlist After Replace Library Gates Manipulation:

```
module top ( clkin , rst_in , out );
input clkin , rst_in ;
output out ;
    not inst001 ( .in ( clkin ) , .out ( out ) );
endmodule
```

Uppercase Library Gate Ports

With this manipulation you can fix common casing errors. It will upper case all ports for all library gates.

Uppercase Library Gate Ports Example:

Source Netlist:

```
module main (clkin, rst_in, out);
input clkin , rst_in  ;
output out ;
  sec inst001(.clkin ( clkin ), .rst_in ( rst_in ), .out ( outwire ) );
  x_buf inst002 ( .i ( outwire ), .o ( out ) );
endmodule

module sec ( clkin, rst_in, out);
input clkin , rst_in  ;
output out ;
  x_buf inst002 ( .i ( clkin ), .o ( out ) );
  x_inv inst003 ( .i ( rst_in ), .o ( nc_wire ) );
endmodule
```

Netlist After Uppercase Manipulation:

```
module main ( clkin , rst_in , out );
input clkin , rst_in ;
output out ;
  sec inst001 ( .clkin ( clkin ) , .rst_in ( rst_in ) , .out ( outwire ) );
  x_buf inst002 ( .I ( outwire ) , .O ( out ) );
endmodule

module sec ( clkin , rst_in , out );
input clkin , rst_in ;
output out ;
  x_buf inst002 ( .I ( clkin ) , .O ( out ) );
  x_inv inst003 ( .I ( rst_in ) , .O ( nc_wire ) );
endmodule
```

Count Library Gates

- This report generates an accurate count of all the library gates within the module you choose.
- If you run it on the **top module** you will get an accurate count of all the library gates in the netlist

Report Example:

```
//
// EDA Tools - Verilog Gate-Level Studio, Version 1.0.2.1 by Elicon
// Visit us here - http://www.elicon.biz/
//

==============================
Module                   Count
==============================

an2                       1135
aoi22                     4
i1                        1140
ic                        15
iobhc                     10
mx21                      1742
mx21i                     695
oai22                     1
or2                       779
or3                       4
or4                       7
xo2                       605
```

List Library Gates

- This report generates a list of all the library gates in the netlist.
- For any of the gates we ensure that all the ports are shown.

Report Example:

```
//
// EDA Tools - Verilog Gate-Level Studio, Version 1.0.2.1 by Elicon
// Visit us here - http://www.elicon.biz/
//

module bd ( z , a );
module mx21 ( z , a , b , sb );
module b1 ( z , a );
module mx21i ( zn , a , b , sb );
module an2 ( z , a , b );
module an3 ( z , a , b , c );
module or2 ( z , a , b );
module oai211 ( c , b , a2 , a1 , zn );
module or3 ( c , b , a , z );
module por ( zn );
```

List Physical Paths

This report generates a list of all **physical paths** for each module you specify.

Report Example:

```
//
// EDA Tools - Verilog Gate-Level Studio, Version 1.0.2.1 by Elicon
// Visit us here - http://www.elicon.biz/
//

// Modules to list: x_lut4_0xff0c, x_lut4_0x6a5f

//
// x_lut4_0xff0c instances:
//
CDU/SCK_a1_I/tlib000001/tlib000004
// x_lut4_0xff0c has 1 instance

//
// x_lut4_0x6a5f instances:
//
CDU/BITCOUNT_BLOCK/COUNT_rtl_0_aauto_generated_acounter_cella7/tlib000001/t
lib000073
CDU/BITCOUNT_BLOCK/COUNT_rtl_0_aauto_generated_acounter_cella7/tlib000001/t
lib000074
CDU/BITCOUNT_BLOCK/COUNT_rtl_0_aauto_generated_acounter_cella7/tlib000001/t
lib000075
CDU/BITCOUNT_BLOCK/COUNT_rtl_0_aauto_generated_acounter_cella3/tlib000001/t
lib000062
// x_lut4_0x6a5f has 4 instances
```