

# **SystemVerilog Ports & Data Types For Simple, Efficient and Enhanced HDL Modeling**

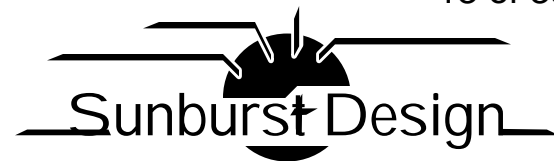
**Clifford E. Cummings**

**Sunburst Design, Inc.**

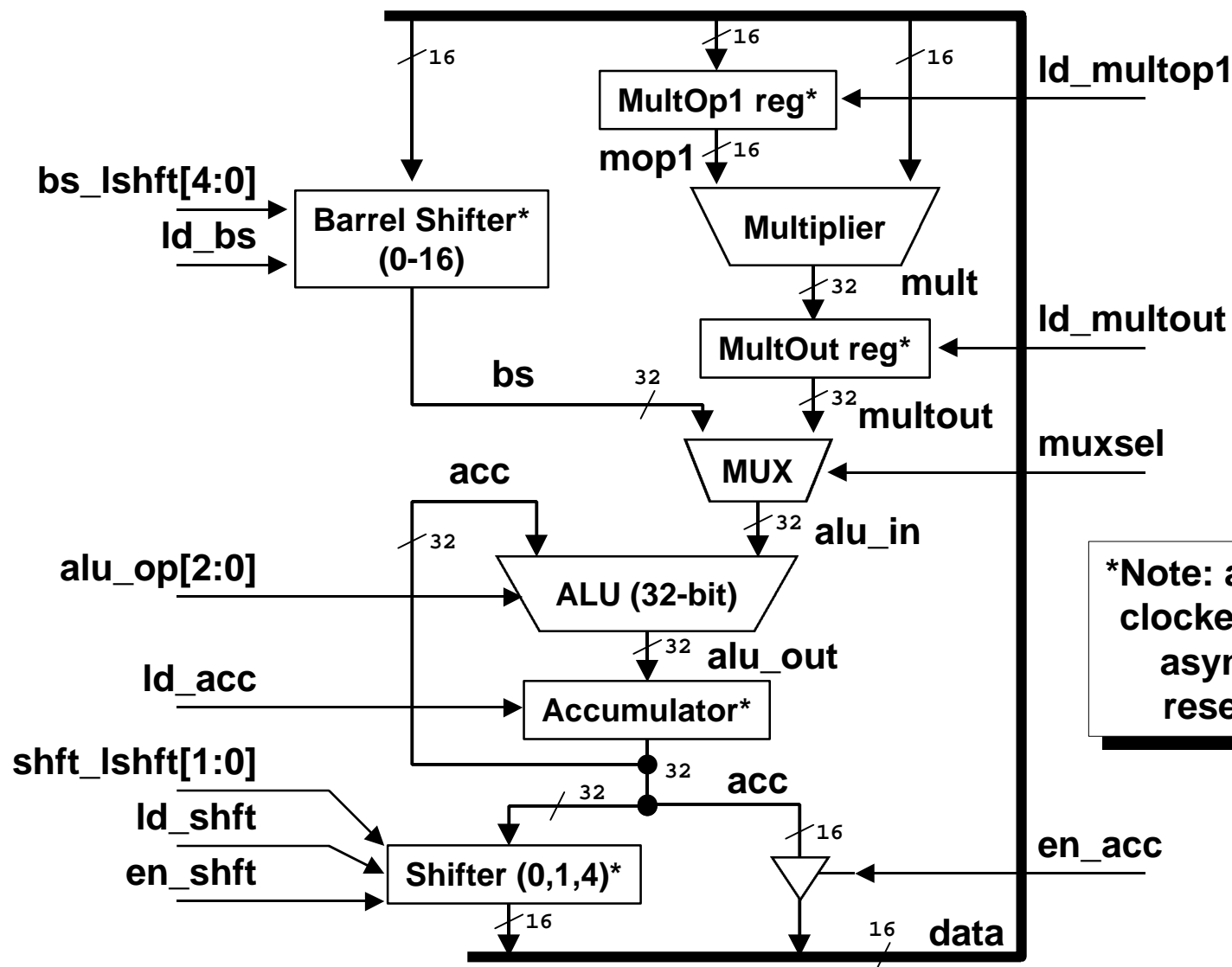
[cliffc@sunburst-design.com](mailto:cliffc@sunburst-design.com)

[www.sunburst-design.com](http://www.sunburst-design.com)

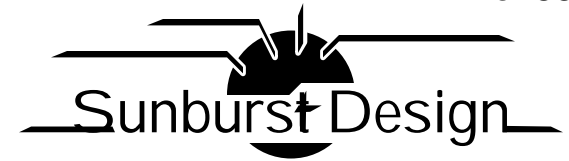
**Expert Verilog HDL & Verilog Synthesis Training**



# Central Arithmetic Logic Unit (CALU) Block Diagram



**\*Note: all registers are clocked by "clk" and asynchronously reset by "rst\_n"**



# Instantiating With Positional Port Connections

---

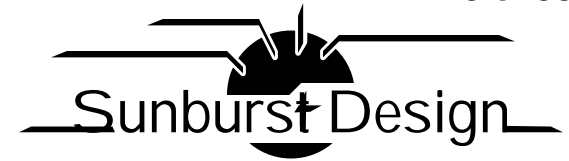
- Modules can be instantiated with positional port connections
- Advantages:
  - Port signal names only have to be listed once
  - Instantiating a module is fast and easy
- Disadvantages:
  - A module instantiated with an incorrect port order can be time consuming to detect and debug
  - If the port order of the instantiated module is changed by a third party, the instantiation will also have to change
  - Minor port order changes are often the most difficult to find and debug

# Instantiating With Named Port Connections

Generally recommended style at most companies



- Modules can be instantiated with named port connections
- Disadvantage:
  - Instantiations are twice as verbose as using positional port connections and take longer to type
- Advantages:
  - Mis-ordering of instance ports does not cause design failures
  - If the port order of the instantiated module is changed by a third party, no instantiation changes are required
  - Minor port order changes do not impact a design
  - If the port names change, the compiler gives immediate feedback that the specified ports do not exist



# CALU Top-Level Module 1/2

Verilog-1995/2001 Named Port Connections Version

```

module calu1 (data, bs_lshft, alu_op, shft_lshft, calu_muxsel,
             en_shft, ld_acc, ld_bs, ld_multop1, ld_multout,
             ld_shft, en_acc, clk, rst_n);

  inout  [15:0] data;
  input  [ 4:0] bs_lshft;
  input  [ 2:0] alu_op;
  input  [ 1:0] shft_lshft;
  input          calu_muxsel, en_shft, ld_acc, ld_bs;
  input          ld_multop1, ld_multout, ld_shft, en_acc;
  input          clk, rst_n;
  wire  [31:0] acc, alu_in, alu_out, bs, mult, multout;
  wire  [15:0] mop1;

  multop1      multop1      (.mop1(mop1), .data(data),
                           .ld_multop1(ld_multop1),
                           .clk(clk), .rst_n(rst_n));

  multiplier   multiplier   (.mult(mult), .mop1(mop1),
                           .data(data));

  multoutreg   multoutreg   (.multout(multout),
                           .mult(mult),
                           .ld_multout(ld_multout),
                           .clk(clk), .rst_n(rst_n));

```

...

**Very  
verbose!**

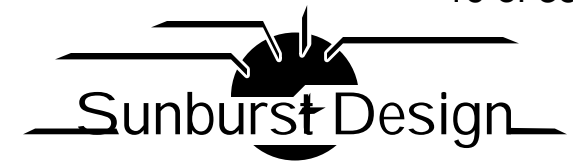
**Lots of  
typing!**

**Lots of  
opportunity  
to introduce  
typos!**

**MultOp1 reg\***

**Multiplier**

**MultOut reg\***



# CALU Top-Level Module 2/2

Verilog-1995/2001 Named Port Connections Version

Barrel Shifter\*  
(0-16)

MUX

ALU (32-bit)

Accumulator\*

Shifter (0,1,4)\*



```

...
    barrel_shifter barrel_shifter (.bs(bs), .data(data),
                                   .bs_lshft(bs_lshft),
                                   .ld_bs(ld_bs),
                                   .clk(clk), .rst_n(rst_n));

    mux2            mux
    alu             alu
    accumulator     accumulator
    shifter         shifter
    tribuf          tribuf

    (.y(alu_in),
     .i0(multout),
     .i1(acc),
     .sel1(calu_muxsel));

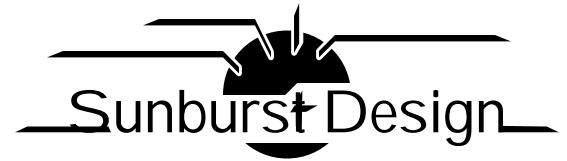
    (.alu_out(alu_out),
     .zero(), .neg(),
     .alu_in(alu_in), .acc(acc),
     .alu_op(alu_op));

    (.acc(acc), .alu_out(alu_out),
     .ld_acc(ld_acc), .clk(clk),
     .rst_n(rst_n));

    (.data(data), .acc(acc),
     .shft_lshft(shft_lshft),
     .ld_shft(ld_shft),
     .en_shft(en_shft),
     .clk(clk), .rst_n(rst_n));

    (.data(data), .acc(acc[15:0]),
     .en_acc(en_acc));

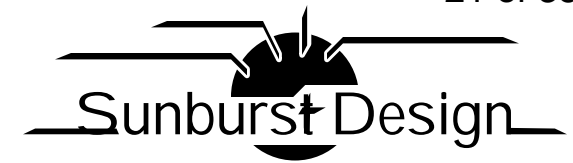
endmodule
  
```



# **SystemVerilog Proposal: Introducing The `.name` Implicit Port Connection Style**

To reduce typing for named port connections

**This part of the proposal is new**



# CALU Top-Level Module 1/2

SystemVerilog .name Implicit Ports Version

```

module calu1 (data, bs_lshft, alu_op, shft_lshft, calu_muxsel,
             en_shft, ld_acc, ld_bs, ld_multop1, ld_multout,
             ld_shft, en_acc, clk, rst_n);

  inout  [15:0] data;
  input  [ 4:0] bs_lshft;
  input  [ 2:0] alu_op;
  input  [ 1:0] shft_lshft;
  input          calu_muxsel, en_shft, ld_acc, ld_bs;
  input          ld_multop1, ld_multout, ld_shft, en_acc;
  input          clk, rst_n;
  wire  [31:0] acc, alu_in, alu_out, bs, mult, multout;
  wire  [15:0] mop1;

```

Matching port names  
are listed just once

```

multop1      multop1      (.mop1, .data, .ld_multop1,
                          .clk, .rst_n);
multiplier   multiplier   (.mult, .mop1, .data);
multoutreg   multoutreg   (.multout, .mult, .ld_multout,
                          .clk, .rst_n);
...

```

MultOp1 reg\*

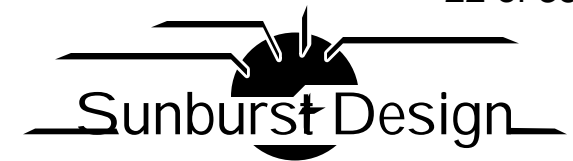
Multiplier

MultOut reg\*

```

multop1 multop1 (.mop1, .data, .ld_multop1,
                .clk, .rst_n);

```



# CALU Top-Level Module 2/2

SystemVerilog .name Implicit Ports Version

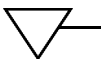
Barrel Shifter\*  
(0-16)

MUX

ALU (32-bit)

Accumulator\*

Shifter (0,1,4)\*



```

...
    barrel_shifter barrel_shifter (.bs, .data, .bs_lshft,
                                   .ld_bs, .clk, .rst_n);

    mux2          mux          (.y(alu_in),
                                .i0(multout),
                                .i1(acc),
                                .sel1(calu_muxsel));

    alu           alu          (.alu_out, .zero(), .neg(),
                                .alu_in, .acc, .alu_op);

    accumulator  accumulator  (.acc, .alu_out,
                                .ld_acc, .clk, .rst_n);

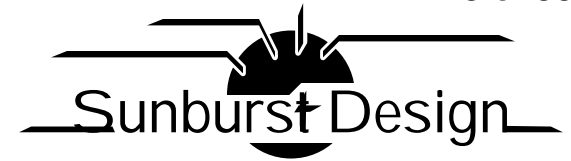
    shifter       shifter     (.data, .acc, .shft_lshft,
                                .ld_shft, .en_shft,
                                .clk, .rst_n);

    tribuf        tribuf      (.data, .acc(acc[15:0]),
                                .en_acc);

endmodule
  
```

All of the advantages of  
named port connections

Less  
verbose!



# Rules for Implicit `.name` Port Connections

(Proposed - Not Official Yet)

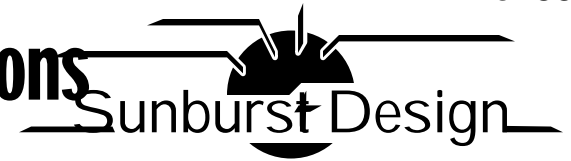
---

- Rule: mixing `.*` and `.name` ports in the same instantiation is prohibited

To help design teams who want to enforce a `.name` style

- Permitted: `.name` and `.name(signal)` connections in the same instantiation
- Rules: the `.name(signal)` connections are required for:
  - size-mismatch
  - name-mismatch
  - unconnected ports

# Advantages of Implicit `.name` Port Connections



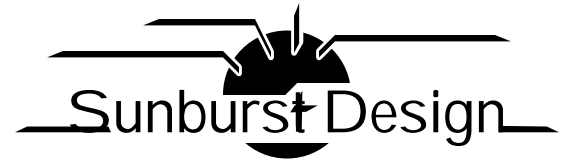
(Proposed - Not Official Yet)

- Shows all port names in the instantiation

Uses `.name` notation without having to repeat the signal name inside of parentheses

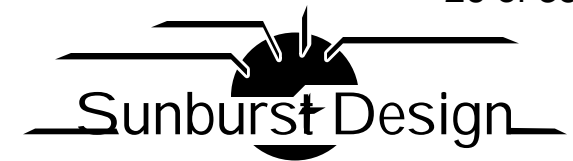
Natural abbreviation style of named port connections

- Port order is not important
- Still has all of the advantages of explicit (or named) port connections
- Not as verbose as explicit (or named) port connections
- Helps design teams who want to enforce a `.name` style



# **SystemVerilog Proposal: Introducing The `.*` Implicit Port Connection Style**

To greatly simplify top-level module  
instantiation



# CALU Top-Level Module

SystemVerilog .\* Implicit Ports Version

```

module calu2 (
  inout  [15:0] data,
  input  [ 4:0] bs_lshft,
  input  [ 2:0] alu_op,
  input  [ 1:0] shft_lshft,
  input          calu_muxsel, en_shft, ld_acc, ld_bs,
  input          ld_multop1, ld_multout, ld_shft, en_acc,
  input          clk, rst_n);

  wire  [31:0] acc, alu_in, alu_out, bs, mult, multout;
  wire  [15:0] mop1;

  multop1      multop1      (*);
  multiplier   multiplier   (*);
  multoutreg   multoutreg   (*);
  barrel_shifter barrel_shifter (*);
  mux2         mux         (.y(alu_in), .i0(multout),
                          .i1(acc), .sel1(calu_muxsel));

  alu          alu         (*, .zero(), .neg());
  accumulator  accumulator (*);
  shifter      shifter     (*);
  tribuf       tribuf      (*, .acc(acc[15:0]));
endmodule

```

**This style emphasizes  
where port differences  
occur**

**Much less  
verbose!**

**MultOp1 reg\***

**Multiplier**

**MultOut reg\***

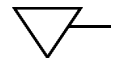
**Barrel Shifter\*  
(0-16)**

**MUX**

**ALU (32-bit)**

**Accumulator\***

**Shifter (0,1,4)\***



# SystemVerilog .\* Implicit Ports

(A Closer Look - 1/4)

Same basic rules apply to the  
.name instantiation syntax

sign

```
module calu2 (
  inout  [15:0] data,
  input  [ 4:0] bs_lshft,
  input  [ 2:0] alu_op,
  input  [ 1:0] shft_lshft,
  input          calu_muxsel, en_shft, ld_acc, ld_bs,
  input          ld_multop1, ld_multout, ld_shft, en_acc,
  input          clk, rst_n);
```

```
wire  [31:0] acc, alu_in, alu_out, bs, mult, multout;
wire  [15:0] mop1;
```

Internal wire busses  
must be declared

```
multop1      multop1      (*);
multiplier   multiplier   (*);
multoutreg   multoutreg   (*);
barrel_shifter barrel_shifter (*);
mux2         mux          (.y(alu_in), .i0(multout),
                          .i1(acc), .sel1(calu_muxsel));

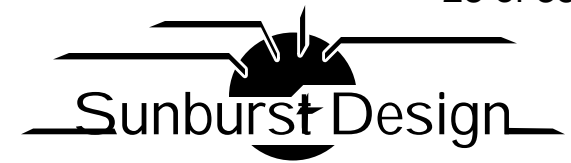
alu          alu          (*, .zero(), .neg());
accumulator  accumulator  (*);
shifter      shifter      (*);
tribuf       tribuf       (*, .acc(acc[15:0]));
endmodule
```

All port names and port  
sizes match the nets  
connected to the ports

```
module multop1 (
  output [15:0] mop1,
  input  [15:0] data,
  input          ld_multop1, clk, rst_n);
  ...
```

# SystemVerilog .\* Implicit Ports

(A Closer Look - 2/4)



```

module calu2 (
  inout  [15:0] data,
  input  [ 4:0] bs_lshft,
  input  [ 2:0] alu_op,
  input  [ 1:0] shft_lshft,
  input          calu_muxsel, en_shft, ld_acc, ld_bs,
  input          ld_multopl, ld_multout, ld_shft, en_acc,
  input          clk, rst_n);

  wire  [31:0] acc, alu_in, alu_out, bs, mult, multout;
  wire  [15:0] mop1;

  multopl      multopl      (*);
  multiplier   multiplier   (*);
  multoutreg   multoutreg   (*);
  barrel_shifter barrel_shifter (*);
  mux2         mux         (.y(alu_in), .i0(multout),
                          .i1(acc), .sel1(calu_muxsel));

  alu          alu         (*, .zero(), .neg());
  accumulator  accumulator (*);
  shifter      shifter     (*);
  tribuf       tribuf      (*, .acc(acc[15:0]));
endmodule

```

```

module mux2 (
  output [31:0] y,
  input  [31:0] i1, i0,
  input          sel1);
...

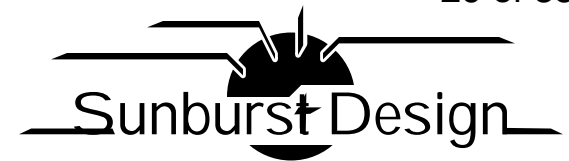
```

None of the port names match the nets connected to the ports

All ports must be connected by name

# SystemVerilog .\* Implicit Ports

(A Closer Look - 3/4)



```

module calu2 (
  inout  [15:0] data,
  input  [ 4:0] bs_lshft,
  input  [ 2:0] alu_op,
  input  [ 1:0] shft_lshft,
  input          calu_muxsel, en_shft, ld_acc, ld_bs,
  input          ld_multopl, ld_multout, ld_shft, en_acc,
  input          clk, rst_n);

  wire  [31:0] acc, alu_in, alu_out, bs, mult, multout;
  wire  [15:0] mop1;

  multopl      multopl      (*);
  multiplier   multiplier   (*);
  multoutreg   multoutreg   (*);
  barrel_shifter barrel_shifter (*);
  mux2         mux         (.y(alu_in), .i0(multout),
                          .i1(acc), .sel1(calu_muxsel));

  alu          alu          (*, .zero(), .neg());
  accumulator  accumulator  (*);
  shifter      shifter      (*);
  tribuf       tribuf       (*, .acc(acc[15:0]));
endmodule

```

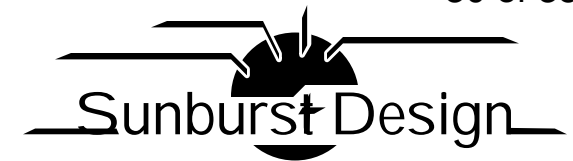
```

module alu (
  output [31:0] alu_out,
  output          zero, neg,
  input  [31:0] acc, alu_in,
  input  [ 2:0] alu_op);
...

```

All ports with matching names and port sizes are connected to the matching nets

Unconnected ports must be explicitly listed



# SystemVerilog .\* Implicit Ports

(A Closer Look - 4/4)

```

module calu2 (
  inout  [15:0] data,
  input  [ 4:0] bs_lshft,
  input  [ 2:0] alu_op,
  input  [ 1:0] shft_lshft,
  input          calu_muxsel, en_shft, ld_acc, ld_bs,
  input          ld_multopl, ld_multout, ld_shft, en_acc,
  clk, rst_n);

```

32-bit bus

```

wire [31:0] acc, alu_in, alu_out, bs, mult, multout;
wire [15:0] mop1;

```

```

multopl      multopl      (*);
multiplier   multiplier   (*);
multoutreg   multoutreg   (*);
barrel_shifter barrel_shifter (*);
mux2         mux          (.y(alu_in), .i0(multout),
                          .i1(acc), .sel1(calu_muxsel));

alu          alu          (*, .zero(), .neg());
accumulator  accumulator (*);
shifter      shifter      (*);
tribuf       tribuf       (*, .acc(acc[15:0]));
endmodule

```

```

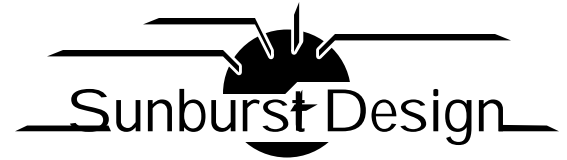
module tribuf #(parameter SIZE=16)
  (output [SIZE-1:0] data,
   input  [SIZE-1:0] acc,
   input          en_acc);
  ...

```

16-bit port

Ports with matching names  
but different sizes must be  
connected by name

All other ports can be  
connected implicitly



# Rules for Implicit `.*` Port Connections

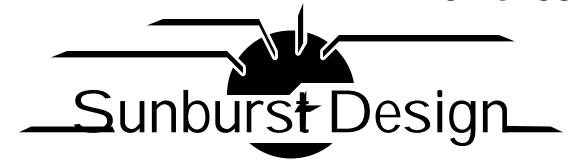
(Proposed - Not Official Yet)

- Rule: mixing `.*` and `.name` ports in the same instantiation is prohibited

As noted on an earlier slide

- Permitted: `.*` and `.name(signal)` connections in the same instantiation
- Rules: the `.name(signal)` connections are required for:
  - size-mismatch
  - name-mismatch
  - unconnected ports

Same rule as was shown for  
`.name` port connections



# Advantages of Implicit .\* Port Connections

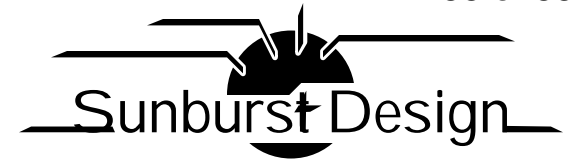
(Proposed - Not Official Yet)

- Top-level designs are generally just verbose connections
- Port order is not important
- Still has all of the advantages of explicit port connections
- Not as verbose as either `.name` or explicit port connections
- Easier to scan a top-level design with 100's of instantiated modules and 1,000's of ports

The `.*` notation removes all of the unnecessary verbosity

All ports are implicitly connected

Only exceptions are listed and stand out



# Advantages of Implicit .\* Port Connections

(Proposed - Not Official Yet)

- The .\* notation facilitates quick generation of block-level testbenches

**Just make connections from the testbench to the module with the same names**

- Debugging the .\* notation should not be a problem

**The Verilog compiler makes the connections so the port names should exist in the database**

**One company with an in-house HDL language has already implemented and successfully used this capability for years**

**Too much emphasis is placed on declarations and port-name visibility**

**For most designers of large ASICs, the top-level module is a painful and necessary evil to satisfy the requirements of a simulator**