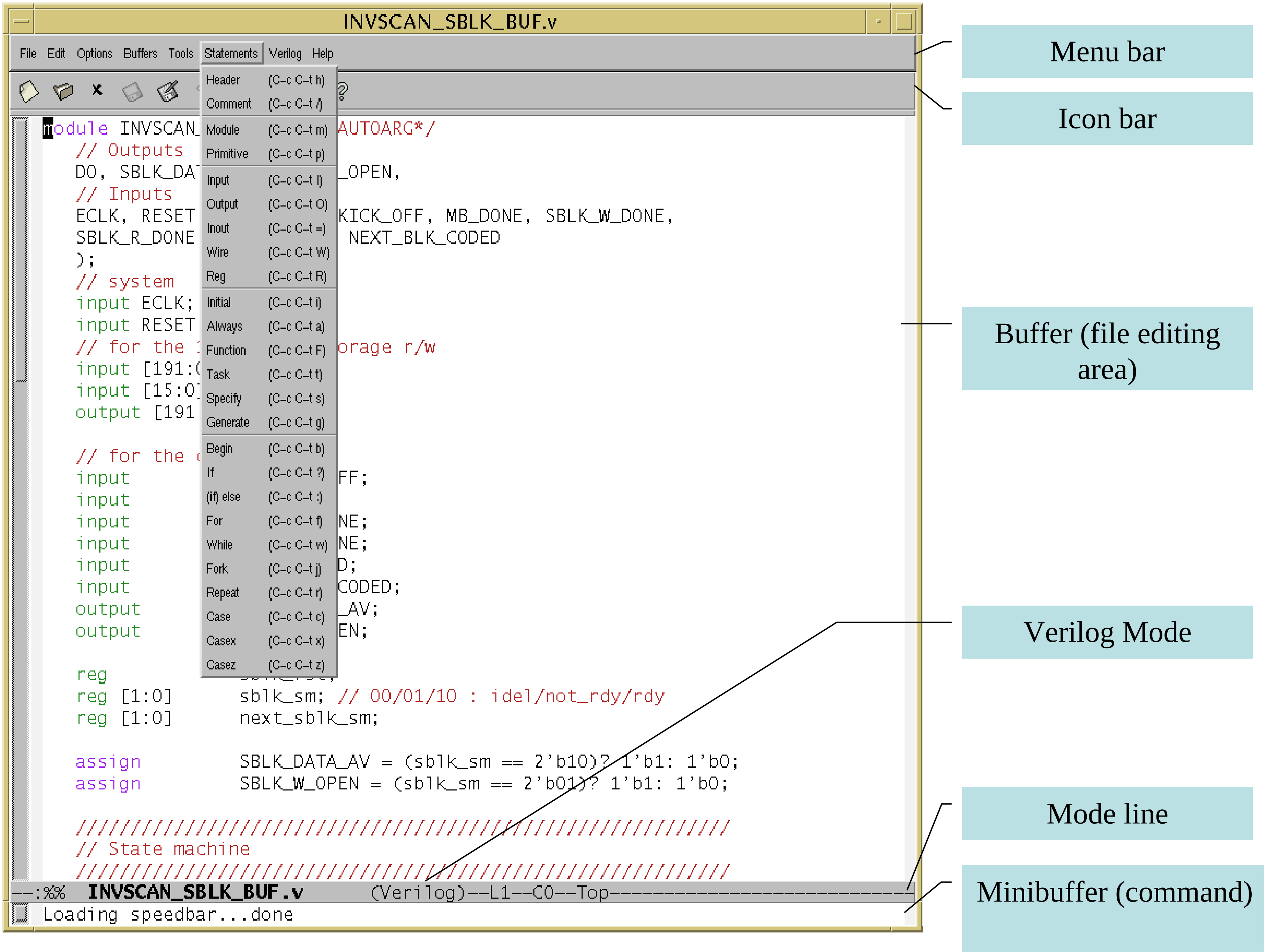
“emacs” & verilog-mode

By Chenggang Liu

Introduction to “emacs”

- What is “emacs”
  - – Emacs is a full featured editor (like “vi”)
  - – Emacs is a extensible editor (more than fundamental editing, can add minor mode to help different kind of editings, like Verilog Code, perl scripts, C program etc.)
  - – Emacs is a free software
  - – Eamcs can be customized to reflect your personal need.
- A real “emacs” looks like this…



```
module INVSCAN_                  AUTOARG*/
   // Outputs
   DO, SBLK_DA                   _OPEN,
   // Inputs
   ECLK, RESET                   KICK_OFF, MB_DONE, SBLK_W_DONE,
   SBLK_R_DONE                    NEXT_BLK_CODED
   );
   // system
   input ECLK;
   input RESET
   // for the                    orage r/w
   input [191:(
   input [15:0]
   output [191

   // for the
   input                         FF;
   input
   input                         NE;
   input                         NE;
   input                         D;
   input                         CODED;
   output                        _AV;
   output                        EN;

   reg
   reg [1:0]      sblk_sm; // 00/01/10 : idel/not_rdy/rdy
   reg [1:0]      next_sblk_sm;

   assign         SBLK_DATA_AV = (sblk_sm == 2'b10)? 1'b1: 1'b0;
   assign         SBLK_W_OPEN = (sblk_sm == 2'b01)? 1'b1: 1'b0;

   ////////////////////////////////////////////////////////////
   // State machine
   ////////////////////////////////////////////////////////////
--:%%  INVSCAN_SBLK_BUF.v        (Verilog)--L1--C0--Top----------------------
Loading speedbar...done
```

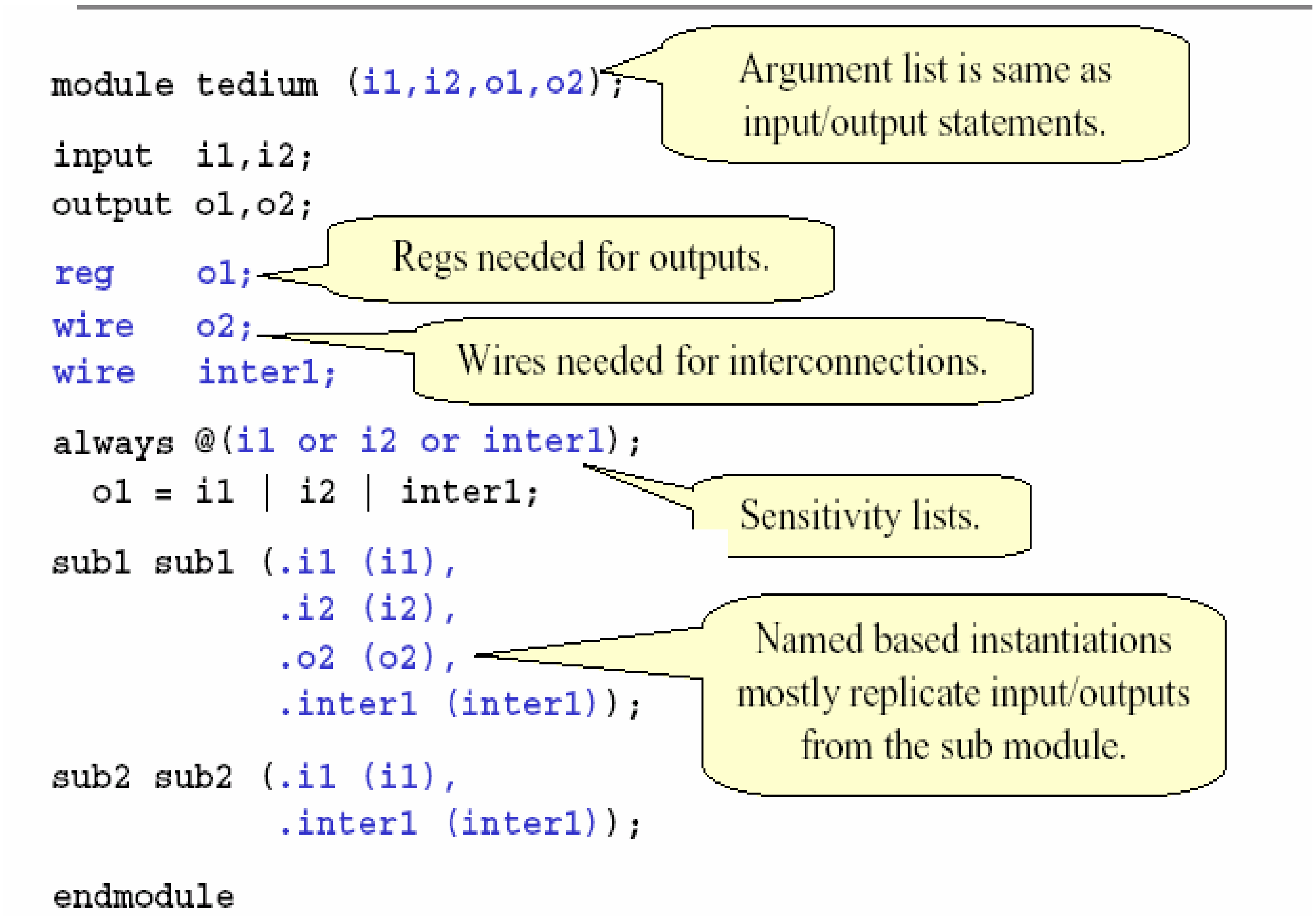
How to invoke it

- Search Path
  - Unix & Linux (.cshrc)
    - Thanks to our IT group, latest “emacs” has been installed on Unix and every Linux machines.
    - Make sure /usr/local/bin & /usr/bin is in your $path variable
    - Make sure /unixtools/ut/external/export…… not in your $path
  - PC
    - Copy it to your HD
    - Change to $copy_dir/emacs21.2/bin → runemacs.exe
- .emacs
  - A file to customized you “emacs”, like printer, minor mode binding… etc
  - You can copy my as a sample:
    - /u/cgliu/.emacs

Basic Operation

Action	Keystrokes	Menu Bar
File		
Open	C-x C-f	*File->Open File*
Save	C-x C-s	*File->Save Buffer*
Save as	C-x C-w	File->Save buffer as
Close/Kill	C-x C-k	File->Close
Exit	C-x C-c	File->Exit Emacs
Moving		
Anywhere		Left Mouse Clicking
Quick Scroll		Right Mouse Clicking & Dragging on scroll bar
Scroll forward page	C-v	
Scroll backward page	M-v	
Editing		
Select Region		Left Click + Drag
Copy	Esc w	Edit->Copy
Cut	C-w	Edit->Cut
Paste	C-y	Edit->Paste
Undo	C-u	Edit->Undo
Windows		
Delete other window	C-x 1	Files->Unsplit Windows
Frame		
Make new frame	C-x 5 2	Files->New Frame
Delete frame	C-x 5 0	Files->Delete Frame

Verilog-mode



```
module tedium (i1,i2,o1,o2);

input  i1,i2;
output o1,o2;

reg    o1;
wire   o2;
wire   inter1;

always @(i1 or i2 or inter1);
  o1 = i1 | i2 | inter1;

sub1 sub1 (.i1 (i1),
           .i2 (i2),
           .o2 (o2),
           .inter1 (inter1));

sub2 sub2 (.i1 (i1),
           .inter1 (inter1));

endmodule
```

Argument list is same as input/output statements.

Regs needed for outputs.

Wires needed for interconnections.

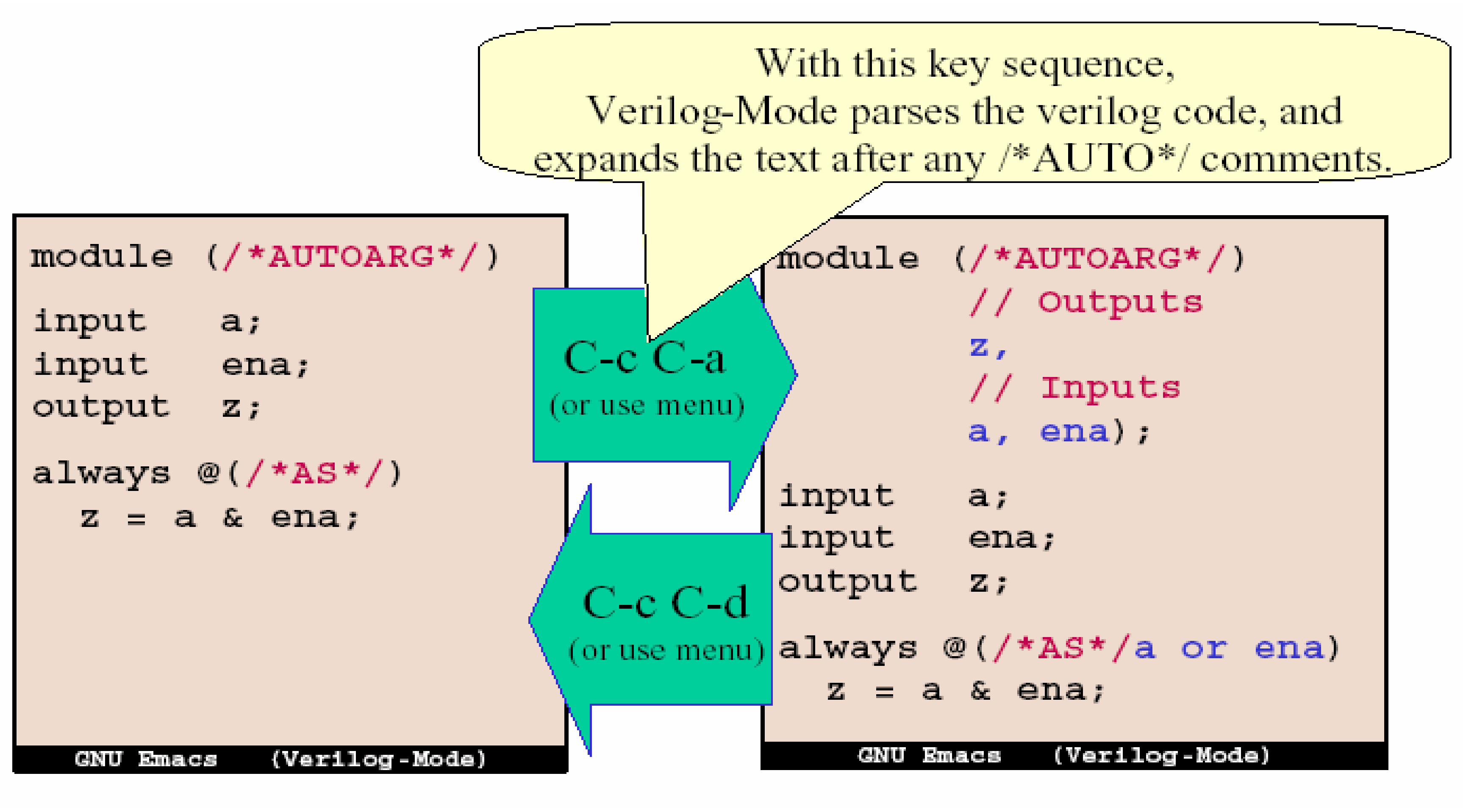
Sensitivity lists.

Named based instantiations mostly replicate input/outputs from the sub module.

Verilog-mode Features

- Argument list
- Wires
- Regs
- Sensitivity list
- Instantiations

C-c C-a and C-c C-k



With this key sequence, Verilog-Mode parses the verilog code, and expands the text after any /*AUTO*/ comments.

```
module (/*AUTOARG*/)

input   a;
input   ena;
output  z;

always @(/*AS*/)
  z = a & ena;
```

GNU Emacs (Verilog-Mode)

C-c C-a

(or use menu)

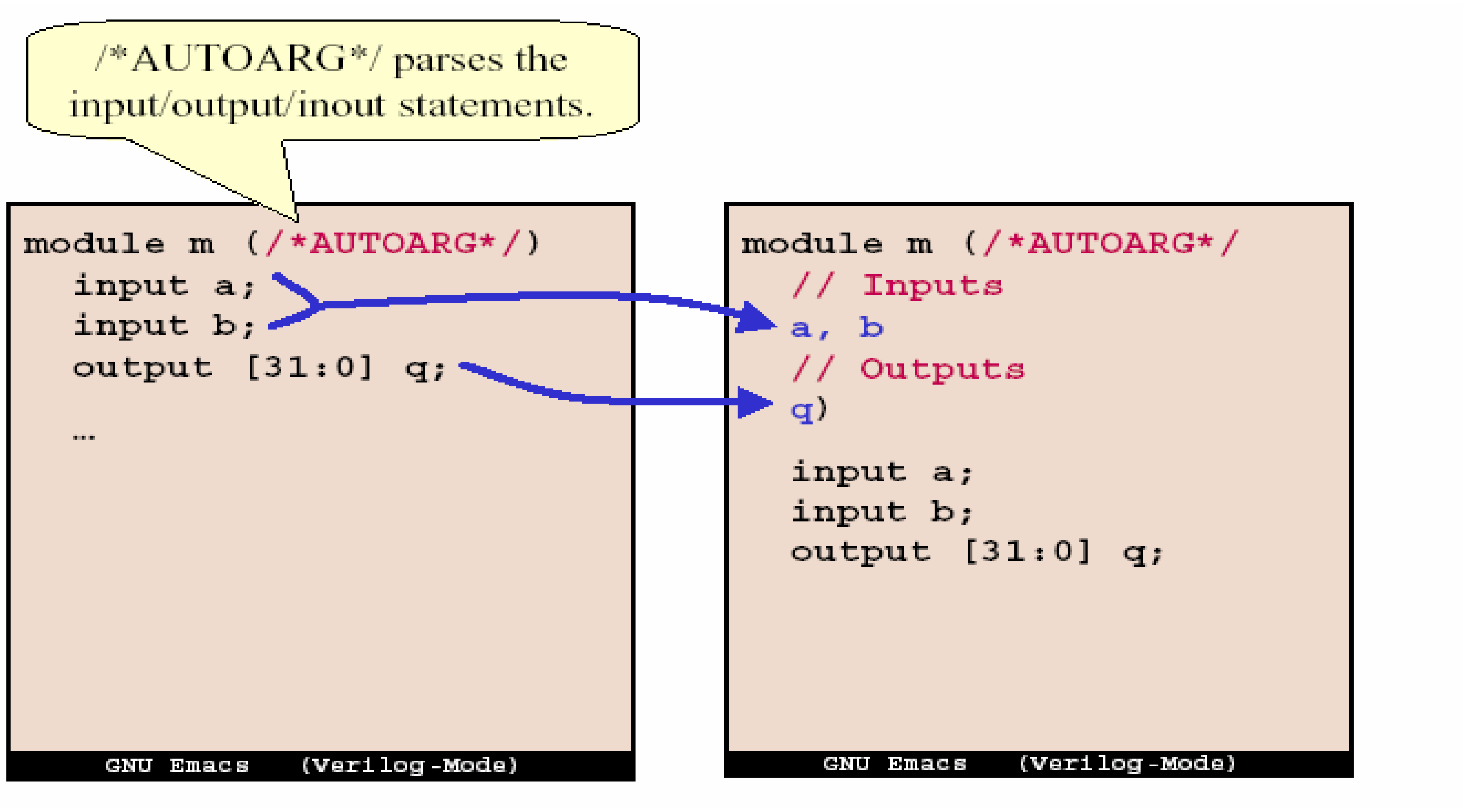
C-c C-d

(or use menu)

```
module (/*AUTOARG*/)
        // Outputs
        z,
        // Inputs
        a, ena);

input   a;
input   ena;
output  z;

always @(/*AS*/a or ena)
  z = a & ena;
```

GNU Emacs (Verilog-Mode)

Argument lists

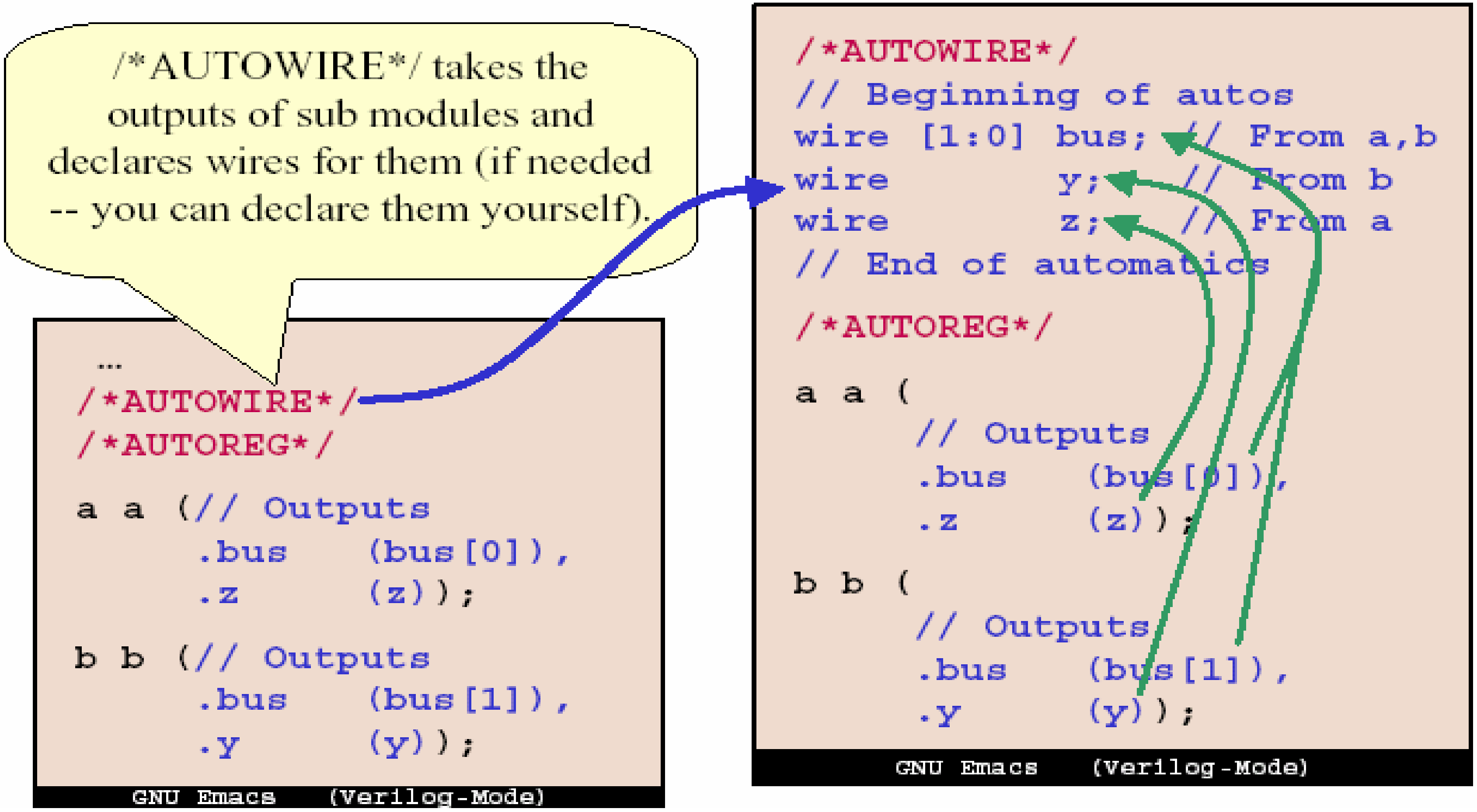


/*AUTOARG*/ parses the input/output/inout statements.

```
module m (/*AUTOARG*/)
   input a;
   input b;
   output [31:0] q;

   …
```

```
module m (/*AUTOARG*/
   // Inputs
   a, b
   // Outputs
   q)

   input a;
   input b;
   output [31:0] q;
```

Automatic Wires

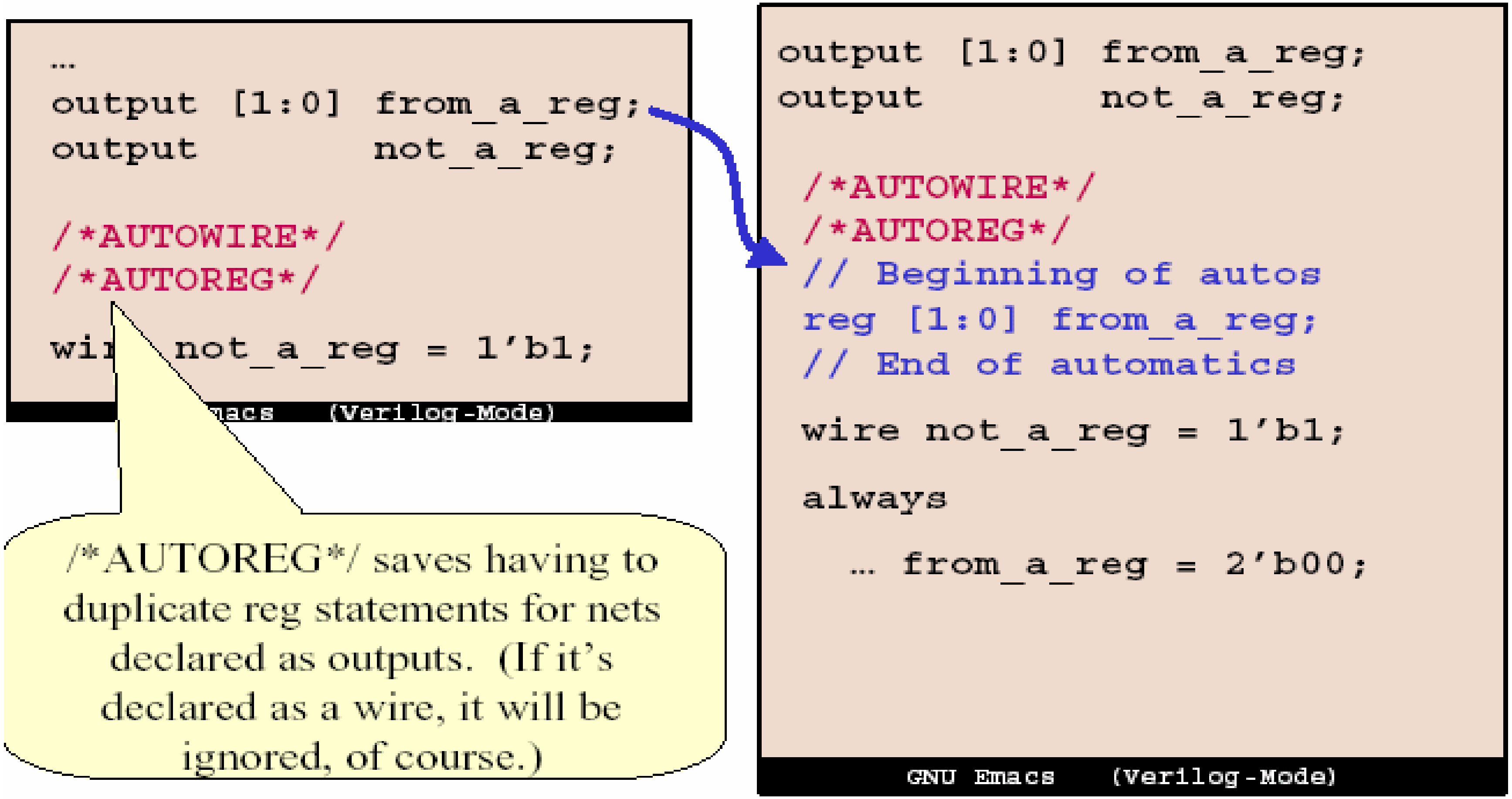


/*AUTOWIRE*/ takes the outputs of sub modules and declares wires for them (if needed -- you can declare them yourself).

```
...
/*AUTOWIRE*/
/*AUTOREG*/

a a (// Outputs
       .bus    (bus[0]),
       .z      (z));

b b (// Outputs
       .bus    (bus[1]),
       .y      (y));
```

```
/*AUTOWIRE*/
// Beginning of autos
wire [1:0] bus; // From a,b
wire         y; // From b
wire         z; // From a
// End of automatics
/*AUTOREG*/

a a (
     // Outputs
     .bus    (bus[0]),
     .z      (z));

b b (
     // Outputs
     .bus    (bus[1]),
     .y      (y));
```

Automatic Registers

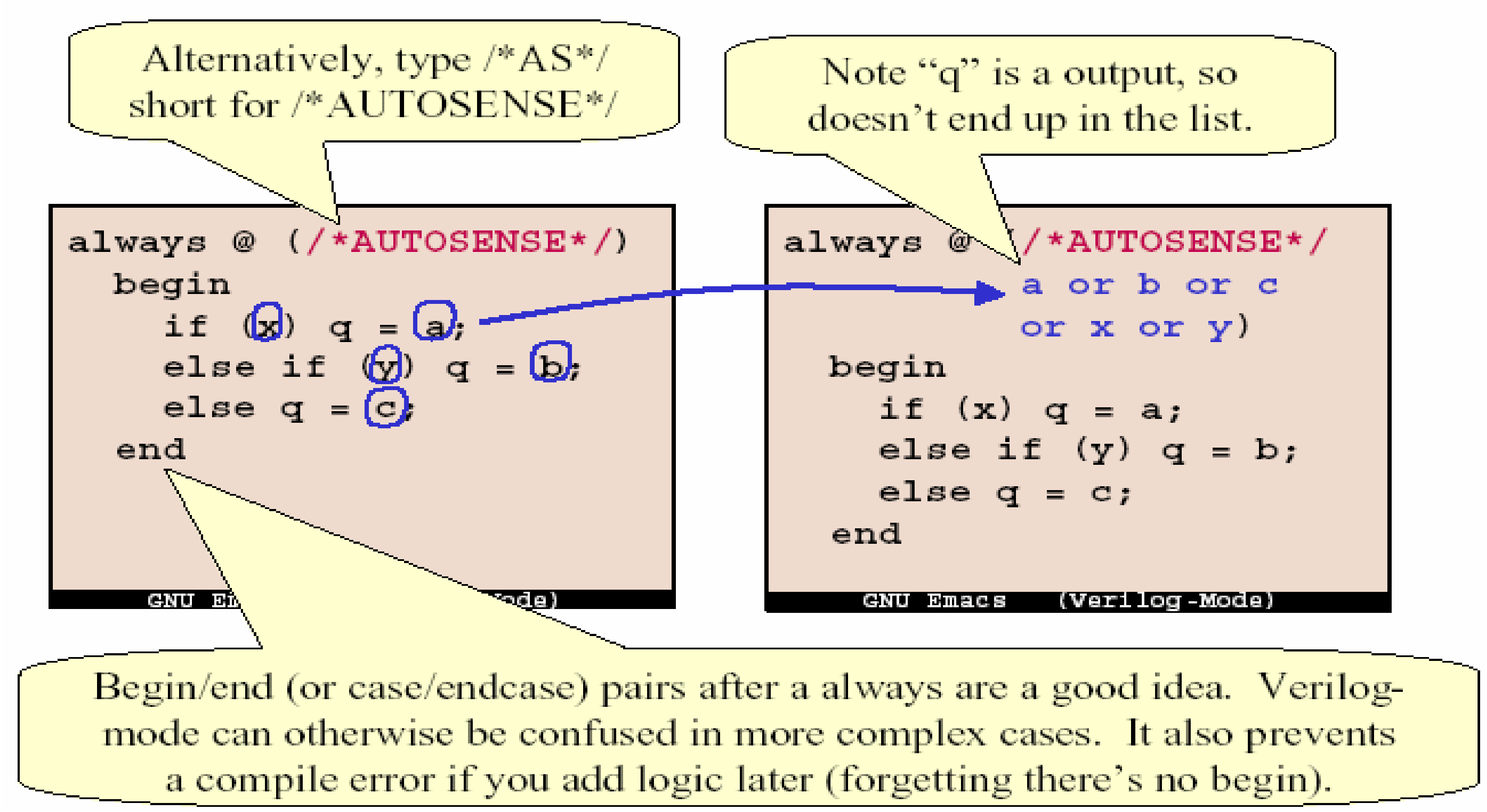


```
…
output  [1:0]  from_a_reg;
output         not_a_reg;

/*AUTOWIRE*/
/*AUTOREG*/

wi     not_a_reg = 1’b1;
```

/*AUTOREG*/ saves having to duplicate reg statements for nets declared as outputs. (If it’s declared as a wire, it will be ignored, of course.)

```
output  [1:0]  from_a_reg;
output         not_a_reg;

 /*AUTOWIRE*/
 /*AUTOREG*/
 // Beginning of autos
 reg  [1:0]  from_a_reg;
 // End of automatics

 wire not_a_reg = 1’b1;

 always

   … from_a_reg = 2’b00;
```

Sensitivity Lists



Alternatively, type /*AS*/ short for /*AUTOSENSE*/

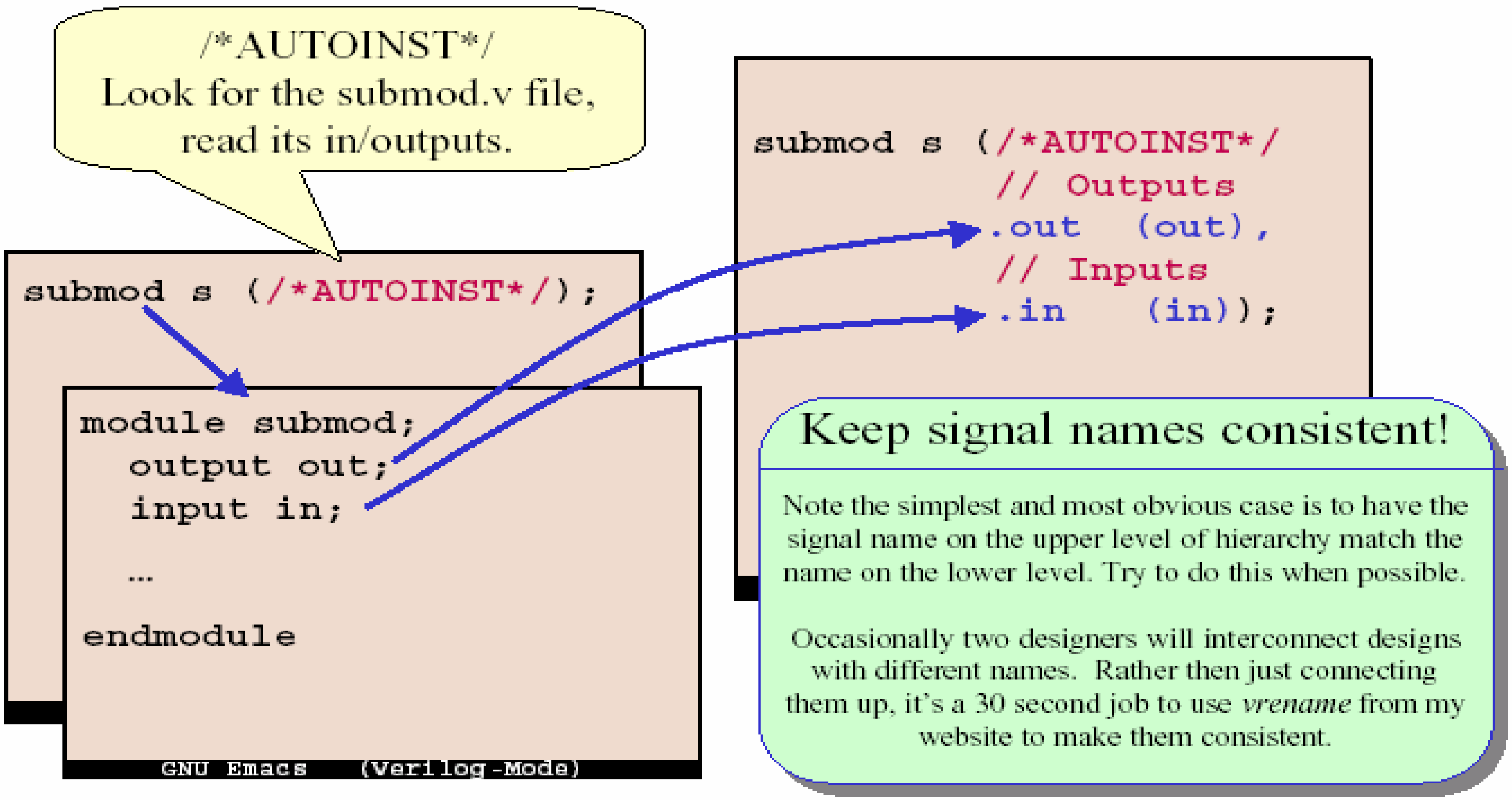
Note “q” is a output, so doesn’t end up in the list.

```
always @ (/*AUTOSENSE*/)
  begin
    if (x) q = a;
    else if (y) q = b;
    else q = c;
  end
```

```
always @ (/*AUTOSENSE*/
          a or b or c
          or x or y)
  begin
    if (x) q = a;
    else if (y) q = b;
    else q = c;
  end
```

Begin/end (or case/endcase) pairs after a always are a good idea. Verilog-mode can otherwise be confused in more complex cases. It also prevents a compile error if you add logic later (forgetting there’s no begin).

Simple Instantiations



/*AUTOINST*/
Look for the submod.v file,
read its in/outputs.

```
submod s (/*AUTOINST*/);
```

```
module submod;
   output out;
   input in;

   ...

endmodule
```

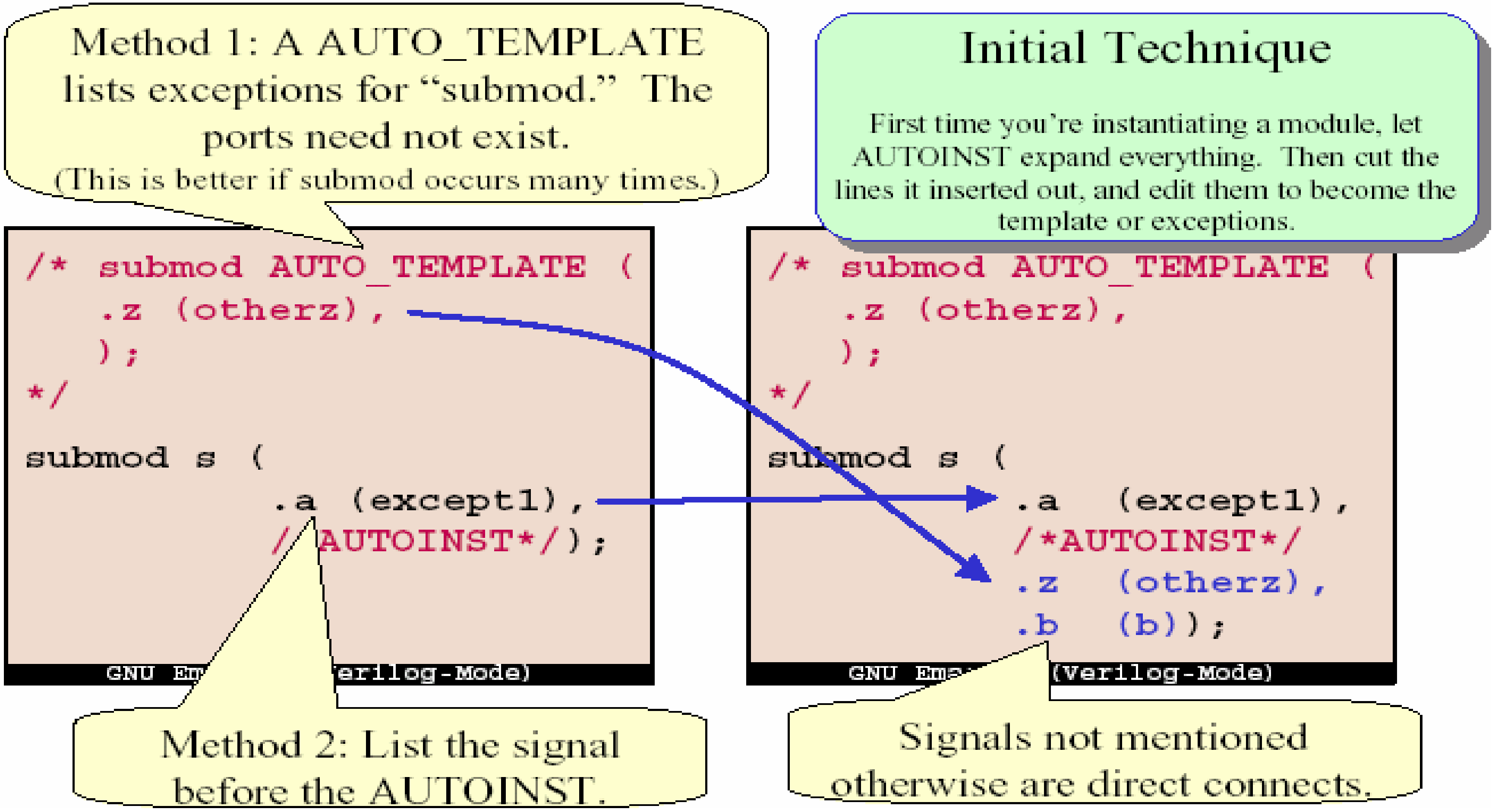
```
submod s (/*AUTOINST*/
          // Outputs
          .out   (out),
          // Inputs
          .in    (in));
```

Keep signal names consistent!

Note the simplest and most obvious case is to have the signal name on the upper level of hierarchy match the name on the lower level. Try to do this when possible.

Occasionally two designers will interconnect designs with different names. Rather then just connecting them up, it's a 30 second job to use *vrename* from my website to make them consistent.

Exceptions to Instantiations



Method 1: A AUTO_TEMPLATE lists exceptions for "submod." The ports need not exist.
(This is better if submod occurs many times.)

Initial Technique

First time you're instantiating a module, let AUTOINST expand everything. Then cut the lines it inserted out, and edit them to become the template or exceptions.

```
/* submod AUTO_TEMPLATE (
    .z (otherz),
    );
*/

submod s (
          .a (except1),
          /*AUTOINST*/);
```

```
/* submod AUTO_TEMPLATE (
    .z (otherz),
    );
*/

submod s (
          .a   (except1),
          /*AUTOINST*/
          .z   (otherz),
          .b   (b));
```

Method 2: List the signal before the AUTOINST.

Signals not mentioned otherwise are direct connects.

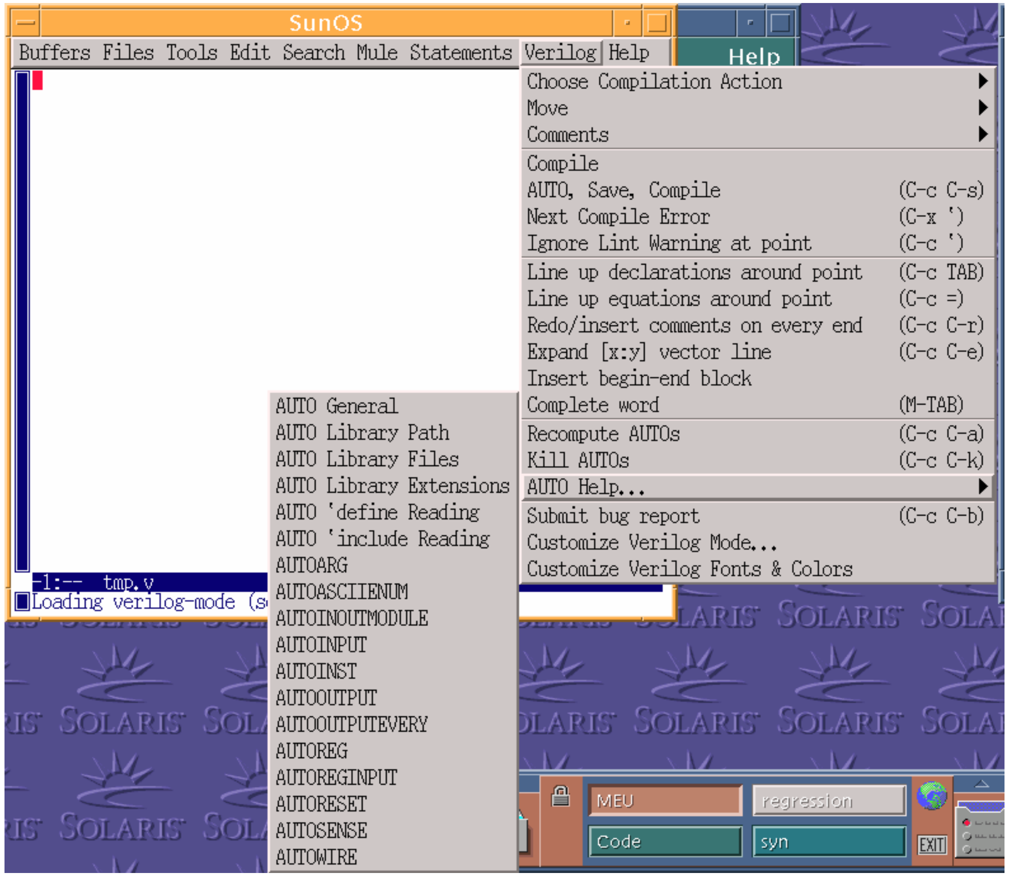
Making upper level modules

- Building a null or shell modules
  - You want a module with same input/output list as another module
  - /*AUTOINOUTMODULE(“modulename”)*/
- Output/input all sinals
  - You have a shell with outputs every thing
  - /*AUTOOUTPUT*/
  - /*AUTOINPUT*/

Benefit

- Reduce sensitivity problem
- Make it easier to name signals consistently through the hierarchy
- Less code to “look” at
- Less time typing
- Less error

Verilog AUTO Help


SunOS
Buffers Files Tools Edit Search Mule Statements Verilog Help
Help
Choose Compilation Action
Move
Comments
Compile
AUTO, Save, Compile (C-c C-s)
Next Compile Error (C-x ')
Ignore Lint Warning at point (C-c ')
Line up declarations around point (C-c TAB)
Line up equations around point (C-c =)
Redo/insert comments on every end (C-c C-r)
Expand [x:y] vector line (C-c C-e)
Insert begin-end block
Complete word (M-TAB)
Recompute AUTOs (C-c C-a)
Kill AUTOs (C-c C-k)
AUTO Help...
Submit bug report (C-c C-b)
Customize Verilog Mode...
Customize Verilog Fonts & Colors
AUTO General
AUTO Library Path
AUTO Library Files
AUTO Library Extensions
AUTO 'define Reading
AUTO 'include Reading
AUTOARG
AUTOASCIIENUM
AUTOINOUTMODULE
AUTOINPUT
AUTOINST
AUTOOUTPUT
AUTOOUTPUTEVERY
AUTOREG
AUTOREGINPUT
AUTORESET
AUTOSENSE
AUTOWIRE
-1:-- tmp.v
Loading verilog-mode (s
MEU
regression
Code
syn
EXIT

- More help http://www.delorie.com/gnu/docs/emacs/emacs_toc.html
- Wish you enjoy “emacs” & “verilog-mode”