Ten Edits I Make Against Most IP
(And Wish I Didn’t Have To…)

Wilson Snyder
wsnyder@wsnyder.org

September 18, 2007
Agenda

• IP In Our Last Project
• The Ten Edits
• Extra Credit
• Conclusions
• Q&A
IP in our last project

• For What?
  – Custom processor chip for cluster-in-a-box
    5832  Gigaflops
    7776  Gigabytes DDR memory
    972   6-core 64-bit nodes
    4     TBit/s bisection bandwidth
    18    KW

• What IP?
  – 9+ Vendors
  – Integrating them into one model took weeks-ish each
  – Thus this paper!
SiCortex Compute Node Layout

- CPU
- L2
- DMA
- Fabric Switch
- CPU
- DDR0
- DDR1
- PCI Express
- Clocks + Misc
- Misc
- Fabric PHY

Legend:
- Orange: IP
- Yellow: Modified IP
The Edits
0. Encrypted Verilog: Just Say No

• Vendors try to give you only encrypted Verilog.

• Problem?
  – Something breaks, can you trust the vendor’s response time?
  – Tool or even only a tool version change voids the encryption.
  – What do you do 3 years from now when a bug is found?
  – All of these delay your project!
  – And you need to apply the rest of this presentation? 😊

• Solution?
  – Refuse encrypted Verilog or choose another vendor.
  – Make sure your contract states non-encrypted up front.
  – If it’s a support issue, use the encrypted for support calls, but have the unencrypted version.
1. De-”Do”

• What’s wrong with this?

    module buffer (  
        input di,  
        output do);

• Problem?
  – “do” is a SystemVerilog keyword  
  – Breaks all SystemVerilog tools.

• Solution?
  – Use vrename program to replace “do”, “ref”, “bit”, etc.
2. Split libraries into one file per module

- Vendors ship a library file with many modules in it.

- Problem?
  - Requires a custom input.vc file that has “–y library”
    - That’s a pain, and we get warnings if it isn’t used
  - We can’t lint or compile a single module from inside the library

- Solution?
  - Split it into many files
  - Make one file per module
  - Make module name match the filename
  - vsplitmodule will fix this
3. Prefix module names

• What’s wrong with this?
  Vendor A:
  ```verilog
  module inv;
  ```
  Vendor B:
  ```verilog
  module inv;
  ```

• Problem?
  – Module names must be unique across all IP!
  – Given an error message mentioning a module, what vendor created that module?

• Solution?
  – Prefix all module names with vendor prefix
  – For Verilog, vrename will fix this
  – For GDS, use Hercules’ gdsin and gdsout
4. Convert to synthesis `ifdefs

• What’s wrong with this?
  
  // synopsys translate_off
  $display (...)
  // synopsys translate_on

• Problem?
  – You can’t select what code the simulation/ lint tools will see.
  – You may have mismatched off/on pairs

• Solution?
  – Use Ifndefs
    `ifndef VENDOR_SYNTHESIS
    – vsplitmodule will fix this
    – Note synthesis off’s aren’t needed at all around PLI calls!
5. Add header guards

- Vendors include a header once in the “top” file.

- Problem?
  - If we try to read only lower level modules, we hit missing defines
  - We could `include the file before every module, but this causes redefinition errors.

- Solution?
  - Add header guards around all includes, ala C++
    `ifndef _VENDOR_FILE_V
    `define _VENDOR_FILE_V     // uppercased filename
    ...
    `endif // Guard
6. Add common include

• Vendor doesn’t have any common includes

• Problem?
  – We need to globally turn off warnings or set some defines.
  – We can’t change the vendor’s simulation timescale

• Solution?
  – All files with modules should include a vendor header,
    `include "vendor_defines.v"
  – And a vendor_timescale.v right before the module statement.
    `include "vendor_defines.v"
    `include "vendor_timescale.v"
    module vendor_mod...
7. Make modules lint clean

• Vendor never runs any lint program
  – Or, assumes you only run it on the top level

• Problem?
  – We want a “lint clean” entire chip!
  – It’s ok to disable some warnings

• Solution?
  – Make all files lint clean, generally by finding all lint warnings
    and turning them off on a file-by-file basis.
  – The big violators are the next few issues…
8. Casex to casez

- What’s wrong with this?
  ```
  casex (sig)
  3’b1xx: ...;
  ```

- Problem?
  - Casex statements don’t work right with unknowns.

- Solution?
  - “grep casex”
  - Replace with casez:
    ```
    casez (sig)
    3’b1??: ...;
    ```
9. Fix width violations

• What’s wrong with this?

```vhdl
wire [15:0] a;
wire [16:0] b = a+1;
```

• Problem?
  – This is a width violation…
    • [17bits] = [16bits] + [32 bits]
  – Results in lint warnings, and may hide errors
    (if a were redeclared as 33 bits wide)

• Solution?
  – Don’t turn off the warnings! Fix it for ALL tools:
```vhdl
wire [16:0] b = {1'b0,a} + 17'd1;
```
10. Add unused_ok’s

• What’s wrong with this?

    wire never_used = ...;

• Problem?
  – Lint warnings about unused signals.
  – Hard to know if the warnings are from something we broke
    hooking up the IP, or an IP internal signal that can be ignored.

• Solution?
  – Concat all unused nets in one obvious place per module.

    wire _unused_ok = &{1'b0,
    never_used,
    1'b0};

  – This doesn’t need a translate off, and won’t toggle!
Extra Credit
Add Verilog-Mode AUTOs

• If I’m going to edit the IP, I’d like to have the Verilog-Mode Automatics already there.

• Problem?
  – It’s a pain to maintain the instance pin lists, etc.

• Solution?
  – Use `C-cC-z` in Emacs.
  – Maybe reindent, too.

```verilog
/*AUTOWIRE*/
// Beginning of autos
wire [1:0] bus; // From a,b
wire y; // From b
wire z; // From a
// End of automatics
a a (/*AUTOINST*/
  // Outputs
  .bus (bus[0]),
  .z (z));
```

GNU Emacs (Verilog-Mode)
Conclusions

• The Code compiles
  – 0. Not encrypted
  – 1. De-”Do”
  – 2. Split libraries into one file per module
  – 3. Prefix module names
  – 4. Convert to synthesis `ifdefs
  – 5. Add header guards
  – 6. Add common include

• It’s lint clean (ish)
  – 7. Make modules lint clean
  – 8. Casex to casez
  – 9. Fix width violations
  – 10. Add unused_ok’s

• We can easily edit it
  – Add Verilog-Mode AUTOs
Public Tool Sources

• The public domain design tools are available at http://www.veripool.com
  – Verilog-Mode - /*AUTO…*/ Expansion
  – Verilog-Perl – Renaming and splitting modules

• Additional Tools
  – Make::Cache - Object caching for faster compiles
  – Schedule::Load – Load Balancing (ala LSF)
  – SystemPerl - /*AUTOs*/ for SystemC
  – Verilator – Compile SystemVerilog into SystemC
  – Vregs – Extract register and class declarations from documentation