How to Shield Sections of VHDL Code from Synthesis

You can have Synergy ignore parts of your VHDL design by selectively disabling source-code translation during analysis. This is useful if you want to simulate, but not synthesize, parts of your design.

To shield sections of your VHDL code from synthesis, surround those sections of code with a pair of “meta_comments”. The meta_commented sections of code do not appear in the synthesized netlist.

For information about using meta_comments during your synthesis run, see the *synel_options* text command in the *Synergy VHDL Synthesizer and Optimizer Reference*.

Defining Meta_Comments in Your hdl.var File

You must define two meta_comment strings in your *hdl.var* file:

- A disable-synthesis meta_comment string

```
DEFINE SYN_META_XOFF <identifier1> [<identifier2>]
[<identifier3>]
```

```
DEFINE SYN_META_XOFF Synergy synthesis_off
```

identifier1 identifier2

- An enable-synthesis meta_comment string

```
DEFINE SYN_META_XON <identifier1> [<identifier2>]
[<identifier3>]
```

```
DEFINE SYN_META_XON  Synergy synthesis_on
```

identifier1 is required

identifier2 and *identifier3* are optional. If you specify more than three identifiers, the analyzer truncates the meta_comment string to three identifiers and issues a warning.

Identifiers are alphanumeric strings and must be separated by white space (tabs or spaces).

The following gives three examples of valid *hdl.var* file meta_comment definitions:

```
DEFINE SYN_META_XOFF Synergy translate_off
DEFINE SYN_META_XON   Synergy translate_on

DEFINE SYN_META_XOFF Disable
DEFINE SYN_META_XON   Enable
```

Using Meta_Comments in Your RTL Source Code

Place meta_comments around the lines of code you do not want synthesized. Meta_comment strings in your design only work if those strings have been defined in your *hdl.var* file:

- To disable synthesis — insert the VHDL comment token (two dashes, --) followed by the SYN_META_XOFF meta_comment string immediately above the first line of code you want Synergy to skip.
- To re-enable synthesis — insert the VHDL comment token (two dashes, --) followed by the SYN_META_XON meta_comment string immediately below the last line of code you want Synergy to skip.

```
process (clk, reset) begin
-- synergy translate_off
   if reset = '0' then
      counter <= 0;
      elsif rising_edge(clk) then
      counter <= counter + 1;
   end if;
-- synergy translate_on
end process;
```

You can use multiple meta_comment pairs to disable disjoint sections of code.

You can place the meta_comment flag across design units in a single file — several design units can be disabled with one meta_comment pair. You cannot place the disabling and re-enabling meta_comment pairs in different files.

Troubleshooting Meta_Comments

If your meta_comments in your design don't work as you expect, check this troubleshooting list for a possible solution:

Problem: **Synthesis fails with an error message about meta_commented code.**

Potential Cause:

- **Syntax errors in the meta_commented VHDL code.**

  The meta_comments do not cause your VHDL code to be ignored completely. The meta_commented VHDL code undergoes partial analysis by *cv,* and syntax errors are flagged.

- **Meta_commented section of VHDL code is referenced in the unmasked section of code:**

```
--Synergy translate_off
   signal A : std_logic;
--Synergy translate_on
   signal B : std_logic;
   begin
      A <= B;
```

Problem: **The meta_commented section of code is synthesized.**

Potential Cause:

- ***hdl.var* file is missing definition of meta_comment string.**
- ***cv* invoked without turning on meta_comments.**

  You must inform the analyzer, *cv*, that your design file has meta_comments:

```
cv -SYN_META_XOFF
```

- **Synthesis invoked without turning on meta_comments.**

  You must inform Synergy that your design file has meta_comments:

```
synel_options -syn_meta_xoff
```

- **Meta_comment strings surround an incomplete statement, not a complete statement:**

  The metacomments must surround an entire statement, for example, an entire *if – endif* statement can be metacommented out or an entire statement internal to the *if – endif* statement can be metacommented out.:

<u>Works</u>

```
process (clk, reset) begin
if reset = '0' then
   counter <= 0;
   elsif
      rising_edge(clk) then
-- synergy synthesis_off
   counter <= counter + 1;
-- synergy synthesis_on
end if;
end process;
```

<u>Does Not Work</u>

```
process (clk, reset) begin
if reset = '0' then
   counter <= 0;
-- synergy synthesis_off
   elsif
      rising_edge(clk) then
   counter <= counter + 1;
-- synergy synthesis_on
end if;
end process;
```

- **Typographical error in meta_comment string.**

  The following lists show working and non-working syntax variations given a meta_comment string definition of:

```
DEFINE SYN_META_XOFF Synergy syn_off
```

<u>Works</u>

```
-- Synergy syn_off
--SyNeRgY sYn_OFF
A <= B; --Synergy syn_off
```

<u>Does Not Work</u>

```
-- --Synergy syn_off
-- Synergy syn_off extra
-- Synergy syn off
```

- **Design is missing one of the meta_comment string pair:**

  `-- Synergy syn_off` without `-- Synergy syn_on`

- ***hdl.var* comment character (#) is included in meta_comment string.**

  To use the *hdl.var* comment character (#) as part of your meta_comment string, you must surround the string with double quotation marks in your *hdl.var* file definition:

```
DEFINE SYN_META_XOFF "translate#off"
DEFINE SYN_META_XON  "translate#on"
```