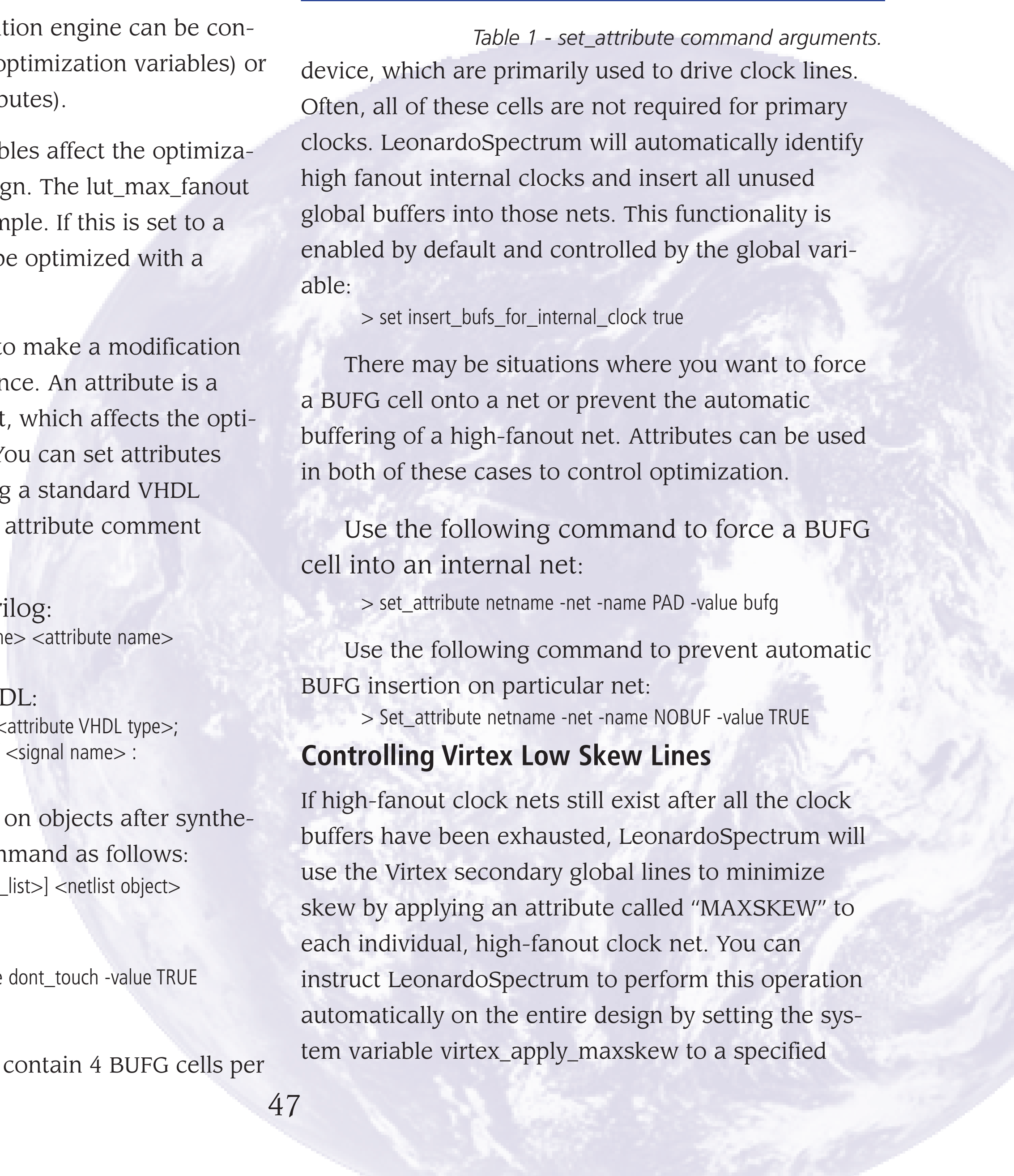
How to Control Virtex Design Optimization
USING VARIABLES AND ATTRIBUTES

With Exemplar's LeonardoSpectrum, you can easily control every aspect of your design.

by Tom Hill, Technical Marketing Manager, Exemplar Logic, tom.hill@exemplar.com



LeonardoSpectrum is the FPGA Synthesis Tool from Exemplar Logic, used to Synthesize RTL HDL code and target Xilinx Virtex devices.

Using Attributes and Variables

LeonardoSpectrum's optimization engine can be controlled globally (using global optimization variables) or at the netlist level (using attributes).

Global optimization variables affect the optimization of every block in the design. The lut_max_fanout variable provides a good example. If this is set to a value of 12 then all nets will be optimized with a fanout no greater than 12.

Attributes provide a way to make a modification to a specific net, cell, or instance. An attribute is a property assigned to an object, which affects the optimization of only that object. You can set attributes prior to RTL synthesis by using a standard VHDL attribute statement or Verilog attribute comment statement as follows:

- Setting Attributes in Verilog:
```
//Exemplar attribute <signal name> <attribute name>
<attribute value>
```
- Setting Attributes in VHDL:
```
ATTRIBUTE <attribute name> : <attribute VHDL type>;
ATTRIBUTE <attribute name> of <signal name> :
signal is <attribute value>;
```

Attributes can also be set on objects after synthesis using the set_attribute command as follows:

```
usage : "set_attribute" [<object_list>] <netlist object>
<attribute name> <value>
```

For example:

```
set_attribute u1 -instance -name dont_touch -value TRUE
```

Controlling Clock Buffers

Virtex and VirtexE FPGAs contain 4 BUFG cells per device, which are primarily used to drive clock lines. Often, all of these cells are not required for primary clocks. LeonardoSpectrum will automatically identify high fanout internal clocks and insert all unused global buffers into those nets. This functionality is enabled by default and controlled by the global variable:

```
> set insert_bufs_for_internal_clock true
```

Argument	Set_attribute Switch	Description
Netlist object	-port | -net | -instance	Indicates the type of netlist object
Attribute Name	-name	Name of the attribute
Attribute Value	-value	Value of the attribute

Table 1 - set_attribute command arguments.

There may be situations where you want to force a BUFG cell onto a net or prevent the automatic buffering of a high-fanout net. Attributes can be used in both of these cases to control optimization.

Use the following command to force a BUFG cell into an internal net:

```
> set_attribute netname -net -name PAD -value bufg
```

Use the following command to prevent automatic BUFG insertion on particular net:

```
> Set_attribute netname -net -name NOBUF -value TRUE
```

Controlling Virtex Low Skew Lines

If high-fanout clock nets still exist after all the clock buffers have been exhausted, LeonardoSpectrum will use the Virtex secondary global lines to minimize skew by applying an attribute called "MAXSKEW" to each individual, high-fanout clock net. You can instruct LeonardoSpectrum to perform this operation automatically on the entire design by setting the system variable virtex_apply_maxskew to a specified maximum skew value as follows:

```
> Set virtex_apply_maxskew 7
```

The "maxskew" attributes can be used to specify a maximum skew on a particular net. Use the following command to limit skew on a particular net:

```
> Set_attribute <internal netname> -net -name MAXSKEW
-value 7
```

If you set a very low skew value, one that the Xilinx Alliance Series software is unable to meet, then an error will be issued during place and route. For that reason it is important not to over constrain the skew value.

Controlling Fanout

Fanout, which is usually controlled globally, is set to a default of 64 in LeonardoSpectrum, for Virtex devices. This is generous but usually gives good results. If timing is difficult to meet in the place and route environment, especially if large routing delays are the culprit, then you can modify design fanout in certain areas. Using the lut_max_fanout attribute, LeonardoSpectrum allows you to override a global fanout specification with a unique value for a particular block or net. Net fanout can be specified on a global basis using the following command:

```
> set lut_max_fanout 12
```

Use the following command to redefine the fanout on a specific net from 64 (default) to 16:

```
> Set_attribute mynetname -net -name lut_max_fanout -value 16
```

Controlling IOB Registers

LeonardoSpectrum, by default, does not optimize registers to the IOB. When loading the Virtex technology, you are presented with an option to "Map IOB Registers," which will set the global optimization variable:

```
> set virtex_map_iob_registers TRUE
```

Once set this variable will instruct LeonardoSpectrum to pull all possible registers into the Virtex IOB blocks. When a single register is used to drive more than 1 output port LeonardoSpectrum will replicate that register once for each additional port. Often, however, users wish to pull only a few select registers into the IOB. This can be accomplished by assigning an attribute called "IOB" to the particular register instance.

Use the following command to force a register with an instance name called "reg_state(7)" into the IOB:

```
set_attribute reg_state(7) -instance -name IOB -value TRUE
```

You can also use wildcards "*" to set all the registers of a bus at once. For example:

```
"reg_state*"
```

Controlling Block RAM

Both global variables and attributes can be used to control block RAM inferencing. When developing a single module of a larger device you may wish to allocate all the block RAM resources to another block. In this case, using the global variable to disable the block RAM inference makes the most sense. You can do this by using the following command:

```
> set extract_ram FALSE
```

When block RAM resources start to run low, you may choose to disable block RAM inferencing on a section of the design. You can accomplish this by assigning the block_ram attribute to the storage signal used during the RTL RAM inference. You can do this in the RTL code by setting an attribute on the memory signal.

```
Verilog Example:
Reg [7:0] mem[63:0]
//Exemplar attribute mem block_ram FALSE
VHDL Example:
TYPE mem_type IS ARRAY 0 TO 256 OF std_logic_vector
(15 DOWNTO 0);
SIGNAL mem : mem_type;
ATTRIBUTE block_ram : Boolean;
ATTRIBUTE block_ram of mem : signal is TRUE;
```

The block_ram attribute can also be applied to the "mem" signal, after synthesis, from the LeonardoSpectrum command line. LeonardoSpectrum's design browser can help you identify correct signal pathnames for hierarchical blocks. For flat designs the signal name alone would be sufficient. Below is an example of using the set_attribute command to disable RAM inferencing on a memory signal "mem" that resides within a sub-block of the design "blockA:"

```
> Set_attribute .work.blockA.rtl.mem* -net -name block_ram -
value false
```

Conclusion

With LeonardoSpectrum, it's easy to optimize Virtex FPGA designs. For more information about LeonardoSpectrum, see www.exemplar.com.