

Summary

This application note shows the tradeoffs that can be made to gain the greatest possible densities and speeds for schematic, behavioral, and VHDL implementations.

Xilinx Family

XC9500

Introduction

The advanced architecture of the XC9500 CPLDs can implement a wide range of design densities and speeds, while allowing significant design changes to be made without modifying the original pinouts. The Xilinx XACTstep software (fitter) makes full use of this flexibility, giving designers a number of easy to use options for optimizing design performance.

Optimizing Density

The following techniques can be used for those designs requiring the maximum amount of device resources.

Controlling the Macrocell Input and Pterm Limits

The Xilinx XACTstep design software has options for setting the limits of the fitting algorithms. Two key options are input collapsing and product term collapsing. The defaults are 36 inputs and 15 p-terms.

The 36 input limit restricts the fitter from collapsing logic functions that will require more than 36 inputs. This is

based on the actual number of input signals delivered into each function block (FB). Although this is a physical limit, it is possible to exceed it by permitting the software to wire AND signals in the FastCONNECT switch matrix. The software will detect and utilize wire ANDing whenever it is required. This will allow the product of several signals to be used as inputs to the FB on a single input line.

Similarly, the default of 15 p-terms restricts the fitter from collapsing logic functions that would use more than 15 product terms. This limits the size and the cascade timing penalty. Designers may adjust these two options to achieve improved fitting.

When a design fails to fit, a report is created, summarizing the unmapped logic and pins. This may be used as a guide for adjusting the input collapsing limit and the product term collapsing limit. The easiest approach is to experiment with different values. For example, when a design fails to fit, as shown in [Figure 1](#), adjust the input collapsing limit and run another pass with the fitter. This will result in a successful fit or a new report of unmapped logic and pins.

```
*****Resources Required by Unmapped Logic and Pins*****
** Logic **
Signal          Total   Signals   Pwr    Slew
Name            Pt      Used     Mode   Rate
$18N1345        12      19       STD
DPCS            10      19       STD
IO_CYC_DET      10      14       STD
LBE1_N          10      13       STD
SPEC_CYC        10      14       STD
*****Function Block Resource Summary*****
Function  # of   FB Inputs  Signals  Total  O/IO  IO
Block    Macrocells  Used    Used    Pt Used  Req  Avail
FB1       13         36      39      47    0/0   17
FB2       12         36      37      56   10/0   17
FB3       14         36      36      73    5/1   17
FB4       11         36      38      84    7/0   17
FB5       12         36      40      84    8/1   17
FB6       10         36      38      83    7/0   16
FB7       12         35      35      83   10/1   16
FB8       10         36      37      43    5/3   16
----
          94                    553    52/6   133
```

Figure 1: Report Showing a Design That Failed to Fit

If the list of unmapped resources appears to be growing, either raise or lower the limit for the next attempted fit. At some point, the list will not improve, indicating that the best ratio has been found. Then, adjust the product term collapsing limit and repeat the process to find the optimum ratio. Being systematic about the process ensures the quickest convergence. **Figure 4** shows a report file for a design that fit after the input and pterm limits were adjusted according to this process.

On the PC platform, the global collapsing p-term limit and collapsing input limit can be set in the optimization template of the flow engine. The same limits are set on the workstation platforms by using the “-pterm” and the “-inputs” switch on the command line.

Preventing Logic Collapsing

Multiple levels of logic will frequently be collapsed to achieve the fastest pin-to-pin speeds or the greatest f_{MAX}, as shown in **Figure 2**.

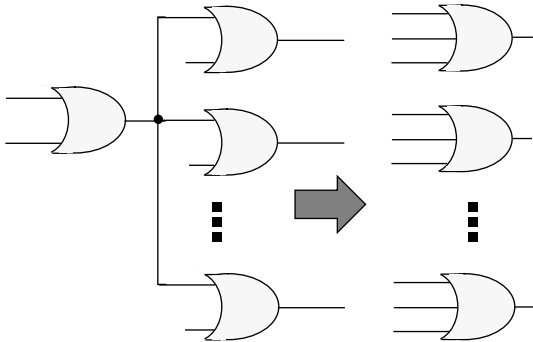


Figure 2: Logic Collapsing

Designers can easily control the collapsing of specific functions by specifying the fanout nodes that are to be exempt

from the collapsing process. This can be accomplished in a schematic by using the **OPT=OFF** attribute. In HDL designs, this is accomplished by specifying the exempt nodes within the text of the HDL design file using properties or attributes. After flattening, the specified nodes are maintained as shown in **Figure 3**, resulting in greater density designs.

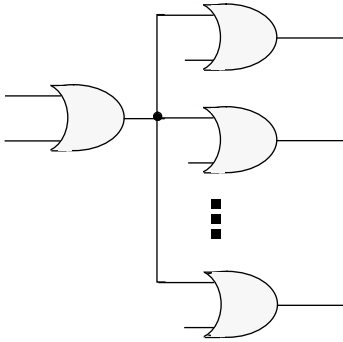


Figure 3: Uncollapsed Node

The increased density is achieved by implementing a high fanout function in a single macrocell versus collapsing the same function into each of the fanouts. The greater the number of fanouts an equation has, the greater the payoff.

Using Global Control Signals

By taking advantage of the global clocks (GCK), global set/reset signals (GSR), and global 3-state signals (GTS), which are available in every macrocell, designers can save the local product terms that would have been used for those same functions. These global signals also provide higher performance than local signals and they conserve the valuable FB inputs which can then be used for other logic signals.

*****Function Block Resource Summary*****						
Function Block	# of Macrocells	FB Inputs Used	Signals Used	Total Pt Used	O/IO Req	IO Avail
FB1	11	36	41	28	0/0	17
FB2	14	36	43	45	0/0	17
FB3	18	36	37	46	12/1	17
FB4	11	36	41	47	8/1	17
FB5	14	36	36	54	8/1	17
FB6	10	35	37	46	8/1	16
FB7	13	36	39	45	7/2	16
FB8	18	36	36	78	9/0	16
-----				-----	-----	-----
109				389	52/6	133

Figure 4: Report Showing a Design That Fits

Optimizing Timing

XACT^{step} provides three convenient means for timing optimization: Automatic timing optimization, TSPECs, and variable p-term collapse limits.

Using Automatic Timing Optimization

The automatic timing optimization minimizes the longest path in a design. It also improves the general timing of the whole design if detailed timing control (TSPECs) is not requested. Furthermore, it usually has no significant effect on design density or software run time. Timing optimization can be controlled in the optimization template on the PC, or by using the -notiming switch on workstation platforms.

The default for timing optimization is OFF.

Using TSPECs

Timing Specifications (TSPECs) define the required timing for specific paths in a design, causing the software to choose the optimum logic mapping to achieve the specified delays. TSPECs allow precise control over which paths are to be optimized. Specifically, this affects the collapsing order and the assignment of critical product terms near a macrocell output. When a signal has a **TSPEC** attribute, the fitter will use various methods to obtain the specified performance. These methods include using logic collapsing, limiting product term cascading, and using fast feedback paths within the FB.

See Appendix 1 for more information on using TSPECs.

Using P-Term Collapse Limits

Increasing the global p-term collapse limits increases the collapsing of combinatorial nodes into their fanouts. This usually increases the speed of the design but requires more device resources to implement.

Optimizing Schematic Designs

The following optimization techniques can be used in schematic designs.

Slew Rate Control

Assign the **FAST** attribute to output buffers or pads to specify the fastest slew rate. Unspecified outputs default to the slow slew rate.

Power Control

Assign the **LOWPWR** attribute to those signals that don't require the highest speed, in order to reduce power consumption. This attribute only affects macrocells, not the I/O blocks. All unspecified outputs default to the higher power, higher speed option.

Local Feedbacks

To use a local feedback path, use the **LOC** attribute to assign both the source and destination components to the same function block. Then, apply a TSPEC to that path such that it can only be met using a local feedback path.

Logic Optimization

Use the **OPT=OFF** attribute on selected nodes to inhibit the collapsing of those logic functions.

Optimizing ABEL Designs

The Xepld property statements are used within ABEL designs to control optimization. These property statements are passed directly to the Xilinx fitter and are not used by ABEL.

Slew Rate Control

The **FAST** property controls the output slew rate, and there can only be one **FAST** property used in each design. If there are only a few signals that require a fast slew rate, they can be listed individually after the property, and the remaining signals will be slew rate limited. Or, if there are only a few signals that need to be slew rate limited, then those signals can be listed.

```
xepld property 'fast on';
"all pins have fast slew rate

xepld property 'fast on x1 x2';
" only x1 and x2 are fast
" the remaining pins are slew limited

xepld property 'fast off x1 x2';
"only x1 and x2 are slew limited
"the remaining pins are fast
```

Local Feedbacks

The **PARTITION** property is used to force the specified signals into a specified FB. For example, the following statement will force signal A and B into function block 1.

```
xepld property 'PARTITION FB1 A B';
```

This xepld property statement, can be combined with a TSPEC in a .CST constraints file to specify timing. For example:

```
TIMESPEC = "TS01=FROM:FFS(A):TO:FFS(B)=3"
```

Power Control

The **PWR** property controls the power settings for individual macrocells.

```
xepld property 'pwr low';
"places all macrocells in low power mode

xepld property 'pwr low x1 x2'
```

```
"places x1 and x2 in low power mode
"the remaining in STD power mode

xepld property 'pwr std x1 x2'
"places x1 and x2 in STD power mode
"the remaining in low power mode
```

Logic Optimization

The **LOGIC_OPT** property allows the user to control the logic optimization done by the fitter. This should be used on selective nodes where collapsing those nodes would cause the design to become very large.

```
xepld property 'logic_opt off';
"Preserves all combinatorial nodes

xepld property 'logic_opt off x1';
"preserve x1 and collapse other nodes to
fitter limits
```

Optimizing VHDL Designs

VHDL support varies among the existing vendors in how the information is passed to the fitter. The summaries that follow describe the methods used with the Synopsys and Metamor VHDL tools.

Synopsis

- **Slew Rate Control** is accomplished with a script command:

```
-set_pad_type-slewrates NONE <port names>
```
- **Local Feedbacks** are accomplished by using the **LOC** attribute with a TSPEC (.CST file). **LOC** attributes are handled as follow (the italic text indicates additions to a standard script):

```
edifout_write_properties_list = loc
edifout_netlist_only = true
edifout_power_and_ground_representation =
cell
analyze-format vhdl<design_file_name>
elaborate <design_name>
compile
set_attribute <output_port> LOC <fb_name>
-type string

set_pad_type -slewrates NONE all_outputs()
set_port_is_pad ""
insert_pads
write -format edif -hierarchy -output
<design_name>.edif
```

- **Power Control** is accomplished with the following script alterations, which pass low power properties into the design file (italic type indicates additions to a standard script):

```
edifout_write_properties_list = LOWPOWER
edifout_netlist_only = true
edifout_power_and_ground_representation =
cell
analyze-format vhdl <design_file_name>
elaborate <design_name>
compile
```

```
set_attribute <register_net>_REG LOWPWR
ON -type string
```

```
set_pad_type -slewrates NONE all_outputs()
set_port_is_pad ""
insert_pads
write -format edif -hierarchy -output
<design_name>.edif
```

Note that the register net will have appended a extra _REG to the name.

Metamor

- Slew rate control is managed with a Metamor library attribute. For a library attribute, include the metamor libraries. For example:

```
LIBRARY metamor;
USE metamor.attributes.all;
```

Then, apply a property to the output signal as follows:

```
attribute property of B : signal is "Fast"
```
- Local feedbacks are handled with the **LOC** attribute and a TSPEC. To apply a **LOC** attribute, the designer must first define it. This is done with the following code.

```
attribute LOC : string;
```

Then, apply the attribute to specific signals, as follows:

```
attribute LOC of B : signal is "FB2";
```

A TSPEC can also be applied using a .CST constraints file.
- Power control is accomplished by defining a **LOWPWR** attribute and applying it to signals as well. This can be done with the following code:

```
attribute LOWPWR : string;
```

Then, apply the attribute to specific signals as follows:

```
attribute LOWPWR of B : signal is "on";
```

Appendix 1 — TSPEC and .CST File Format

TSPECs can be applied to designs using constraint files. Constraint files can be used with all software flows, and there are several ways to use them - depending on the specific path taken to the fitter step. The constraint file is named "design.cst", where "design" is the same character string as the other design files. The .CST file is simply a text file comprised of several lines that dictate specific timing relationships must be met within the design.

There are only two types of signals defined in TSPECs. These are FFS (flip-flops) and PADS (primary I/Os). In general, four parameters need to be controlled:

- **t_{PD}**, the time a combinatorial output becomes valid after an input has changed is defined using a PADS to PADS definition.
- **t_{SU}**, the external setup time for a signal at the pin of the part before applying a clock is defined using a PADS to FFS definition.
- **f_{MAX}**, the cycle time at which a set of flops can be clocked internally is defined using FFS to FFS definition.
- **t_{CO}**, the clock to out time of the design is specified by using a FFS to PADS definition.

Specifying the estimated external clock speed requires the user to specify t_{SU}, f_{MAX}, and t_{CO}. The external clock speed will be calculated by using the longest delay of the three parameters.

The following examples will show the basic syntax of the constraint file and apply these definitions to control timing.

Example 1: Specifying t_{PD}

```
TIMESPEC = "TS01 = FROM:PADS:TO:PADS = 7.5";
```

This TSPEC will constrain all signals that originate and terminate at pins with a maximum time of 7.5 ns. When specifying t_{PD}, however, the designer is usually only interested in certain paths in the design. The designer can specify groups of signals using the "timegrp" command. For instance, assume the design has a set of inputs A15 to A0, and a set of outputs X15 to X0. By using a "timegrp", the designer can limit the constraints to only those signals. Note that the designer can use the "*" for a wildcard match.

```
TIMEGRP = "AddressIn = PADS(A*)";
TIMEGRP = "AddressOut = PADS(X*)";
TIMESPEC = "TS01 = FROM: AddressIn:TO:AddressOut = 7.5";
```

The TIMESPEC command specifies that the delay between the inputs (A*) and the outputs (X*) is less than or equal to 7.5 nanoseconds.

Example 2: Specifying f_{MAX}

f_{MAX} is an extension of the TIMESPEC concept mentioned in Example 1. To specify the maximum required operating frequency for a flip flop or a group of flip flops, just use the following equation:

$$\text{TIMESPEC_value} = (1/f_{\text{MAX}})$$

where f_{MAX} is the required operating frequency. To get 100 Mhz operation for a group of flip flops, 10 ns must be specified. Again, this can be applied over a group of flip flops using the TIMEGRP expression.

As a more concrete example, consider a design where multiple asynchronous chip select signals, Csel3 to Csel0, are being synchronized by a set of input registers. Both the input registers and the Csel registers are clocked by one source. First, the designer can define the destination registers, again using a wildcard match. Then, the designer can apply a TSPEC from all FFS to the timegroup, Csel.

```
TIMEGRP = "Csel=FFS(C*)";
TIMESPEC = "TS01 = FROM:FFS:TO:Csel=10";
```

This constraint limits the cycle time from the output of the input registers to the Csel registers to be a maximum of 10ns. This would allow us to clock the design at 100mhz without violating internal setup times.

Example 3: Specifying t_{SU}

To specify the worst case setup time for device inputs, use the following equation:

$$\text{TIMESPEC_value} = (\text{desired setup time}) + (t_{\text{GCK}})$$

where t_{GCK} is the fast clock buffer time obtained from the target device data sheet.

For example, assume the desired setup time to clock delay is no more than 5.5 ns for the timespec design. The value of t_{GCK} for the XC95108-7 is 2.5 ns. Therefore the TIMESPEC value is (5.5 ns) + (2.5ns) = 8.0 ns.

In this example, assume that the Csel registers are being driven by inputs A15 to A0. the designer can now group the inputs and the registers together, and apply a TSPEC from the inputs to the flip-flops.

```
TIMEGRP = "Address=PADS(A*)";
TIMEGRP = "Csel=FFS(C*)";
TIMESPEC = "TS01=FROM:Address:TO:Csel=8.0";
```

Example 4: Specifying t_{CO}

To specify the worst case clock-to-output time for a registered device output clocked by a global clock (t_{CO}), use the following equation:

$$\text{TIMESPEC_value} = (\text{desired } t_{CO}) - (t_{GCK})$$

In this example, assume that the Csel registers require a t_{CO} of 5.5 ns. Again, the designer can group the registers together, and apply a TSPEC from the flip-flops to the output pads.

```
TIMEGRP = "Csel=FFS(C*)";
```

```
TIMESPEC = "TS01=FROM:Csel:TO:PADS=3";
```

This TSPEC will constrain the signal path delay from the output of a flip-flop to the pin to 3 ns.