ProASIC^PLUS Timing Closure in Libero® IDE v5.2

Introduction

This application note discusses the new ProASIC^PLUS timing-driven place-and-route (TDPR) flow introduced in Libero Integrated Design Environment (IDE) v5.2 and procedures for achieving timing closure. The tighter integration between the timing engine and TDPR offers push-button results and reduces the number of design iterations required to achieve timing closure.

Timing constraints guide synthesis and place-and-route tools toward achieving the required timing performance for a design. In Libero IDE v5.1 and earlier releases, you are required to deliver ProASIC^PLUS timing constraints to TDPR through GCF (ProASIC Constraints File) files.

In Libero IDE v5.2, all timing constraints for the ProASIC^PLUS family are processed and stored by the Design Constraint System (DCS). You can enter timing constraints into DCS through the Timer GUI, by importing SDC auxiliary files, or during the GCF to SDC (GCF2SDC) conversion. The Layout tools then rely on Timer to provide the appropriate timing constraint information. During TDPR, if you do not specify any timing constraints, Timer automatically generates clock constraints for all potential clocks. This allows layout to exploit the benefits provided by the features introduced in Libero IDE v5.2. Physical constraints (Table 1 on page 2) are still delivered to TDPR through GCF constraints.

To facilitate moving ProASIC^PLUS designs and GCF constraints to Libero IDE v5.2 from earlier releases, a new, automatic GCF2SDC conversion program is available in Libero IDE v5.2. If you use Libero IDE v5.2 to open a ProASIC^PLUS design with GCF timing constraints already applied, the tool automatically converts the GCF timing constraints to SDC constraints. The new SDC constraints are saved in the DCS, and an SDC file is automatically exported. The log of the GCF2SDC translation is appended at the end of the exported file. The conversion completes by giving a summary in the log window of how many constraints have been converted and the resulting status. It also indicates the path to the exported SDC file. You must review this file. If any constraint changes are needed before running TDPR, you can manually change in the SDC file, which can be re-imported. You can also make changes to the constraints using the Timer GUI. Figure 1 shows the new design flow in Libero IDE v5.2.

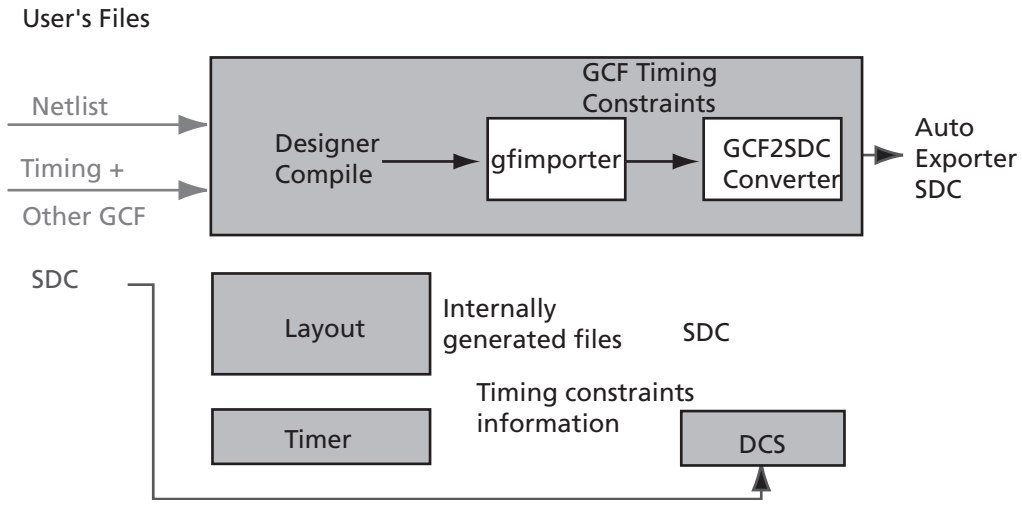

Figure 1 • **Design Flow for ProASIC^PLUS in Libero IDE v5.2**

Setting Constraints for New ProASIC^PLUS Designs in Libero IDE v5.2

Actel recommends using SDC or the Timer GUI for setting timing constraints.

In Libero IDE v5.2, you can explore more options in the timing constraints before adding in physical placement constraints. Closer coupling of the Timer with Layout tools facilitates timing closure. This often eliminates the need for additional physical constraints. Start with clock constraints and timing exception constraints, and then add path specific delays, if needed. If performance requirements are still not met after exploring the timing constraints, add spine constraints, then memory constraints, and, finally, region constraints. See the "Timing Closure Procedures" section on page 3 for step-by-step instructions about how to achieve timing closure.

Physical Constraints

Physical constraints for ProASIC^PLUS devices still use the GCF format. Actel recommends that you create a GCF file specifically for physical constraints and an SDC file for timing constraints. You must import the GCF file along with the netlist as a source file. Table 1 lists physical constraints (physical placement constraints and global resource constraints) that are available. The SDC file is imported as an auxiliary file anytime after the design has been compiled. Timing constraints imported in SDC or set in the Timer GUI are stored in the DCS for use by Timer during TDPR. See the "Merging SDC constraints" section on page 10.

Table 1 • **List of Physical Constraints**

Physical Placement Constraints	Global Resources Constraints
set_io	set_global
set_initial_io	set_noglobal
set_empty_io	use_global
set_io_region	dont_fix_globals
set_location	set_auto_global
set_initial_location	set_auto_global_fanout
set_empty_location	
set_net_region	

SDC Constraints

Libero IDE v5.2 supports the following SDC constraints for ProASIC^PLUS devices:

- `create_clock -period period_value [-waveform edge_list] port_pin_list`
  - The `create_clock` constraint applies clock constraints to a clock pin or port in the design.
- `set_max_delay [-from from_list] [-to to_list] delay_value`
  - The `set_max_delay` constraint sets the path delay from and to the specified pins or ports to a restricted value.
- `set_false_path -through through_list`
  - The `set_false_path` constraint identifies paths in the design that are to be marked as false, so they are not considered during timing analysis. Libero IDE does not support `–from` and `–to` options for the `set_false_path` constraint in Libero IDE v5.2.
- `set_multicycle_path path_multiplier [–from from_list] [-to to_list]`
  - Use `set_multicycle_path` constraint when a path can take two clock periods or more. Libero IDE v5.2 does not support `-through` option for the `set_multicycle_path` constraint.

Constraints to Avoid in the New ProASIC^PLUS TDPR Flow

Actel recommends that you avoid the following constraints in the new ProASIC^PLUS TDPR flow:

1. Avoid selective buffering and buffer prevention on nets during synthesis and/or TDPR.

   For various reasons, such as reducing area utilization and controlling congestion, ProASIC^PLUS users have been preventing buffering and replication of nets both in synthesis and layout. This results in high fanout nets being passed to TDPR, which may negatively impact performance.
2. Avoid using physical placement or routing constraints such as `set_location` and `set_critical`.

   In Libero IDE v5.2, performance goals may be achieved with timing constraints alone. Utilize physical placement constraints (Table 1 on page 2) only when necessary. If not carefully used, the physical placement constraints may adversely impact performance.
3. Do not use `generate_paths` and `set_switch_threshold` constraints. These constraints were used in Libero IDE v5.1 and earlier releases but have no effect on TDPR in Libero IDE v5.2. Use timing constraints to achieve similar results.
4. Do not run layout with the "Route Incrementally" option after modifying a global or clock constraint.

   If the placement changes considerably following the constraint modification, the router will have difficulty and issue a series of repetitive warning messages. Perform a complete re-layout again without using the "Route Incrementally" option after changing global or clock constraints.

Timing Closure Procedures

With the new ProASIC^PLUS TDPR flow introduced in Libero IDE v5.2, the place-and-route responds to the timing constraints better than previous releases, typically resulting in better performance. This section describes the steps in achieving timing closure for two different cases:

- "New Designs" section on page 3
- "Migrating Existing Designs with GCF Constraints to Libero IDE v5.2" section on page 7

New Designs

As mentioned in the previous section, Actel recommends that you explore more options with timing constraints before adding in physical placement constraints. Follow the steps below for new designs or designs with timing constraints created in Libero IDE v5.2 to achieve timing closure. After each layout, determine if performance targets are achieved before proceeding to the next step. Generally, you can achieve timing closure using step 1 to step 4 below.

1. Check timing exception constraints syntax

   In the SDC file, verify that the `set_false_path` and `set_multi_cycle_path` constraints options are in the Libero IDE v5.2 supported format (Table 2 on page 12).
2. Use timing constraints and check performance

   Input the timing constraints via the Timer GUI or SDC timing constraint file. Use the "Multiple Passes Layout" option and check whether performance can be achieved. If not, go to the next step.
3. Assign or adjust global and spine assignments

   When there are global or spine assignments, check to verify if they are consistent with the general design placement of memory blocks, core logic tiles, and pin assignments. If not, make the necessary changes and use the Multiple Passes Layout option again. Check the performance and proceed to the next step if performance is not yet achieved.
4. Consider following memory placement recommendations

   Invoke the MultiView Navigator GUI and Timer to determine if the memory placement is the bottleneck for the design performance (Figure 2 on page 4). To check whether the placement is optimal or not:

– From the Timer GUI, verify if the memory blocks are in the critical path with nets that have high timing delays (> 3 ns).
– From the MVN GUI, verify if the cascaded memory blocks are placed closely together. The software will attempt to place the memory blocks intelligently for optimal performance.
– From the MVN GUI, verify if the memory blocks are placed closely to the fan-in/fan-out of the core logic titles, spine, and I/O pins.

If you observe that any of the above are true, then perform Memory Placement in MVN GUI and perform five multiple passes. Check the performance results. If the performance goals are not met, proceed to the next step.

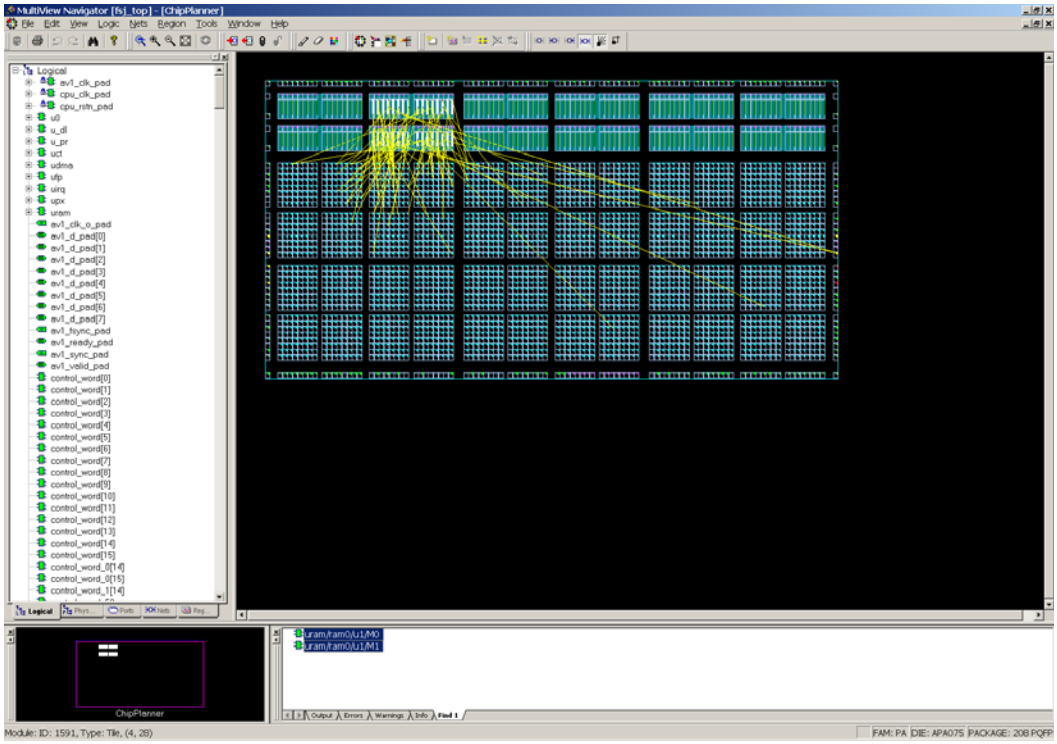

Figure 2 • **Validating Memory Placement in MVN**

5. Apply path-specific timing constraints.

   If there are less than 50 timing violations remaining at this point, add specific path delays using the Timer GUI. A few iterations may be needed to eliminate all violations.

   – In Timer, select the Paths tab and select the specific path sets such as "All Registers to All Registers," "All Inputs to All Registers," or "All Registers to All Outputs" that violate timing requirements. Select the path set and its corresponding path(s) that violate the timing.
   – Set a path delay requirement on the violating paths in the corresponding MaxDelay column. For Timer to accept and honor the new MaxDelay timing constraints, the new timing constraint value must be tighter (smaller) than the existing global constraints. For example, the MaxDelay for Path 1 must be less than 30 ns, and the MaxDelay for Path 5 must be less than 30 ns (Figure 3 on page 5). The new accepted constraints appear as the user-defined paths on the top portion of the Timer -> Paths tab (Figure 4 on page 5).
   – When there are only a few timing violations, perform incremental placement in the layout stage. In this case, timing constraints can be met without affecting the performance of the rest of the design.

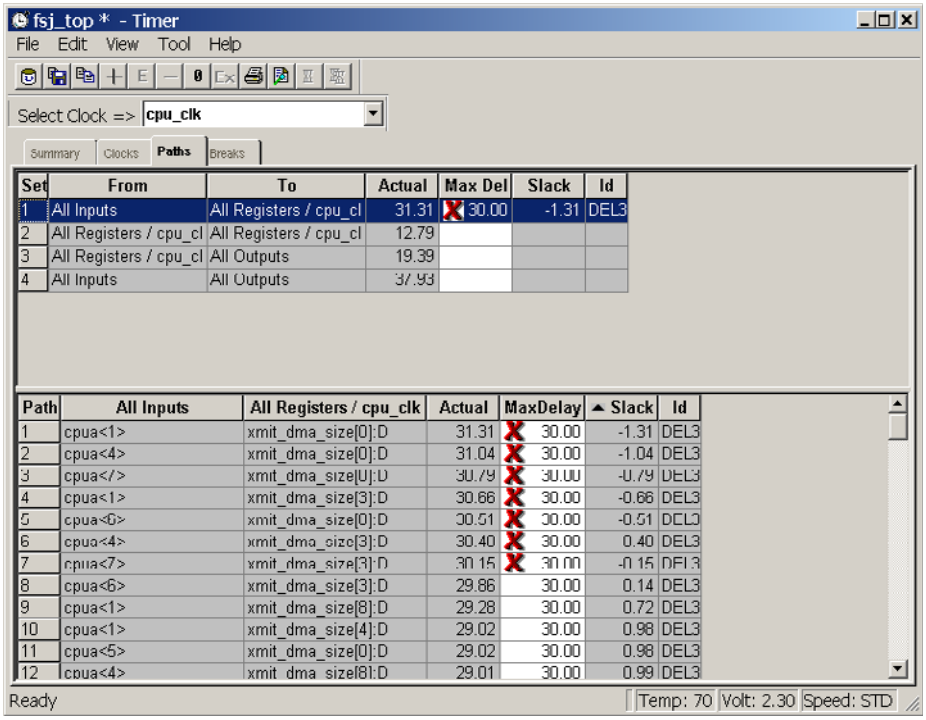

Figure 3 • **Input to Register Paths That Violates Timing**

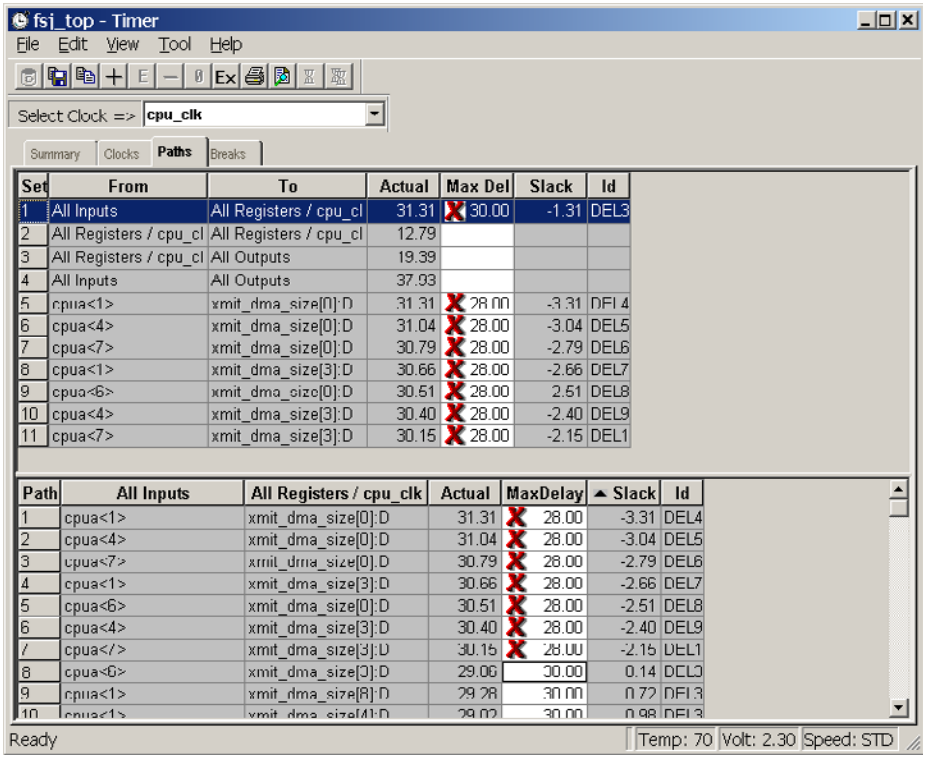

Figure 4 • **User-Specified Input to Register Paths**

There are two types of incremental placement modes:

– Place Incremental ON Mode (Figure 5)
– Place Incremental FIX Mode (Figure 6)

For Incremental ON Mode, the placer runs partially with the new and tighter timing constraints as defined by you. Choose this option when the number of violations is more than 25.

For Incremental FIX Mode, the existing placement is unchanged, and the performance improvement comes from better routing. Choose this option when the number of violations is minimal (approximately 10).

Please note that whenever timing constraints are modified, the router must be run in full mode (the checkbox for "Route incrementally" must NOT be checked in the Layout Options).

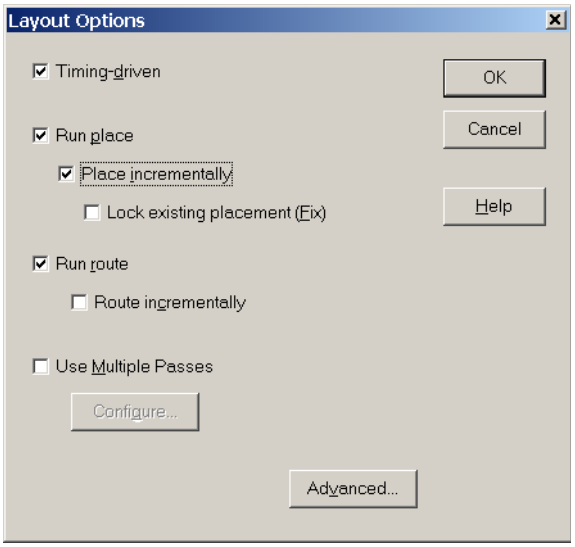

Figure 5 • **Place Incremental ON Mode**

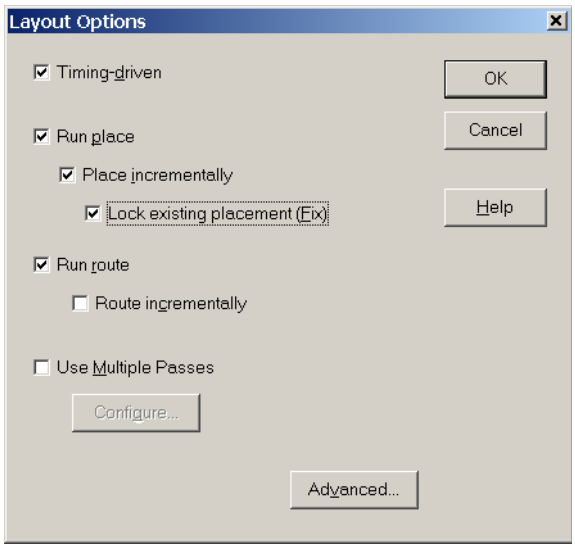

Figure 6 • **Place Incremental Fix Mode**

6. The final step in achieving timing closure is to perform floorplanning.
   – Invoke the MVN GUI and look at the layout of the design
   – Depending on the global utilization, memory placement, spine allocation, and pin placement, either create an Inclusive Region(s) or an Empty Region(s)
   – Make sure that the Region is 15-20% bigger than the required size to avoid congestion

   Run Layout using the "Run Multiple Passes" option. If performance is still not achieved, this may be the maximum that can be obtained for this netlist.

Migrating Existing Designs with GCF Constraints to Libero IDE v5.2

Follow this sequence of steps to ensure optimal performance of your design if you used GCF constraints in a release earlier than Libero IDE v5.2. After each layout, evaluate if performance targets are achieved before proceeding to the next step. Generally, you can achieve timing closure using step 1 to step 4.

1. Check timing exception constraints syntax

   In the existing GCF file, verify that the format used is supported in Libero IDE v5.2. In particular, `set_multicycle_path` and `set_false_path` (Table 2 on page 12). Then proceed with bringing your design and GCF constraints into Libero IDE v5.2
2. Run TDPR and check performance

   Use the Multiple Passes Layout option to obtain maximum performance. At this point, many designs will meet performance goals.
3. Remove physical constraints and re-run Layout

   Existing physical constraints may negatively impact performance. See the "Constraints to Avoid in the New ProASICPLUS TDPR Flow" section on page 3.
4. Evaluate spine assignments

   If there are spine assignments, check to verify if they are consistent with the general design placement. Pre-existing assignments may not be optimal for Libero IDE v5.2 TDPR. If changes are made, run Layout again and evaluate the performance. If no changes are needed, proceed to the next step.
5. Apply path-specific timing constraints

   If there are less than 50 timing violations remaining at this point, add specific path delays using the Timer GUI. A few iterations may be needed to eliminate all violations.

   To apply the path-specific timing constraints:
   – From the Timer under the Paths tab, identify the specific path sets such as "All Registers to All Registers," "All Inputs to All Registers," or "All Registers to All Outputs" that violate timing requirements. Select the path set and its corresponding path(s) that violates the timing.

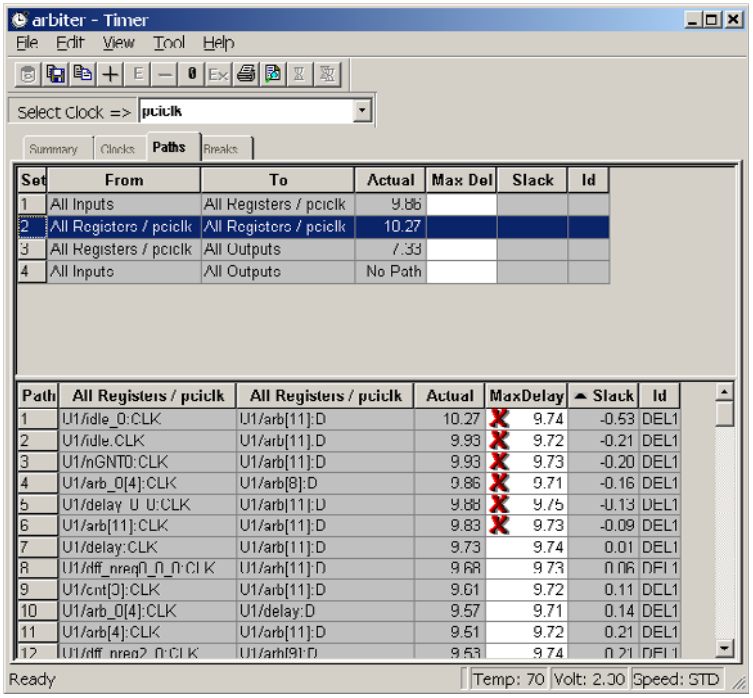

Figure 7 • **Register to Register Paths That Violate Timing**

– Set a path delay requirement on the violating paths in the corresponding MaxDelay column. For the Timer to accept and honor the new MaxDelay timing constraints, the new timing constraint value must be tighter (smaller) than the max delayed derived from the clock period. For example, the MaxDelay for Path 1 must be less than 9.74 ns, and the MaxDelay for Path 5 must be less than 9.75 ns (Figure 7). The new accepted constraints appear as the user-defined paths on the top portion of the Timer -> Paths tab (Figure 8 on page 9).

– When there are only a few timing violations, perform incremental placement in the layout stage. In this case, timing constraints could be met without affecting the performance of the rest of the design.

There are two types of incremental placement modes:

– Place Incremental ON Mode (Figure 5 on page 6)

– Place Incremental FIX Mode (Figure 6 on page 6)

For Incremental ON Mode, there is partial replacement with the new and tighter timing constraints as defined by you. This option should be chosen when the numbers of violations are 25+.

For Incremental FIX Mode, the existing placement remains the same and the performance improvement comes from better routing. This option should be chosen when the numbers of violations are minimal (approximately 10).

Please note that whenever timing constraints are modified, the router must be run in full mode (the checkbox for "Route incrementally" must NOT be checked in the Layout Options).

6. Confine the design to a region

The final step to achieving timing closure is to perform floorplanning

– Invoke the MVN GUI and look at the layout of the design.

– Depending on the global utilization, memory placement, spine allocation, and pin placement, either create an Inclusive Region(s) or an Empty Region(s).

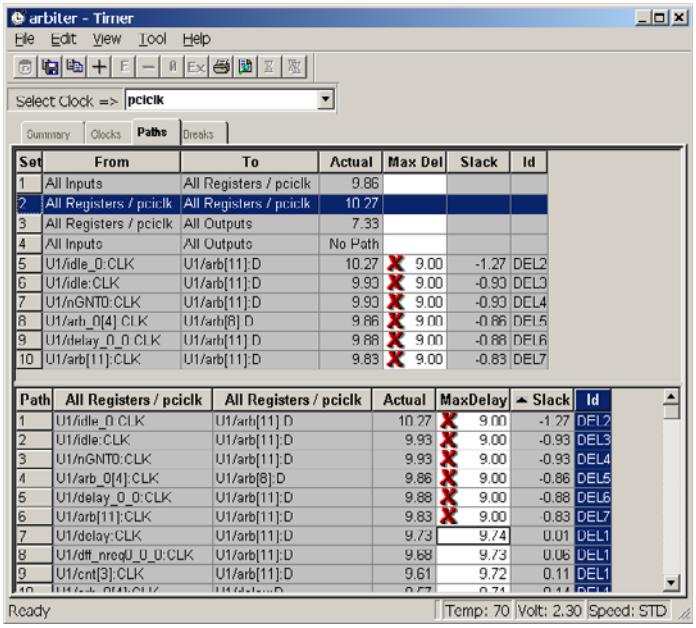

Figure 8 • **User-Defined Register to Register Paths**

– Make sure that the Region is 15-20% bigger than the required size to avoid congestion.

Run Layout using the "Run Multiple Passes" option. If performance is still not achieved, this may be the maximum that can be obtained for this netlist.

Performance Improvements in Libero IDE v5.2

Improved Slack Budgeting

The new TDPR flow provides better budgeting of slacks among different types of timing constraints (Input to Register, Register to Register and Register to Output). In multi-clock designs, where timing constraints are not met, the margin or negative slacks of the constrained clocks will be more uniform when the software attempts to meet the timing constraint.

Overall System Improvement

Generally, the enhanced automatic memory placement algorithm and the new TDPR flow show improved overall system performance without adversely affecting the routability of the designs.

Known Issues

Designs with Single Clock Constraint

Whether designs have a single clock or multiple clocks with a single constrained clock, the design will always achieve the optimal performance for the constrained clock. As a result, changing the value of the single clock constraint does not change the placement or the performance.

High Delay Net in Critical Path

In some situations, nets with high delay may exist in the critical paths. These nets are given a lower priority during the pre-layout delay estimation and do not have the priority to use faster routing resources. In the majority of the cases, you can use the Multiple Passes Layout option (with five passes) to resolve this issue, or you can manually achieve this with the following steps:

Open up the post-layout *.ADB file with Libero IDE.

1. Use MultiView Navigator and Timer to identify this high delay net and its driver
2. Invoke Timer and add a path set for this net
3. For this path set, specify a very tight register to register max delay value (suggested value is path delay minus the net delay).
4. Keep the rest of the constraints and run Place Incremental FIX mode (Figure 6 on page 6).

Merging SDC and GCF Timing Constraints into Libero IDE DCS

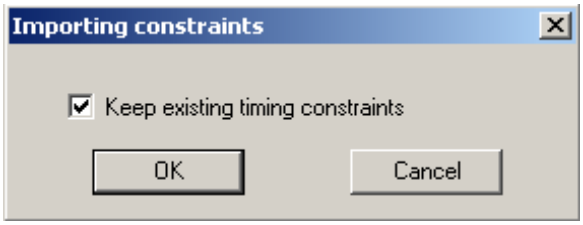

Figure 9 • **Importing Constraints Dialog**

Merging SDC constraints

To merge SDC constraints, use a new check box "Keep existing timing constraints", in the Importing Constraints dialog (Figure 9 on page 10). This option is "On" by default. When checked, timing constraints in imported SDC files are additive to the constraints already existing in the DCS database. If "Keep existing timing constraints" is not checked ("Off"), all the previous DCS timing constraints are removed and replaced by those imported in the SDC files. The behavior of the "Off" option is new in Libero IDE v5.2 and is useful for ensuring a clean set of constraints is provided to TDPR.

Merging GCF constraints

Importing a GCF file in Libero IDE v5.2 always results in appending the translated timing constraints to any existing timing constraints in the DCS. Prior to Libero IDE v5.2, the default behavior with GCF was to overwrite the previous constraints. With the automatic GCF2SDC conversion, there is no option to completely replace existing DCS constraints with those newly imported. You should remove all undesired constraints from the DCS before importing a GCF for conversion. To remove all timing constraints in the DCS: Open Timer and select Edit > Remove All Constraints as shown in Figure 10 on page 11.

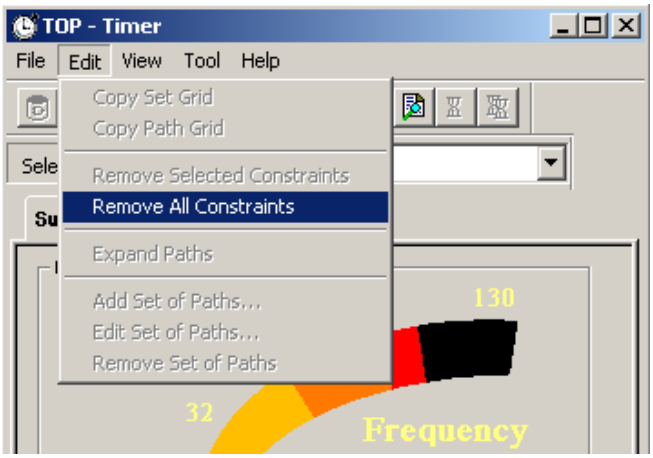

Figure 10 • **Removing Timing Constraints**

Using GCF for ProASIC[PLUS] When Moving a Design to Libero IDE v5.2 from a Previous Release

In Libero IDE v5.1 and earlier, TDPR for ProASIC[PLUS] devices required that you apply timing constraints in the GCF format.

In Libero IDE v5.1, the tool attempts to convert the constraints set in the Timer GUI or SDC to GCF format, but the conversion did not always completely capture the user intent. This limited both TDPR and the static timing analysis capability.

Beginning with Libero IDE v5.2, all GCF timing constraints are automatically converted to SDC. The automatic GCF2SDC conversion happens in two situations:

- The first time a post-compile ADB created in Libero IDE v5.1 or earlier is opened in Libero IDE v5.2
- After compile, to convert any GCF timing constraints that were imported

After the GCF timing constraints are automatically converted and the equivalent SDC constraints are saved into the DCS, an SDC file is exported that contains all constraints present in the DCS. A log appended in the SDC file provides a detailed account of each translated constraint. See the "Merging GCF constraints" section on page 11.

If the software is unable to read or write to the files used during conversion, the following error message is provided in the Libero IDE log window:

```
Error: GCF2SDC Translation System: System error when generating the conversion file.
Unable to access disk.

Error: GCF2SDC conversion failed and a temporary log file will be created.
```

You should review the generated SDC and consider if it needs to be updated. Then import the SDC file into Libero IDE.

Rules for GCF2SDC conversion:

- Verification rule: If a GCF constraint is not recognized by the conversion program, the translation of that GCF constraint is skipped and a failure is reported in the log window (in the final status). This information is included in the log in the exported SDC file.
- Matching rule: If an identical constraint already exists in the DCS, that GCF constraint translation is skipped and the log will report the status as successful.
- Conflict rule: If a GCF constraint conflicts with a constraint already in the DCS, the new GCF constraint gets priority over the existing constraint and the new constraint will be honored. The previous constraint will appear as a comment in the exported SDC file.
- Ambiguity rule: If multiple options are possible during the conversion of a given constraint, one option is selected. The others will appear as comments in the exported SDC file. This situation happens when converting a GCF `create_clock` constraint set on a net.

For an existing design, when the ADB file is opened in Libero IDE v5.2, the tool automatically converts timing constraints from the GCF format to SDC format. The new SDC constraints are then saved into DCS. The GCF2SDC conversion will happen when the GCF file has the following timing constraints:

- `create_clock`
- `set_multicycle_path`
- `set_false_path`
- `set_input_to_register_delay`
- `set_register_to_output_delay`
- `set_max_path_delay`

Table 2 shows the mapping between GCF timing constraints and SDC constraints. Please note that the Libero IDE v5.2 DCS does not support all options for `set_false_path` and `set_multicycle_path`. For this reason, some GCF constraints will not have an equivalent effect when translated into SDC. However, although they will not be used, these constraints with unsupported options are translated and saved, so as not to lose the original user intent.

If the GCF constraint `create_clock` is applied to a net, it needs to be converted to the appropriate potential clock port pin or pin driving that net. In some cases the automatic GCF to SDC conversion may not be able to locate the appropriate driver. You should check your design and manually set a `create_clock` for this clock.

Table 2 • **Mapping GCF to SDC**

GCF Command	SDC Command
`create_clock -period <period_value> portname`	`create_clock -period period_value portname`
`create_clock -period <period_value> netname`	`create_clock -period period_value potential_clock_port_pin_name`
`set_false_path [-from from_port] [-through any_port][-to to_port]`	`set_false_path -through through_list`
`set_input_to_register_delay <delay> [-from inp_port]`	`set_max_delay delay_value [-from from_list][-to to_list]`
`set_multicycle_path <num_cycles> -from reg_port [-through_any_port] [-to_port]`	`set_multicycle_path path_multiplier [-from from_list][-to to_list]`
`set_resgister_to_output_delay <delay> -to out_port`	`set_max_delay delay_value [-from from_list][-to to_list]`
`set_max_delay -from first_pin_in_the_list -to last_pin_in_the_list`	`set_max_delay delay_value [-from from_list][-to to_list]`

Important GCF to SDC Conversion Considerations

You may encounter the following situations during the GCF2SDC conversions:

1. Using multi-cycle constraints with a `-from` only or `-to` only.

   Using `-from` only for multi-cycle constraints results in an excessive effort by Timer. This increases run time and memory usage. Please use both `-from` and `-to` for multi-cycle constraints.

2. The `-from` and `-to` options for `set_false_path` and `-through` option for `set_multi_cycle_path` are not recognized in Libero IDE v5.2.

   Although these options can be imported, saved, and exported in Libero IDE v5.2, Timer does not utilize these constraints for timing analysis and TDPR. These options are not seen in the Timer GUI. Use only the supported constraint options listed in Table 2 on page 12.

3. For multi-cycle path constraints, Timer does only setup checking, no hold checking.

   The hold check does not take the multi-cycle constraint into account, so you may see false hold violations. As an alternative, use clock exceptions and manually check the minimum delay.
   See Timer online help for information about clock exceptions.

4. GCF constraints `set_register_to_output_delay` and `set_input_to_register_delay` do not have clock domain options.

   Libero IDE does not support `set_input_delay` or `set_output_delay` SDC constraints. When translating `set_input_to_register_delay` or `set_register_to_output_delay` GCF constraints into SDC constraints, `set_max_delay` is used with all clock domains. However, this may result in the design being over constrained in some cases. You must add the desired clock domain in the translated SDC or in the Timer GUI.

5. GCF2SDC conversion will report successful completion even when there are errors or warnings reported for some constraints.

   Following GCF2SDC conversion, Libero IDE allows you to proceed to layout even if there are failures reported during conversion. Therefore, it is important to review the log section in the SDC file exported following conversion to ensure that all design constraints are set as expected before proceeding with TDPR. Modifications can be made to the constraints in the SDC file or Timer GUI. If the SDC file is modified, re-import the modified file. See the "Merging SDC constraints" section on page 10.

6. If you use a wild card for `set_input_to_register_delay` or `set_register_to_output_delay` constraints, GCF2SDC conversion will over-constrain the design. Consider the following example:

   ```
   set_register_to_output_delay 12.000000 -to "*";
   ```

   which is translated to the following SDC constraints:

   ```
   set_max_delay 12.00 -from [all_registers] -to [get_ports {*}]
   ```

   This applies to the constraint from all registers to all pins. You should modify the SDC to have the constraint to output ports only.

Conclusion

In Libero IDE v5.2, Timer directly provides layout with the required information based on the user-specified constraints. In cases where no user-specified constraints (clock or max-delays), Timer auto-generates clock constraints for all potential clocks. With this flow, layout benefits from all features implemented in the Libero IDE v5.2 Timer. This makes the software and timing for ProASIC^PLUS family devices more predictable and user friendly and generally improves performance.

Related Documents

Application Notes

Static Timing Analysis Using Libero IDE's Timer
http://www.actel.com/documents/Static_Timing_Analysis_AN.pdf

Optimal Usage of Global Network Spines in ProASIC^PLUS Devices
http://www.actel.com/documents/APA_Spines_AN.pdf

Floorplanning ProASIC/ProASICPLUS Devices for Increased Performance
http://www.actel.com/documents/Flash_Floorplanning_AN.pdf

User's Guides

Libero IDE User's Guide
http://www.actel.com/documents/libero_UG.pdf

Timer User's Guide
http://www.actel.com/documents/timer_UG.pdf

ChipEditor User's Guide
http://www.actel.com/documents/chipeditor_UG.pdf

List of Changes

Previous version	Changes in current version 51900045-1	Page
51900045-0*	Renamed application notes title from ProASIC^PLUS Timing-Driven Flow in Libero™ to ProASIC^PLUS Timing Closure in Libero™	
	Added Timing Closure Procedures	page 3
	Added Performance Improvement in Libero IDE v5.2	page 9
	Added Known Issues	page 10
	Reorganized previous application notes	
	Revised Mapping GCF to SDC	page 10

Note: **This is the part number located on the last page of the document.*

http://www.actel.com

Actel Corporation
2061 Stierlin Court
Mountain View, CA
94043-4655 USA
Phone 650.318.4200
Fax 650.318.4600

Actel Europe Ltd.
Dunlop House, Riverside Way
Camberley, Surrey GU15 3YL
United Kingdom
Phone +44 (0) 1276 401 450
Fax +44 (0) 1276 401 490

Actel Japan
EXOS Ebisu Bldg. 4F
1-24-14 Ebisu Shibuya-ku
Tokyo 150 Japan
Phone +81.03.3445.7671
Fax +81.03.3445.7668

Actel Hong Kong
39th Floor, One Pacific Place
88 Queensway, Admiralty
Hong Kong
Phone +852.227.35712
Fax +852.227.35999

51900045-1/05.04