WP480 (v1.0) August 11, 2016

A Methodology for Repeatable and Reliable Timing Closure

By: John Bieker

Achieving repeatable and reliable timing is the designer's ultimate goal. Xilinx provides the tools and techniques to simplify timing closure.

ABSTRACT

The world is a big place filled with digital designs from many different types of companies. Each company has their own set of requirements and deadlines that drive their methods and techniques for developing customized designs.

One thing that designers all have in common is the need to ensure that their designs work reliably given process, voltage, and temperature variations over time. The task of writing timing constraints and validating the design against those constraints is commonly referred to as timing closure. Timing closure is essential for every design.

But how long does it take to close timing on a design? How do companies measure the progress of a design toward timing closure? How does one determine whether a given design is stable or unstable from a timing perspective? These questions, and others like them, are the subject of great debate across the industry. As the world's leading provider of all programmable FPGAs, SoCs, MPSoCs, and 3D ICs, Xilinx offers a unique perspective.

Understanding the Impact of Resource Utilization

Xilinx offers a wide variety of devices, with each device having a specific mix of available resources. To the extent that a design uses these resources, it either becomes easier or more difficult to close the timing on the design. As resource utilization increases, the solution space for closing timing on a design decreases. Finding a good placement/routing solution for a design that uses a fraction of the available resources is easier than finding a good solution for a design that uses a larger percentage of the device.

Any serious review of timing closure methodology must begin with an assessment of the resources consumed by a design versus the available resources in the selected device. To that end, Xilinx's Vivado® HLS and Vivado Synthesis provide designers with a wide array of tool options, directives, switches, and code attributes that allow the designers to make informed decisions about how to target their designs to the devices they have chosen. Seemingly low-level decisions such as whether to unroll a loop in C++, or what type of memory should be inferred in VHDL or Verilog, can be the difference between whether or not a design is timing-closed. Mastering control over high- and low-level synthesis tools to deliver consistent timing results is a key differentiating factor between designs that exhibit consistent timing closure and those that "don't."

Xilinx offers an extensive selection of training and user guides explicitly designed to help designers increase their level of efficiency and reduce their time-to-market by delivering consistent results. Ensuring that a design is optimally synthesized and properly constrained are necessary to achieve the best possible performance with Xilinx devices. It is important for designers to review utilization for lookup table (LUT), flip-flip (FF), RAMB, and DSP components independently. For example, a design with low LUT/FF utilization might still experience placement difficulties if RAMB utilization is high, and vice versa. A broad discussion of how to create a clean set of timing constraints and manage synthesis options is provided in the design creation and implementation sections of UG949, *UltraFast™ Design Methodology Guide for the Vivado Design Suite*.

Measuring Quality of Results

In the *UltraFast Design Methodology Guide*, Xilinx offers important guidance on evaluating the progress of the timing closure task after each step of the implementation build (optimization, placement, physical optimization, routing). By checking timing after each implementation step, designers can quickly isolate the point in the flow when the timing starts to diverge from closure. For timing closure to be achieved for a design, three conditions must be met:

- Worst negative setup slack (WNS): Must be greater than or equal to zero
- Worst hold slack (WHS): Must be greater than or equal to zero
- Worst pulse width switching slack (WPWS): Must be greater than or equal to zero

Pulse width switching violations (WPWS) are physical limitations of the device such as the maximum clock rate that can be supported. These are typically caused by user definition mistakes. Hold time issues (WHS) are important because designs that have hold violations do not work regardless of how slow they operate. Hold time violations are generally resolved by adding routing delay. Since closing the setup timing (WNS) for a design is generally the most difficult part of the timing closure exercise, the remainder of this white paper is restricted to dealing with closing the setup timing.

It is important to understand the meaning behind the numbers to determine how close or far away the design is from being closed. The total negative setup slack (TNS) represents the sum of all WNS violations for each timing endpoint. It is also important to understand that Vivado design tools optimize designs to improve the WNS, not the TNS. The TNS is reduced as worst paths are resolved. Evaluating the order of magnitude of the TNS, however, does give the designer a clear indication of how close a design is to timing closure. Table 1 provides generic guidelines for WNS and TNS values to help provide a qualitative understanding of the timing on a given design.

Table 1: **Timing Quality of Results Guidance**

WNS (ps)	
+100 to –100	Excellent
–100 to –300	Good
–300 to –600	Fair
–600 and >	Poor
TNS (ns)	
0 to –10	Excellent
–10 to –100	Good
–100 to –1,000	Fair
–1,000 and >	Poor

The first glimpse of the setup timing results can be obtained after the RTL synthesis step is complete. Evaluating timing at this juncture is critical to ensuring the highest quality netlist prior to placing the design. Ideally, the post synthesis design should have the setup timing-closed. Designers should iterate on the synthesis task while modifying coding style, timing constraints, or tool options to close the setup timing prior to placing the design. Evaluating and improving timing results after synthesis is much more efficient and a better use of the designer's time than waiting until the design is fully implemented. The higher the quality of the initial netlist, the more likely the tools are to deliver better performance on a consistent basis.

Upon generating a timing-closed synthesis result, the design can be optimized and placed. Table 2 provides an example of how a designer can tabulate the timing closure results of a design at each stage of the implementation process. The timing results after the placement phase are particularly important because the route estimates are much more realistic when compared with the pre-placement estimates. The improved route estimates are the result of the placement of each design element being established. A careful review of the post placement timing gives an excellent indication of whether or not setup timing closure can be achieved for the given placement. Designers should iterate on the placement task while identifying potential RTL improvements, timing constraints, placement constraints, and tool options to close the setup timing. Evaluating post placement timing involves reviewing both the WNS and the TNS. Looking at both numbers together is necessary to gain an understanding of the nature of the timing of the design. The closer the WNS and TNS values are to the good and excellent range, the easier and more stable the timing closure task becomes.

Table 2: **Timing Measurements of an Individual Build**

Date	Version	Description	Placer Directive	Place		Physical Optimization		Route	
				WNS	TNS	WNS	TNS	WNS	TNS
4-Apr-16	ver1	Initial Build	Explore	–2.018	–7884	–1.87	–1950	–1.544	–8316

Leveraging Multiple Placer Directives

The stated goal of the design engineer is to produce a timing-closed build. However, the ultimate goal is to develop a procedure to close timing in a repeatable and reliable fashion. Closing timing once is good, but having a set procedure to close timing over many design changes and bug fixes is better. A mechanism is needed to gauge not simply whether a single build is timing-closed, but the ease with which the tools were able to close the timing on a design.

To accomplish this task, additional runs are needed to create a set of data that can be analyzed and measured. Fortunately, the Vivado placer comes with a set of tool directives that control the way that the placer functions. These directives enable and disable tool algorithms that impact the packing and placement of logic in the device. Using a subset of directives (e.g., four or five—enough to make the data statistically relevant), the ease or difficulty of timing closure can be measured, and a broader set of conclusions can be drawn about the design.

In Table 3, several common directives were used. The typical conclusion that designers might draw is that the placer directive **SSI_HighUtilSLRs** is the "best directive" for their design. But there is a different, broader conclusion that must be drawn. The timing results were poor for each of the placer directives that were used. As such, rather than focusing on the one "good" directive and moving forward in the timing closure task using traditional techniques, the observation should be made that the design poses a difficult challenge for the tools across all of the directives used. A broader goal that helps promote stability is to create design changes, constraint changes, methodology changes, and/or floorplans that will drive all of the builds toward timing closure. A steady, measured approach of making one change at a time and then measuring the results is necessary to quickly help designers draw meaningful conclusions about each change. When many variables are changed simultaneously, it is difficult to isolate which change helped or hurt the timing closure cause. Measuring the effectiveness of design changes is paramount in helping push the timing closure task quickly toward completion. For more details on timing closure techniques, including the use of physical optimization, please refer to the Design Closure section of UG949, *UltraFast™ Design Methodology Guide for the Vivado Design Suite*.

Designers can decide to use a different set of placement directives, depending on their designs. The full list of available directives is available in UG904, *Vivado Design Suite User Guide: Implementation*. The important principle is to choose directives that are not all the same. For example, choosing **ExtraNetDelay_high**, **ExtraNetDelay_medium**, and **ExtraNetDelay_low** is not the best choice for this analysis because they are very similar and only differ in the degree to which extra weight is applied to net delays. A better choice is to use the directives **Explore**, **ExtraNetDelay_high**, **ExtraPostPlacementOpt**, **WLDrivenBlockPlacement**, and either **SpreadLogic_medium** (for 7 series) or **AltSpreadLogic_medium** (for UltraScale™/UltraScale+™ devices).

Table 3: **Timing Measurement Across Placer Directives**

Date	Version	Description	Placer Directive	Place		Physical Optimization		Route	
				WNS	TNS	WNS	TNS	WNS	TNS
4-Apr-16	ver1	Initial Build	Explore	–2.018	–7884	–1.87	–1950	–1.544	–8316
			SpreadLogic _medium	–1.744	–1477	–1.744	–1218	–1.461	–6195
			SSI _HighUtilSLRs	–1.859	–6079	–0.943	–1067	–0.62	–3023
			WLDrivenBlock Placement	–2.018	–7884	–1.87	–1950	–1.544	–8316
			Average	–1.91	–5831.00	–1.61	–1546.25	–1.29	–6462.50
			Standard Deviation	0.13	3024.81	0.45	470.27	0.45	2501.51

Evaluating Progress

Upon completion of the placement phase of a set of runs, designers can gather a set of statistics from the runs to measure the impact of a given change. In the example below, a simple spreadsheet was created to tabulate and track progress over time. The average value of the placed WNS shows whether or not the worst setup slack values are trending toward zero. The average value of the placed TNS shows whether or not the overall timing across the entire design is progressing toward closure. The standard deviation of the placed TNS indicates whether the build results are converging or diverging across the set of placer directives.

Figure 1 and Table 4 below illustrate the process. Note the progression of the design from a state of unstable and poor timing results toward repeatable timing closure. Each of the changes produced a significant change in the ability of the placer to deliver better performance. The conclusion is that the timing of the build is becoming more stable with each change. In this case, two placer directives produced designs that were timing-closed after routing.

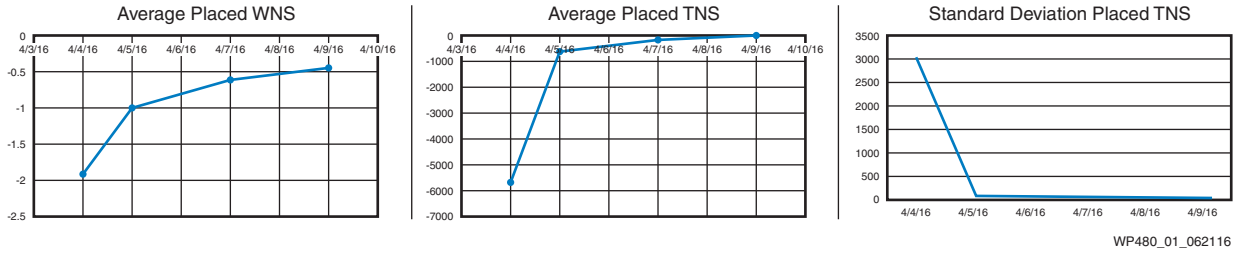

Figure 1: **Timing Measurement over Time (Graphical)**

Table 4: **Timing Measurement over Time (Parametric)**

Date	Version	Description	Placer Directive	Place		Physical Optimization		Route	
				WNS	TNS	WNS	TNS	WNS	TNS
4-Apr-16	ver1	Initial Build	Explore	–2.018	–7884	–1.87	–1950	–1.544	–8316
			SpreadLogic_medium	–1.744	–1477	–1.744	–1218	–1.461	–6195
			SSI_HighUtilSLRs	–1.859	–6079	–0.943	–1067	–0.62	–3023
			WLDrivenBlock Placement	–2.018	–7884	–1.87	–1950	–1.544	–8316
			Average	–1.91	–5831.00	–1.61	–1546.25	–1.29	–6462.50
			Standard Deviation	0.13	3024.81	0.45	470.27	0.45	2501.51
4-Apr-16	ver1	Initial Build	Explore	–1.174	–548	–0.405	–163	–0.525	–723
			SpreadLogic_medium	–0.968	–695	–0.612	–405	–0.367	–463
			SSI_HighUtilSLRs	–0.846	–809	–0.647	–415	–0.608	–2650
			WLDrivenBlock Placement	–1.174	–548	–0.405	–163	–0.525	–723
			Average	–1.04	–650.00	–0.52	–286.50	–0.51	–1139.75
			Standard Deviation	0.16	126.64	0.13	142.66	0.10	1014.27
4-Apr-16	ver1	Initial Build	Explore	–0.392	–195	–0.366	–50	–0.338	–57
			SpreadLogic_medium	–0.608	–105	–0.38	–16	–0.533	–888
			SSI_HighUtilSLRs	–0.61	–77	–0.501	–44	–0.519	–951
			WLDrivenBlock Placement	–0.794	–304	–0.511	–96	–0.733	–2631
			Average	–0.60	–170.25	–0.44	–51.50	–0.53	–1131.75
			Standard Deviation	0.16	102.40	0.08	33.16	0.16	1079.34
4-Apr-16	ver1	Initial Build	Explore	–0.32	–56	–0.178	–2	–0.473	–251
			SpreadLogic_medium	–0.433	–61	–0.199	–6	–0.517	–350
			SSI_HighUtilSLRs	–0.288	–21	–0.161	–1	0	0
			WLDrivenBlock Placement	–0.713	–100	–0.341	–17	0	0
			Average	–0.44	–59.50	–0.22	–6.50	–0.25	–150.25
			Standard Deviation	0.19	32.34	0.08	7.33	0.29	178.14

Identifying Closure Bottlenecks across Placer Directives

The Vivado toolset offers a variety of reporting and diagnostic tools to help designers identify timing bottlenecks. One of the most helpful reports called **report_design_analysis** helps designers quickly answer commonly asked questions such as:

- Does my design exhibit any routing congestion?
- Is my timing closure problem the result of long logic paths?
- Is my timing closure problem restricted to a few clock domains or many?
- Which clock domains are failing most frequently?
- Which clock domains are failing most severely?

Routing congestion can lead to degradation of timing closure during the routing phase. Using report_design_analysis in congestion mode provides designers with visibility into congested regions of a design. Using a congestion optimized synthesis strategy such as "Flow_AlternateRoutability" or congestion optimized placement directives such as AltSpreadLogic_high, AltSpreadLogic_medium, or AltSpreadLogic_low can reduce the impact of routing congestion. For more information on identifying congestion and maximizing routability, refer to the design closure section of UG949, *UltraFast™ Design Methodology Guide for the Vivado Design Suite*.

Using **report_design_analysis** in timing mode allows designers to create a histogram of all of the failing setup endpoints across the design. Tracking the results of the histogram over time gives designers tangible, trackable evidence that their design changes are producing the desired effect.

Consider the two histograms defined in Figure 2. Note that the number of failing timing paths that traverse 8-15 levels of logic has been drastically reduced from one build to the next. This indicates progress toward a more stable, reliable timing closure task. Further note the reduction in overall failing endpoints. The command used to produce these histograms provides a precise list of only failing endpoints up to a maximum of 10,000:

```
report_design_analysis -logic_level_distribution -of_timing_paths
[get_timing_paths -max_paths 10000 -slack_lesser_than 0] -file
violations.rpt
```

2. Logic Level Distribution

End Point Clock	Requirement	0	1	2	3	4	5	6	7	8	9	10	11-15	16-20	21-25	26-30	31+
clk_a	4	55	120	237	279	22	20	77	64	21	20	8	0	0	0	0	0
clk_b	5	121	1272	18	23	12	36	75	174	656	71	35	26	o	o	o	o
clk_c	6.66667	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
clk_d	6.66667	0	0	0	0	0	0	0	0	0	75	47	23	0	0	0	0
clk_e	6.66667	0	0	0	0	0	0	12	163	3	0	0	0	0	0	0	0
clk_f	4	0	0	0	0	0	41	96	67	24	5	0	0	0	0	0	0
clk_rx0	3.103	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
clk_rx1	3.103	0	0	0	0	63	0	0	0	0	0	0	0	0	0	0	0
clk_rx2	3.103	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
clk_rx3	3.103	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
clk_rx4	3.103	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
clk_rx5	3.103	0	16	0	0	72	0	0	0	0	0	2	0	0	0	0	0
clk_rx6	3.103	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

* columns represents the logic levels per end point clock
** distribution is for top worst 4172 paths

2. Logic Level Distribution

End Point Clock	Requirement	0	1	2	3	4	5	6	7	8	9	10	11-15	16-20	21-25	26-30	31+
clk_a	4	313	208	31	121	30	10	63	34	36	4	4	2	0	0	0	0
clk_b	5	157	0	44	1	20	66	55	206	264	26	31	11	0	0	0	0
clk_c	6.66667	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
clk_d	6.66667	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
clk_e	6.66667	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
clk_f	4	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0
clk_rx2	3.103	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
clk_rx4	3.103	0	4	0	0	27	0	0	0	0	0	0	0	0	0	0	0

* columns represents the logic levels per end point clock
** distribution is for top worst 1805 paths

WP480_02_062116

Figure 2: **Failing Path Histogram (By Logic Levels)**

Conclusion

The final goal is not necessarily to close timing for every single placer directive, but rather to improve the overall ease and consistency of closure for any given design. Using this approach helps designers get the most value out of the Vivado software and, ultimately, out of their Xilinx device. The result is that the final design is significantly easier to place and route, which translates to a more reliable and reproducible timing closure task.

This timing evaluation flow is illustrated in Figure 3.

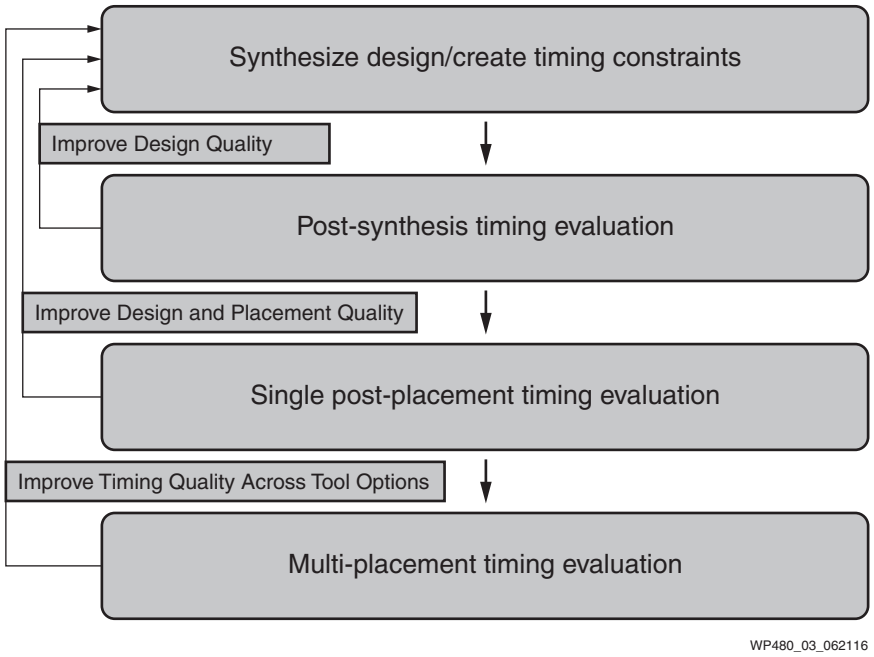

Figure 3: **Timing Evaluation Flow**

For more information on design creation and implementation techniques, refer to UG949, *UltraFast™ Design Methodology Guide for the Vivado Design Suite*.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
08/11/2016	1.0	Initial Xilinx release.

Disclaimer

Automotive Applications Disclaimer