# Predicting Physical Design Results Using Advance Synthesis

Symmid Semiconductor Technology
Shahzad Chowdry

Session # 3.13


cadence designer network

# Introduction

This paper presents the results from Symmid Semiconductor Technology (SST)'s use of the Cadence *RC-Physical* synthesis methodology. The two main components of RC Physical are Physical Layout Estimation (PLE) and QoS Prediction – features in Encounter RTL Compiler 7.1 that provide a new and effective solution to overcoming the problems associated with timing closure. The two traditional approaches to achieving timing closure – using a wire load model based methodology or using a zero wire load methodology are each discussed, along with their associated short-comings. The RC-Physical methodology is then presented, followed by a description of the test plan SST used in testing these methodologies. This paper concludes by presenting the results from the SST timing correlation tests. These tests compare how well RTL Compiler was able to predict back end timing results when designs were synthesized using a standard wire load model methodology, a zero wire load model methodology and the RC-Physical methodology.

## Timing Closure Problems

Few things cause more slips to ASIC tapeout schedules than the unexpected iterations that occur between front end and back end teams as they struggle to close timing. Unfortunately, this scenario plays itself out with increasing frequency with every shrink in process geometries.

Timing closure is achieved when both the front end and back end teams agree that the timing goals for the design have been met. What makes this so difficult is that the two teams often differ in their assessment of which paths are responsible for timing violations and which paths are slack-positive. Problems arise when the front-end team hands off a netlist it believes meets timing but the back-end team's analysis shows it does not. Who is right in such situations? Invariably, it's the back end team, for their timing analysis is based on the actual physically placed cells and routed nets. Front end designers who make decisions based on the timing analysis of a traditional synthesis tool are at a major disadvantage to their back end colleagues. Traditional synthesis tools have no mechanism for taking physical design effects into consideration when optimizing and analyzing a design. This lack of physical awareness has meant the timing numbers generated by traditional synthesis tools, at best, are estimates.

Using estimated values is not necessarily a problem. In fact, during some phases of logic synthesis, timing-driven optimizations take place before a gate-level representation of the design is available. Since you cannot know the actual length of a wire before the cells it interconnects are even available for placement, the only option a synthesis tool has is to estimate interconnect delay. Timing closure problems do not arise from the use of delay estimate models, they arise from the use of delay estimate models that are not predictable, or inaccurately represent the actual wire delay.

Prediction is the ability to forecast future results. And when it comes to ASIC timing closure, predictability is absolutely essential. Good predictability means the timing numbers the front end designer sees coming out of synthesis correlate with the timing numbers the back end designer sees after place and route. Good timing correlation leads to good predictability. Good predictability leads to timing closure that is accomplished with a minimum number of iterations.

## The Problem With Wire Load Models

Most traditional synthesis methodologies involve the use of the wire load model, which attempts to solve one of the most vexing problems traditional synthesis tools face: how to predict interconnect parasitic values when the

physical dimensions of such wires are still unknown. The approach taken by developers of the wire load model was to develop a lookup table that ties the resistance and capacitance of a net to its fanout. The actual parasitic values used in these tables were derived from the statistical analysis of ASIC foundry data for a given process node.

While accuracy of a wire load model has always been crude at best, for process nodes hovering around a micron, the inaccuracy in wire delay estimates had little effect on timing analysis, for it was cell delay that dominated the delay equation at these process geometries.

As designers moved to deep sub-micron process nodes, there was a dramatic switch in the relative contributions that cell delay and interconnect delay made to total timing path delay. Timing path delay was now far more dependent on interconnect timing than on cell timing, which meant the accuracy of the timing numbers calculated by traditional synthesis tools is no better than the crude wire load models upon which they are based.

Wire load models don't always get it wrong. Some percentage of nets in a design are typically short enough that wire load models provides tolerable parasitic estimates. But the percentage of nets that wire load models get right is rapidly dwindling with each feature size shrink or die size increase. Furthermore, there is an estimated 10% of nets that are almost impossible for any delay estimation model to ever accurately predict. These types of nets that fall into this category include:

- Nets connecting to macros
- Nets that route around obstacles
- 'Long' nets that span the die to interconnect clusters of logic or that connect to block level pins that are far removed from each other physically

These types of nets do not lend themselves well to the type of statistical analysis upon which wire load models are based. The parasitic values for these nets are highly correlated to a design's layout, not to these nets' fanout. A minor adjustment to a macro's placement, orientation, size or aspect ratio can cause a net's length to change significantly. Thus, any wire length estimate that is not based on the actual physical placement of cells and nets will invariably be inaccurate.

The problems these hard to estimate nets cause for any pre-layout interconnect modeling technique cannot be underestimated. All the major synthesis tools on the market today are timing-driven, which means a timing path's slack number is the most important factor driving the power, area and timing cost function. Upsizing a cell or swapping a cell for its lower threshold voltage equivalent will improve a path's timing, but at the cost of area and power. Such a tradeoff is acceptable for a path that is timing critical but counterproductive for paths with positive slack. For these latter nets, not only will the timing improvement be of no benefit to these non-critical paths, but their area and power consumption will also increase unnecessarily. This is precisely what happens with traditional synthesis tools. The inability of the synthesis tools to differentiate the paths that are timing critical from those that are not results in some legitimately critical paths being mislabeled as non-critical, and vice versa. Thus, after much hard work, the logic designer produces a netlist that he believes meets timing and is fully optimized for area and power, which in fact contains critical paths that never were optimized for timing and non-critical paths that could have benefited from more aggressive area or power recovery but did not because they were misidentified as timing-critical.

As the netlist makes its way through the physical design process and wire load model estimates are replaced by interconnect parasitic values based on actual wire length, the incorrect assumptions the synthesis tool made are revealed. Back end timing analysis does not match the numbers from synthesis timing analysis.

The front end and back end design teams must then spend time figuring out how to close timing. This is often a labor-intensive process with a completion time that is difficult to estimate. Worse still, effort required to close timing on one chip can rarely be leveraged to help on future designs.

## Zero Wire Load Model Methodology

The zero-wire-load-model (ZWL) methodology was developed to address the lack of predictability of the wire load model. This methodology acknowledges the inaccuracy of wire load models by doing away with them completely. The ZWL methodology uses no wire load model during synthesis, it accounts for interconnect delay instead by increasing clock uncertainty (typically to 30% of the clock period).

While the ZWL methodology attempts to provide a more accurate alternative to the use of wire load models, it doesn't yield noticeably better results. Increasing clock uncertainty results in tighter timing requirements for all paths. This might help with critical paths that the wire load model misidentified as non-critical, but it also penalizes all nets that meet timing as well. Depending on the level of overconstraint, nets that otherwise meet timing may become timing violators. These nets will then be unnecessarily optimized for timing, increasing runtime, and more seriously, preventing area and power recovery techniques from trading off positive slack for smaller, less power consumptive cells.

## A New Approach – RC-Physical Synthesis Methodology

RC-Physical is a new synthesis methodology introduced in RTL Compiler Version 7.1. The biggest advancement the RC-Physical methodology over the previously discussed methodologies is incorporation of physical data into the synthesis process. The RC-Physical methodology is based on two complementary features: Physical Layout Estimation (PLE) and QoS Prediction.

### Enabling PLE

Experimentation and analysis has shown that using the fanout to predict the net capacitance is no longer an acceptable measure. To accurately predict net capacitances, a model should calculate the net capacitance based on accurate process data and the structure of the path being predicted. Additionally, the model should be adaptive - it should have the ability to recalculate the capacitance as the structure of the path is optimized. The PLE feature in the RC-Physical methodology takes this approach. By reading in the LEF representation of the library along with the capacitance table file, an accurate representation of the process is enabled. The PLE algorithms then adaptively estimate net capacitances based on the structure of the logic being optimized. Testing by Cadence reveals that PLE accurately predicts 80% to 90% of the wires in the design.

PLE is used in place of wire load models to estimate interconnect delay. While this feature can be used without reading in a LEF library, capacitance table or floorplan DEF file, Cadence strongly recommends that PLE be run in conjunction with these files so that the highest level of predictability can be achieved.

The PLE methodology used for the testing section of this paper is shown in Figure 1. SST enabled PLE by adding a few commands to the existing wire-load-model flow. The PLE-specific commands are shown in orange and consist of two basic modifications:

1. In addition to reading technology libraries, read LEF library and capacitance table files for the targeted process node. PLE extracts process information from these files to calculate resistance per unit-length and load per unit-length values. Wire area is also determined based on the average net width defined in the LEF.

   The load information found in a capacitance table can also be found in a LEF file, but capacitance table values tend to be more accurate than LEF values, so RTL Compiler will choose the capacitance table values whenever possible.

   Setting the “lef_library” and “cap_table_file” attributes will trigger RTL Compiler to perform a consistency check between the two files to ensure the number of layers defined in both files is equivalent and that layer widths don’t vary beyond an acceptable tolerance. This is done to ensure the two files are consistent.

   Finally, RTL Compiler will use cell-area values defined in the LEF library over those defined in the technology library. Furthermore, any cell that is defined in the .lib but not in the LEF will not be used by RTL Compiler during mapping.

2. Read in a floorplan DEF file after elaboration. A DEF file contains physical placement information such as die size, pin and macro placement, physical constraints, routing geometry and, possibly, standard cell placement locations.

   Like the LEF and capacitance table files, reading in a DEF file is not required but strongly encouraged as it ensures greater predictability. The floorplan DEF must be read after the design is elaborated. In this figure, setting the "interconnect_mode" attribute to "ple" is shown just before the DEF file is specified. Setting this attribute is not required as "interconnect_mode" will automatically be set to "ple" when the LEF library or capacitance table are read. Once "interconnect_mode" has been set to "ple," RTL Compiler will use Physical Layout Estimation to calculate wire delay for synthesis. Timing reports will also list the "Interconnect Mode" as "PLE."

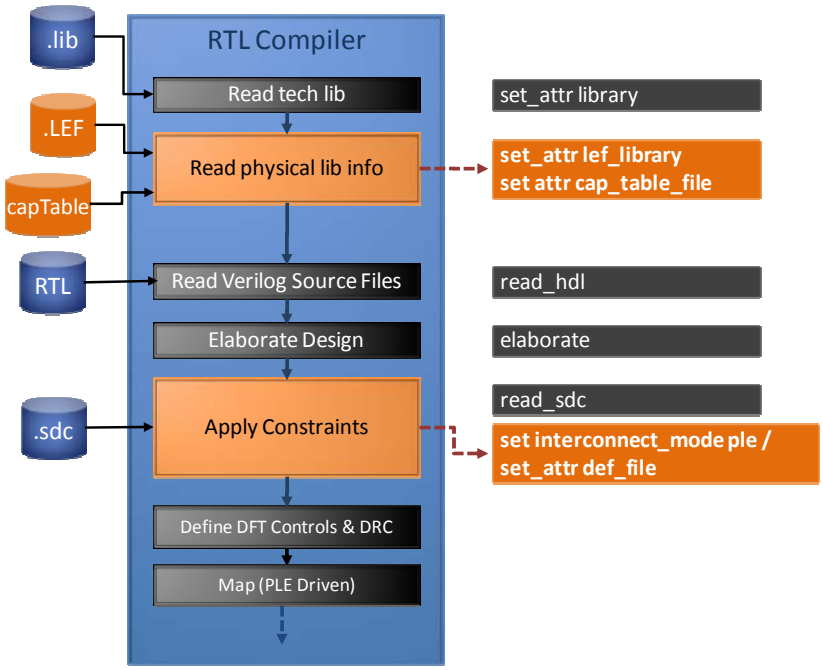



**Figure 1 Enabling PLE**

## QoS Prediction

PLE will generate good delay estimates for 80% to 90% of all the interconnect in a design, but accurate delay measurements for the remaining 10% can only be obtained after gates have been placed and all nets have been routed. The RTL Compiler QoS Prediction feature performs a trial place and route so these nets can be better estimated.

The RTL Compiler "predict_qos" command will invoke the SoC Encounter™ Silicon Virtual Prototyping (SVP) feature in batch mode. SVP is a back-end methodology that will take a synthesized netlist through all the back end steps required to generate extracted parasitics. What differentiates SVP from the standard SoC Encounter methodology is the speed at which SVP generates results. SVP produces parasitic values significantly faster than the standard Encounter methodology. This increased speed comes at a slight decrease in parasitic accuracy, but for the purposes of wire delay estimation during synthesis, the load value accuracy is entirely sufficient.

The physical information generated by SoC Encounter, including a fully placed floorplan DEF file and net load SPEF file are then passed back to RTL Compiler and net delays are adjusted to account for the new annotated loads and placement locations for gates.

**Differences Between PLE and QoS Prediction**

QoS Prediction and PLE both improve interconnect delay estimation accuracy but work in a complimentary manner rather than as alternatives to one another. During the initial structuring phases of logic synthesis, when gates are not available for placement, PLE operates as the exclusive method of wire load delay calculation. But once a design has completed the mapping phase of synthesis, QoS Prediction works in concert with PLE to determine wire delay. The exact method by which a net's delay is calculated is unknown as Cadence considers this proprietary. But the overall strategy involves choosing a net delay based on the PLE value, the back-annotated QoS Prediction value or some combination of the two.

## Enabling QoS Prediction

The primary command that enables QoS Prediction is "predict_qos." As shown in Figure 2, "predict_qos" is called after the design has been mapped. The command will call the SoC Encounter SVP, which will generate net load values based on a trial place and route. Once complete, RTL Compiler performs an incremental synthesis run and generates reports and output files.

Prior to running "predict_qos," a LEF library and capacitance table file must be specified. While a floorplan DEF is not required, Cadence strongly recommends "predict_qos" always be run with a user-supplied floorplan as this will provide a far greater level of accuracy than allowing "predict_qos" to use an auto-created version.

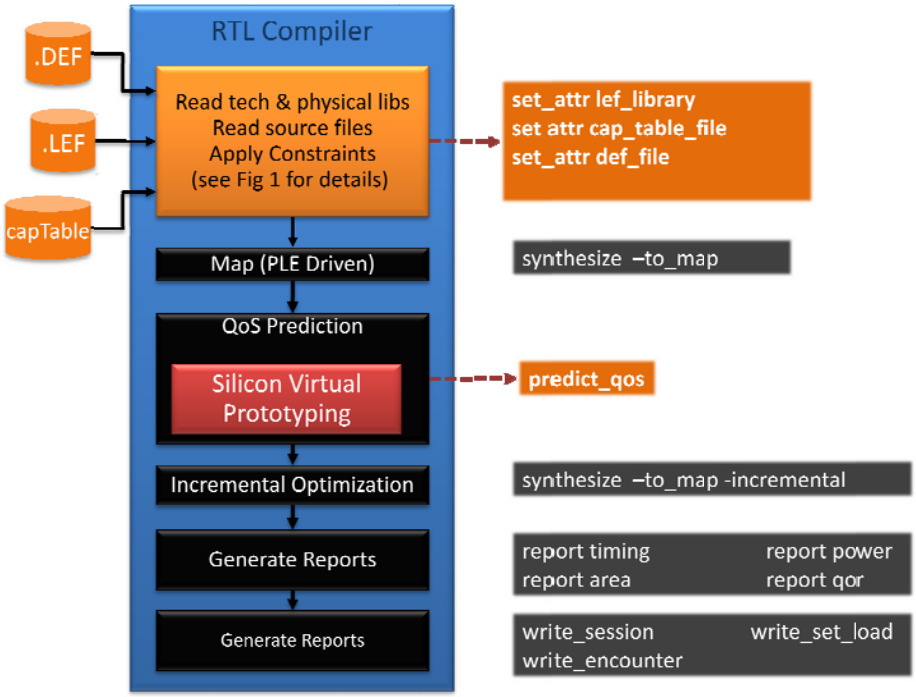



**Figure 2 Enabling QoS Prediction**

## How QoS Prediction Works

As stated earlier, the RTL Compiler QoS Prediction feature works by calling SoC Encounter directly from within the RTL Compiler session. The "predict_qos" command is then used to send the database and necessary physical files to Encounter for rapid placement, routing and parasitic extraction using the SoC Encounter SVP feature.

A step-by-step explanation of how "predict_qos" works is given below:

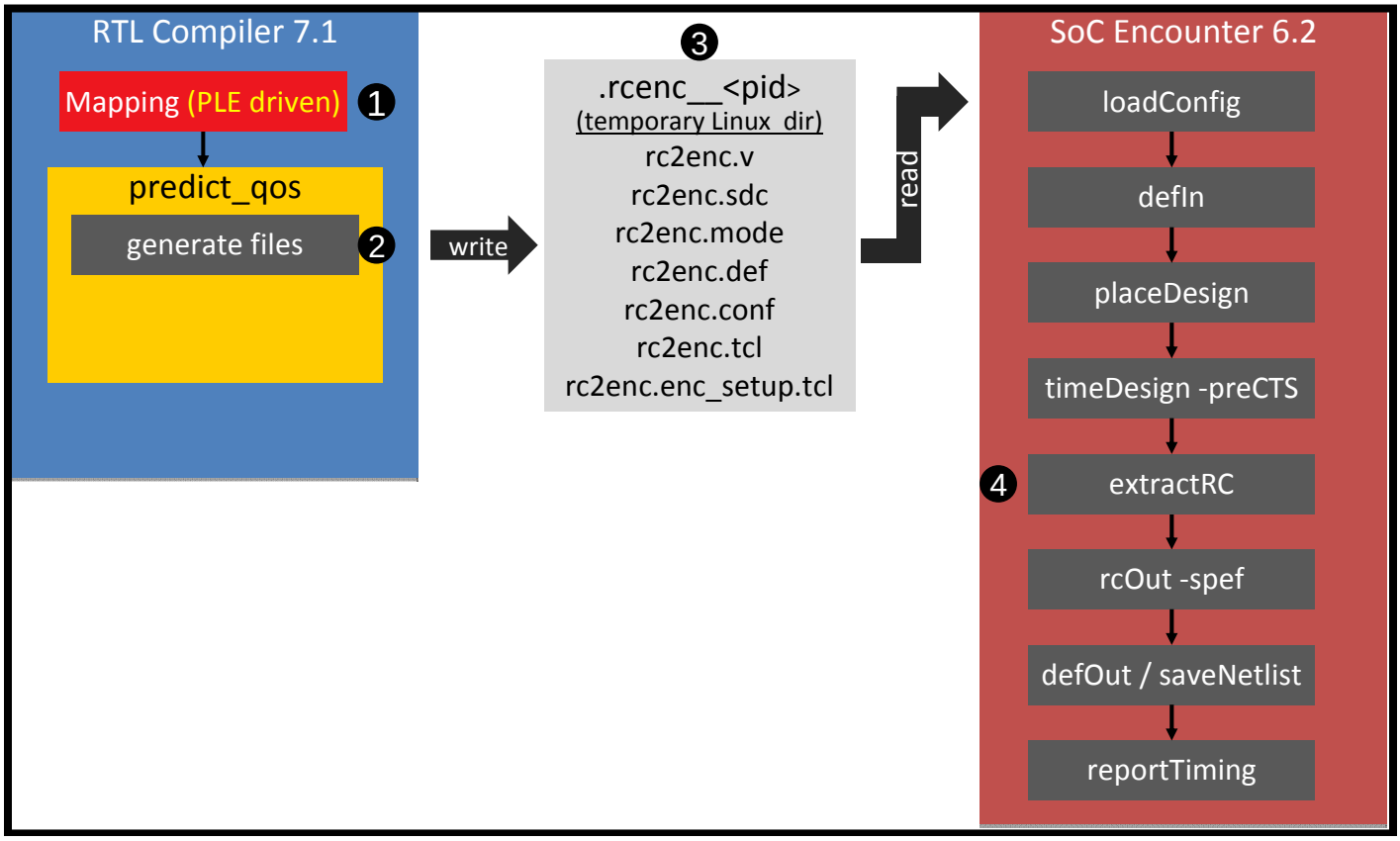



**Figure 3 QoS Prediction: Steps 1-5**

1. Map design to gates
   RTL Compiler maps the design to gates using PLE to estimate interconnect delay.

2. Generate files for Encounter
   The “predict_qos” command is invoked after design mapping. The first operation it undertakes is to generate the files SoC Encounter will require as input. These files include:
   - Mapped netlist (rc2env.v)
   - Design constraints in SDC format (rc2enc.sdc)
   - General mode settings for Encounter (rc2enc.mode)
   - Floorplan (macro placement only) (rc2enc.def)
   - Encounter configuration file (rc2env.conf)
   - Design import script (rc2enc.enc_setup.tcl)
   - Master Encounter run script (rc2enc.tcl)

3. Write files to a temporary directory
   The files generated in step 2 are written to a temporary directory named “.rc__<pid> “ where <pid> represents a unique process ID number.

4. Run Encounter in batch mode
   Encounter is invoked and rc2enc.tcl is sourced. This script commands Encounter to do the following:
   - Load the tech and physical libs, capacitance table and netlist
   - Read in the floorplan (macro and pin placement only)
   - Place the standard cells
   - Perform a trial route
   - Extract interconnect parasitics

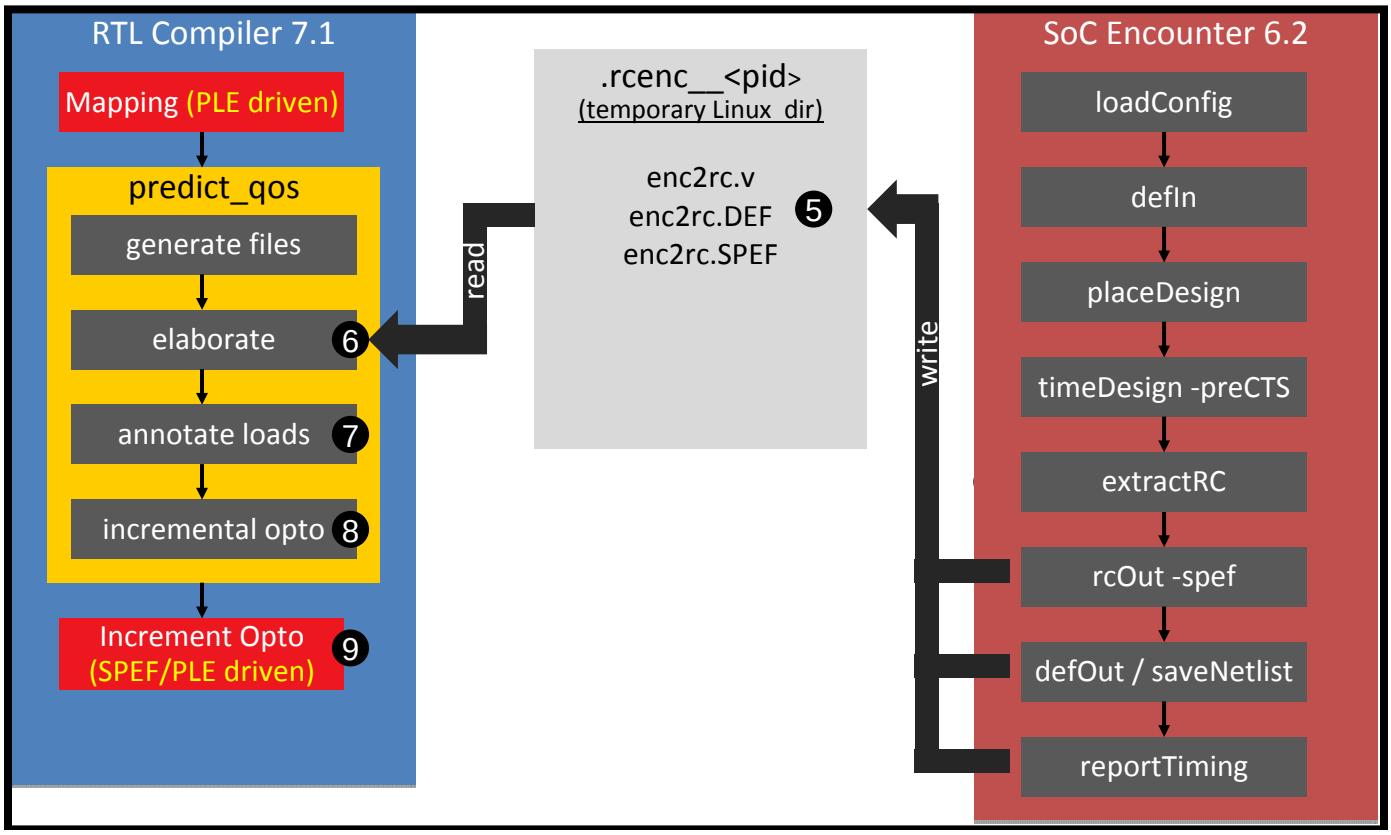



**Figure 4 QoS Prediction: Steps 5 - 9**

5. Write out physical design information
   Once Encounter has finished with extraction, it writes out the following files to the temporary directory:
   - Netlist (enc2rc.v)
   - Fully placed floorplan (enc2rc.DEF)
   - Net load file (enc2rc.SPEF)

6. Elaborate netlist
   RTL Compiler removes the existing version of its design from memory and reads in the Encounter-generated version and elaborates it.

7. Annotate parasitic information
   The SPEF file is then read, annotating nets with their actual delay based on placement.

8. Incrementally optimize
   The final step for "predict_qos" is to initiate an incremental optimization to correct timing issues that have come to light as a result of the new loading and floorplan information.

9. Reoptimize (optional)
   An optional second incremental compile might be desirable if the user adjusts the design constraints after reviewing the results of "predict_qos."

**RC-Physical Benefits**

The biggest benefits this new methodology provides are:

1. It drastically reduces the number of iterations between the front end teams.
   By replacing wire load model estimates with wire lengths based on the actual floorplan, data from a design that has been placed and routed, there is a much stronger correlation between the timing results generated by synthesis and those generated in the back end.

2. The methodology is predictable. Predictability means that the timing results produced after synthesis match very closely with the results produced when the design has been routed. With wire load model based synthesis, designers could not trust their timing reports, and had no way of knowing if this iteration was the final one or not.

# Experimental Testing

## Overview

The two main goals of our experiments were:

1. To measure the quality of results(QoR) produced by RTL Compiler using the wire load, ZWL and RC-Physical synthesis methodologies.
2. For each of the three synthesis methodologies, to measure how accurately RTL Compiler predicted the results produced by SoC Encounter.

## Testing Methodology

Figure 5 illustrates the testing methodology that was employed to generate the experimental data. Two designs were selected as the basis for the experimental work. While they are referred to as "testcases," both designs have previously taped-out and are silicon-proven.

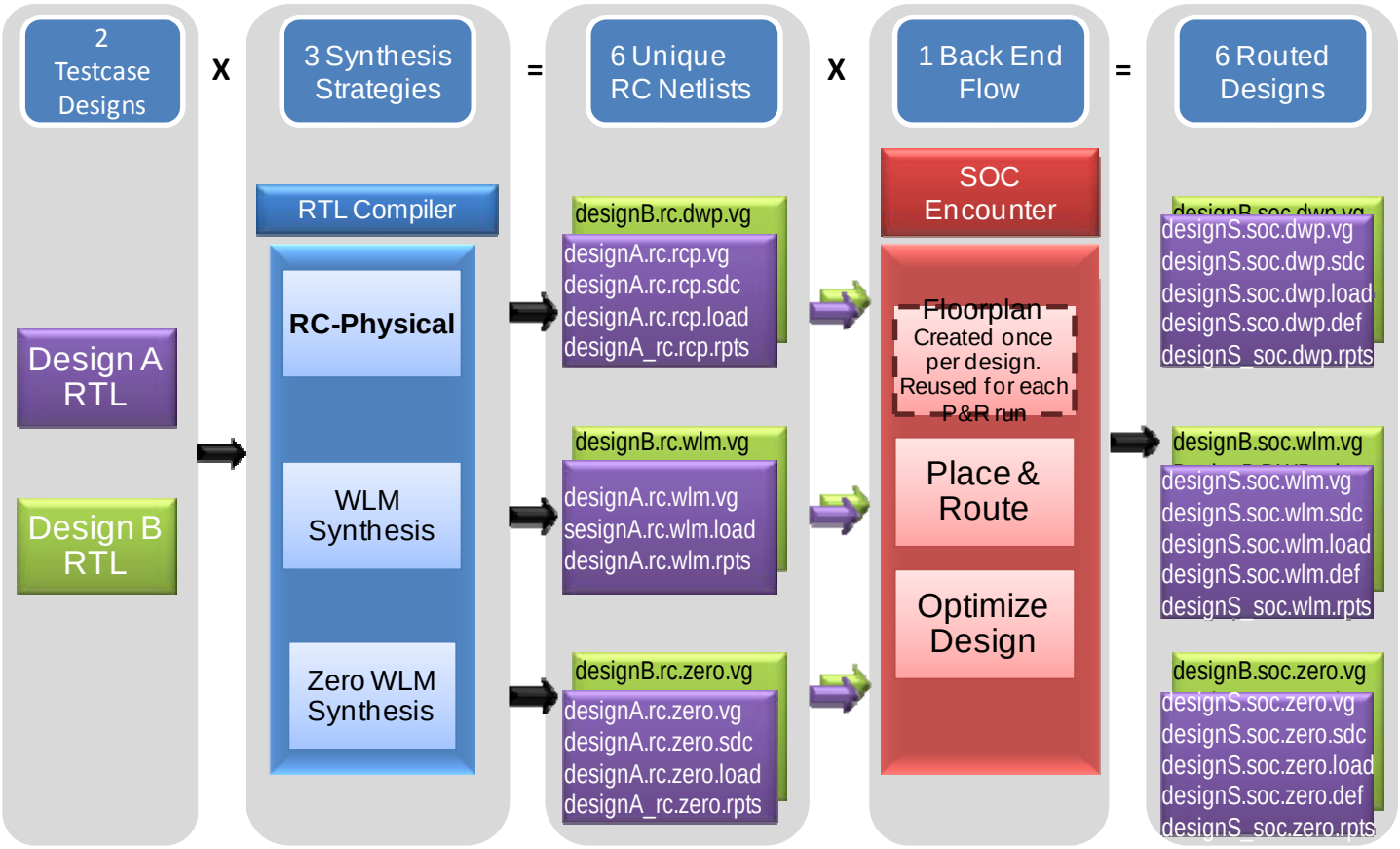



**Figure 5 Testing Methodology**

The RTL for both designs was taken through synthesis, floorplanning, placement, and pre-clock tree trial routing. Had the purpose of this experiment been to close timing on both designs, then the additional steps of clock tree synthesis, global detail route, and 3D extraction would have been included. But the real goals were

to measure the *relative* QoR and predictability of the three synthesis methodologies. To accomplish these goals, terminating the experiment at trial route was deemed sufficient.

For the synthesis phase of the experiment, each netlist was compiled using the RC-Physical, wire -load model and ZWL methodologies. Data relevant to the experiment was then generated, including:

- Final synthesized Verilog® netlist
- Constraint file (SDC)
- Net capacitive load values ("set_load" file)
- Standard QoR reports (timing, area, power)

The two designs being synthesized in three different manners resulted in six netlists and associated data. Each of these netlists was then taken through the back end implementation flow. While there were three variations in how each design was synthesized, all six resulting netlists were taken through the back end process in exactly the same way.

The backend phase began by using the ZWL netlist as the basis to create a production-ready floorplan. After RAM and macro placement was complete and the power grid implemented, the floorplan DEF file was written out. This floorplan file was used for the back end work on the RC-Physical and wire load model netlists as well, ensuring all three methodologies used the same floorplan. Additionally, the floorplan file was used during the RC-Physical synthesis run, as this methodology uses a design's floorplan file to generate optimal results.

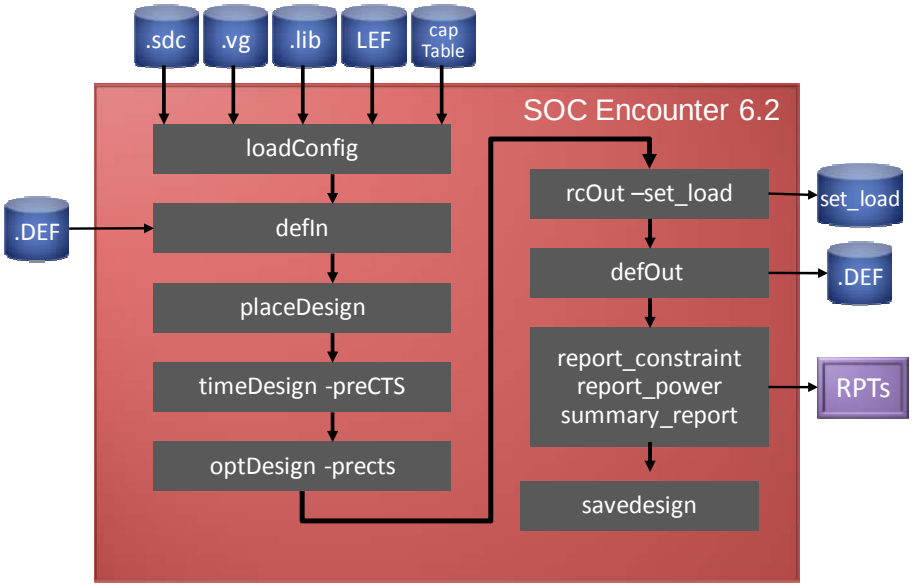



**Figure 6 Back End Methodology**

Following the creation of the floorplan, the back end flow described in Figure 6 was followed. The synthesized netlist, corresponding constraint file and libraries were read into SoC Encounter. The floorplan was then loaded, standard cells auto-placed and a trial route initiated. After the trial route, the netlist was optimized before the "set_load file," reports, and fully placed DEF were written out. The "set_load file" and the reports were used to measure the predictability and performance of the three synthesis methodologies.

**Testcase Designs**

Characteristics for the two testcase designs are show in the Table 1.

| | Design A | Design B |
|---|---|---|
| Basic Function | Network Switch | Encoder/Decoder |
| Design Is Silicon Proven | yes | yes |
| Process Node | 130nm | 65nm |
| Multi-Vt Library | no | no |
| Multi-Supply Voltage | no | no |
| Core Clock Period(s) (ps) | 4500, 5500 | 3000 |
| Instance Count | 170K | 360K |
| Scan Flops Inserted | Yes | Yes |
| Scan Flop Coverage | 98% | 96% |
| RAMs | 125 | 54 |

Table 1 Testcase Design Characteristics

**Measuring Predicability**

Measuring how well RTL Compiler predicted SoC Encounter results was accomplished using two methods:

**Method #1** : Compare the RTL Compiler QoR metrics to those generated by Encounter. One indicator of predictability is how accurately the RTL Compiler QoR measurements track with those generated by Encounter. This was done following the methodology described in Figure 7

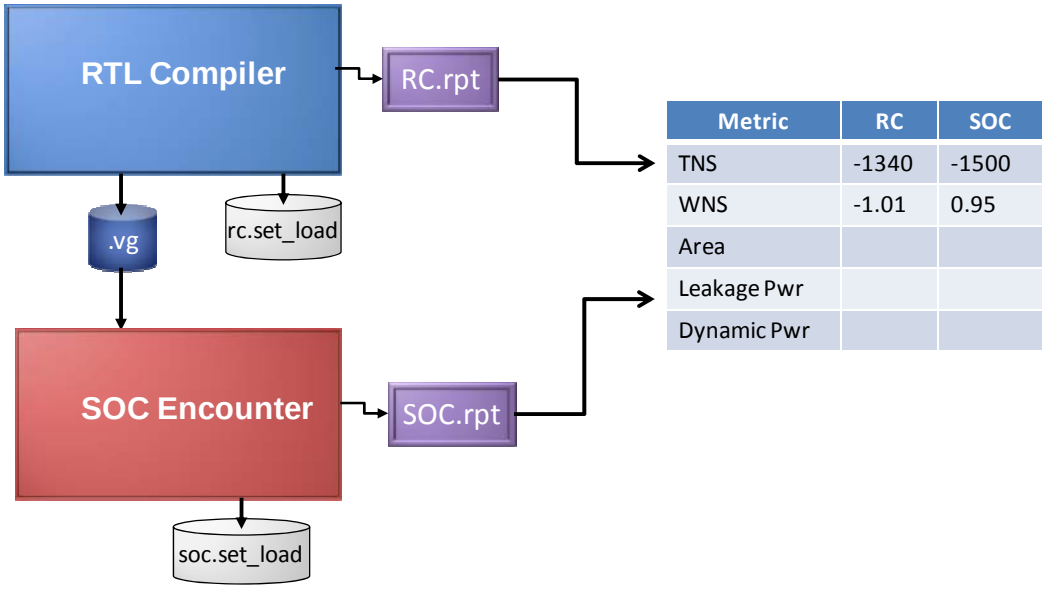



**Figure 7 Measuring QoR Metrics**

The metrics measured were:

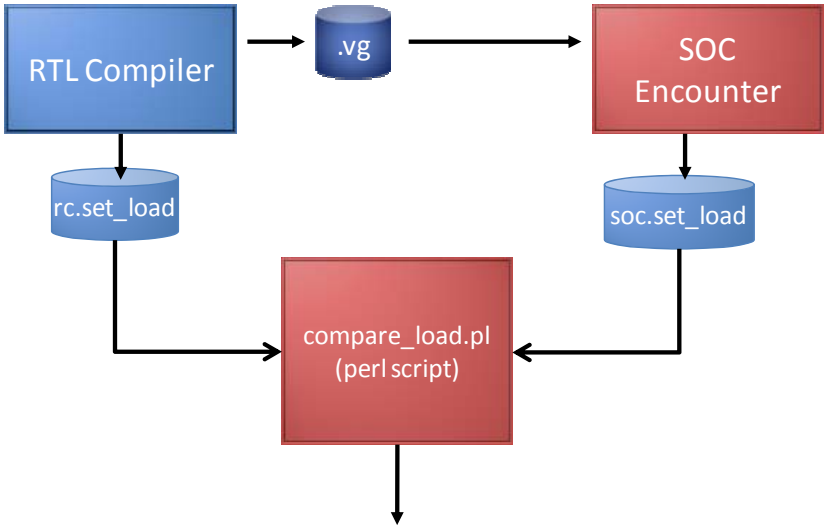
- Worst negative slack (WNS)
- Total negative slack (TNS)
- Design area
- Instance count
- Leakage power
- Dynamic power

WNS and TNS measurements were taken for register-to-register paths only. Dynamic power numbers are rough estimates as they were based on the RTL Compiler default settings for switching probability and switching activity rather than on a switching activity file.

**Method #2 :** Measure net capacitive load values

Another good indicator of predictability is the capacitive load of the nets. Since interconnect delay is a function of net load, comparing the load RTL Compiler calculates for a net to the load value Encounter calculates after "optDesign" will indicate the ability of RTL Compiler to gauge interconnect delay.

Measuring the capacitive load for each net was accomplished using the methodology illustrated in Figure 8




| Net | RC Load | SOC Load | Delta |
|---|---|---|---|
| Net A | 0.033 | 0.030 | 0.003 |
| Net B | 1.0 | 1.200 | -0.2 |

**Figure 8 Measuring Net Load**

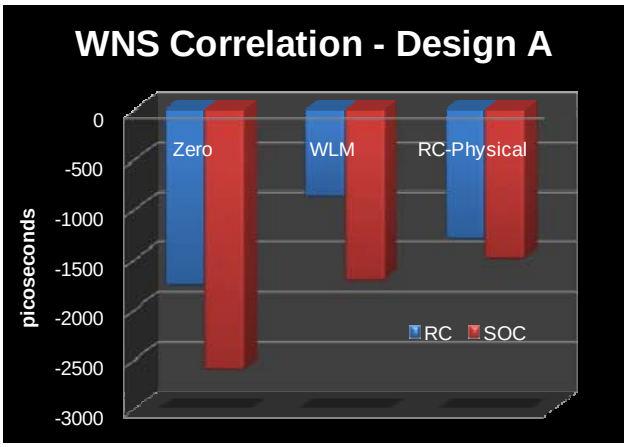
A Perl script was written to parse both "set_load" files, find the common nets, record the load values both tools report and generate a spreadsheet. Perfect correlation occurs if the load that RTL Compiler and SoC Encounter report for every net is identical. To measure how close the three synthesis methodologies came to producing perfect correlation, histograms were created to illustrate the distribution of load-value differences across all the nets in the design.

## Experimental Results

The results of our predictability experiments are show below.

### Part I: QoR Metric Predictability

For each QoR metric, the value produced by RTL Compiler using the three synthesis methodologies is shown. This is followed by the SoC Encounter value for the same metric. The smaller the variation in value, the better the prediction.

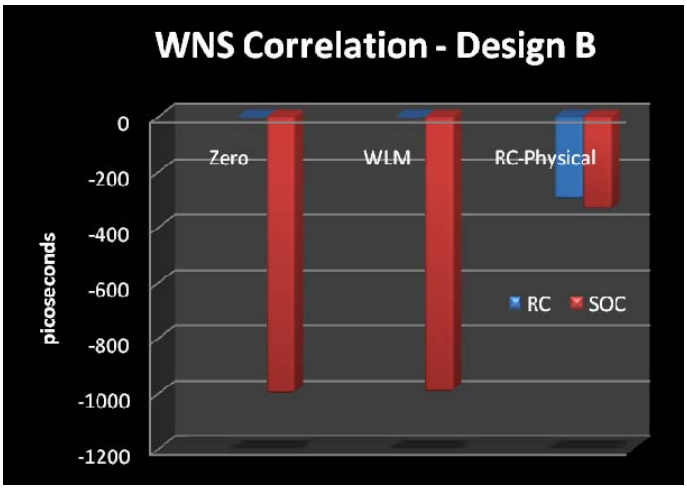



| Worst Negative Slack (ps) - Design A | | | |
|---|---|---|---|
| Synth Methodology | RC | SOC | Variance (%) |
| Zero | -1765.9 | -2612 | -32.4 |
| WLM | -878.6 | -1719 | -48.9 |
| RC-Physical | -1309.2 | -1503 | -12.9 |

| Max Performance - Design A | | | |
|---|---|---|---|
| Synth Methodology | Target Clock Period (ps) | Max Clock Period (ps) | Max Freq (Mhz) |
| Zero | 4500 | 7112 | 140.6 |
| WLM | 4500 | 6219 | 160.8 |
| RC-Physical | 4500 | 6003 | 166.6 |

**Figure 9 WNS Prediction & Performance Data Design A**

Figure 9 shows the WNS numbers that each synthesis methodology generated for Design A. If we limited out QoR analysis to post-synthesis results only, then the WLM synthesis run produced the design with the smallest WNS number (-878.6 ps). This might lead one to believe that WLM synthesis produced the fastest design, however after the netlist has been processed by SOC Encounter, we see RC-Physical actually produced the fastest design with a post-SOC WNS number of -1503ps. Furthermore, RC-Physical was far more accurate in predicting the final WNS number as well with a variance of 12.9% compared to over 30% for ZWL and nearly 50% for WLM.

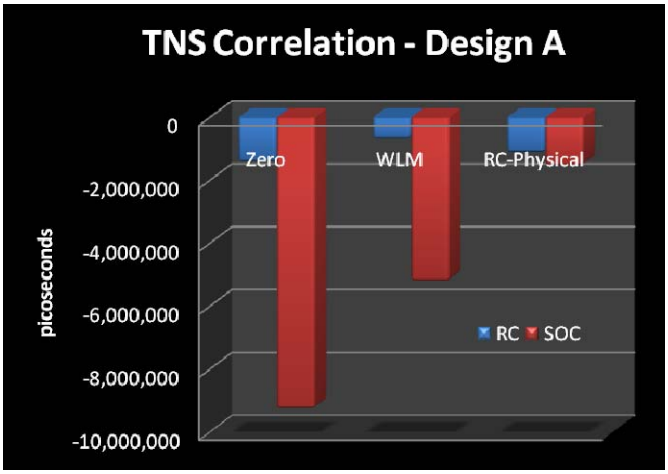


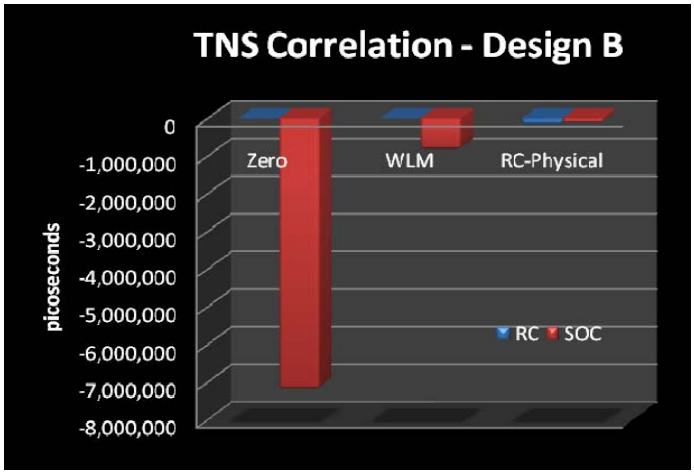

| Worst Negative Slack (ps) - Design B | | | |
|---|---|---|---|
| Synth Methodology | RC | SOC | Variance (%) |
| Zero | 0 | -1003 | -100.0 |
| WLM | 0.3 | -997 | -100.0 |
| RC-Physical | -299.7 | -336 | -10.8 |

| Max Performance - Design B | | | |
|---|---|---|---|
| Synth Methodology | Target Clock Period (ps) | Max Clock Period (ps) | Max Freq (Mhz) |
| Zero | 3000 | 4003 | 249.8 |
| WLM | 3000 | 3997 | 250.2 |
| RC-Physical | 3000 | 3336 | 299.8 |

**Figure 10 WNS Prediction & Performance Data Design B**

**Figure 10** shows the WNS results for Design B. Like Design A, RC-Physical produced the fastest design with a post-SOC WNS of 336ps – about one third the value produced by the other methodologies. Similarly, RC-Physical was far more accurate in predicting the final WNS number with a variation of approximately 11%. WLM and ZWL predicted the design would either meet timing or be off by a third of a picosecond. In reality the WNS was almost 1ps for a 3ps clock!



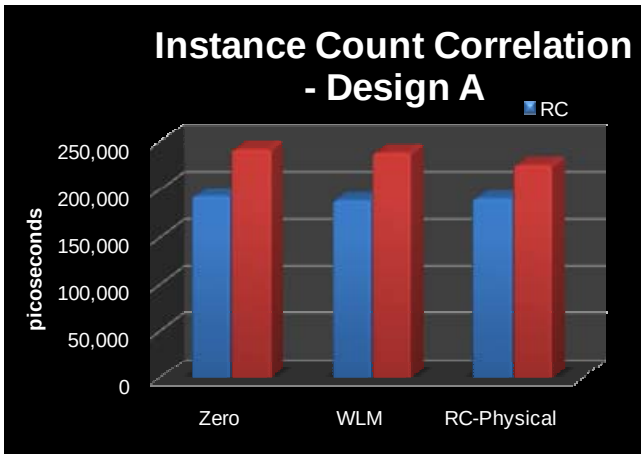



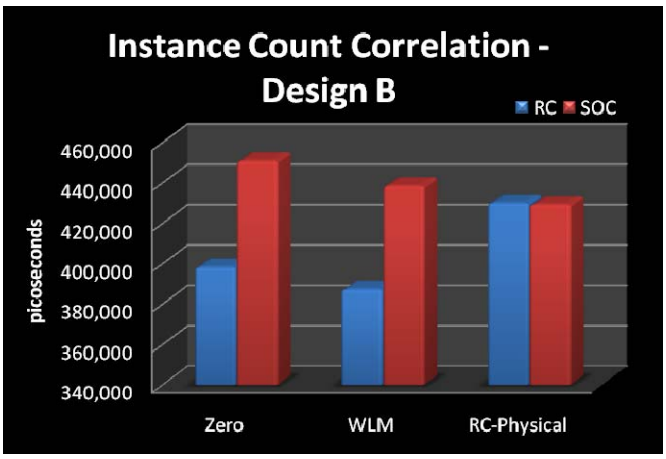

| Total Negative Slack - Design A | | | |
|---|---|---|---|
| Synth Methodology | RC | SOC | Variance (%) |
| Zero | -1,429,286 | -9,270,300 | -84.58 |
| WLM | -682,038 | -5,216,300 | -86.92 |
| RC-Physical | -1,121,632 | -1,546,500 | -27.47 |

| Total Negative Slack - Design B | | | |
|---|---|---|---|
| Synth Methodology | RC | SOC | Variance (%) |
| Zero | 0 | -7,204,400 | -100.00 |
| WLM | 0 | -844,137 | -100.00 |
| RC-Physical | -185,181 | -137,606 | 34.57 |

**Figure 11 TNS Prediction Data**

The TNS numbers for both designs are shown in Figure 11. Notice that for both designs RC-Physical produces the smallest TNS numbers after the design has been processed by Encounter. From a predictability point of view, RC-Physical had variations of approximately 27% & 35% for designs A & B, respectively. This was significantly better than WLM and ZWLM which had variations of 85% to 100%.



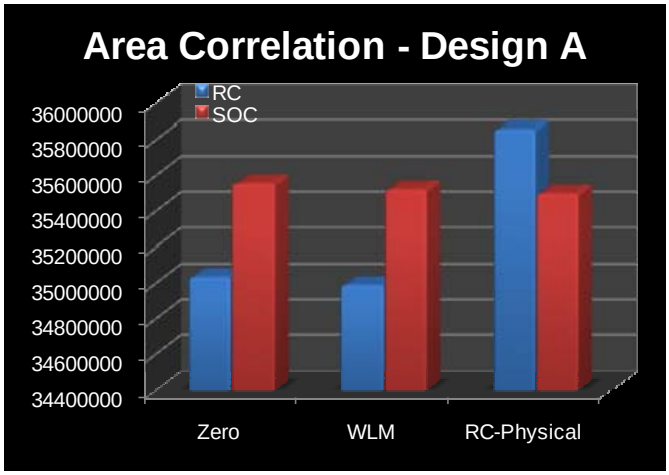



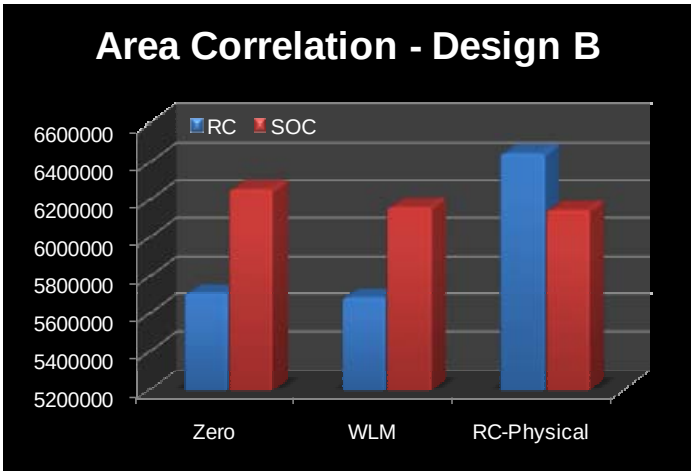

| Instance Count - Design A | | | |
|---|---|---|---|
| Synth Methodology | RC | SOC | Variance (%) |
| Zero | 192,682 | 242,353 | -20.50 |
| WLM | 188,389 | 239,057 | -21.19 |
| RC-Physical | 190,131 | 226,254 | -15.97 |

| Instance Count - Design B | | | |
|---|---|---|---|
| Synth Methodology | RC | SOC | Variance (%) |
| Zero | 398,143 | 450,808 | -11.68 |
| WLM | 387,215 | 438,414 | -11.68 |
| RC-Physical | 429,934 | 429,250 | 0.16 |

**Figure 12 Instance Count Prediction Data**

The total number of instances created by each design was consistent with previous results – RC-Physical produced designs with the fewest number of instances and it was the most accurate synthesis methodology when it came to predicting what the post-SOC instance count was. In the case of Design B, RC-Physical predicted the final number almost perfectly.



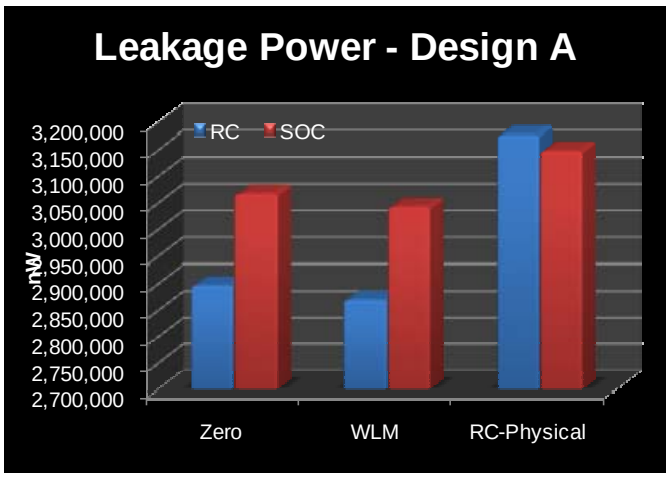



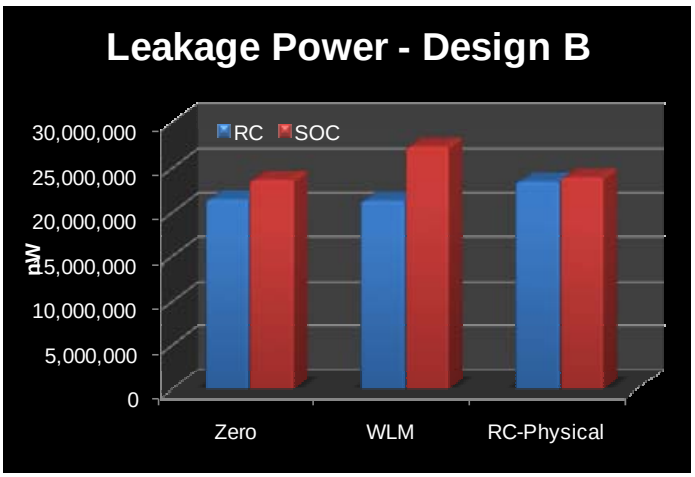

| Area - Design A | | | |
|---|---|---|---|
| Synth Methodology | RC | SOC | Variance (%) |
| Zero | 35,034,968 | 35,555,027 | -1.5 |
| WLM | 34,989,550 | 35,521,032 | -1.5 |
| RC-Physical | 35,858,842 | 35,498,556 | 1.0 |

| Area - Design B | | | |
|---|---|---|---|
| Synth Methodology | RC | SOC | Variance (%) |
| Zero | 5,708,741 | 6,258,129 | -8.8 |
| WLM | 5,686,524 | 6,162,961 | -7.7 |
| RC-Physical | 6,449,804 | 6,148,593 | 4.9 |

**Figure 13 Area Prediction Data**

The area numbers shown in Figure 13 further demonstrate that RC-Physical is the best methodology when comparing design size and correlation with final physical design numbers.



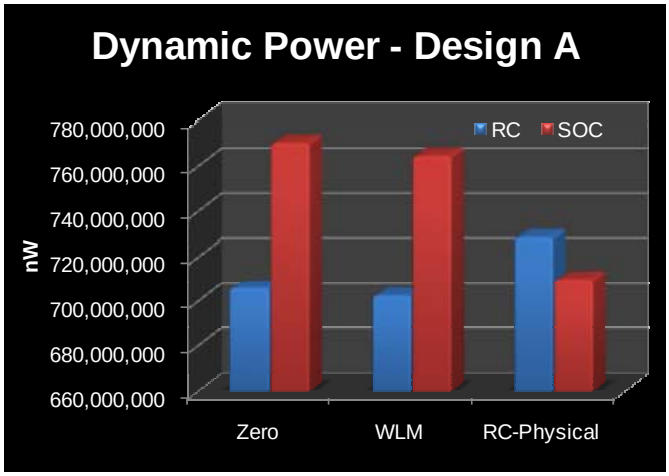



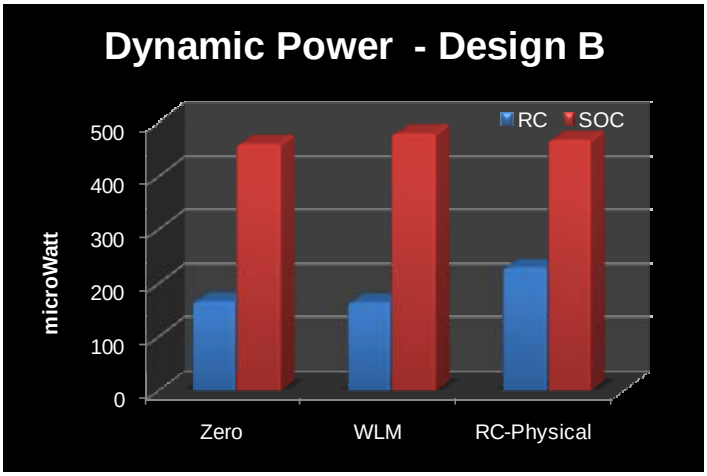

| Leakage Current (nW) - Design A | | | |
|---|---|---|---|
| Synth Methodology | RC | SOC | Variance (%) |
| Zero | 2,897,507 | 3,067,821 | -5.55 |
| WLM | 2,870,407 | 3,042,765 | -5.66 |
| RC-Physical | 3,177,894 | 3,149,049 | 0.92 |

| Leakage Current (nW) - Design B | | | |
|---|---|---|---|
| Synth Methodology | RC | SOC | Variance (%) |
| Zero | 21,494,055 | 23,689,264 | -9.27 |
| WLM | 21,348,786 | 27,438,608 | -22.19 |
| RC-Physical | 23,497,194 | 23,965,839 | -1.96 |

**Figure 14 Leakage Power Prediction Data**

Leakage power numbers show that the WLM design produced the least power-consumptive netlist for Design A. However ZWLM and RC-Physical netlists produced designs with only slightly more power. For Design B, RC-Physical produced the least power-consumptive design and for both designs, RC-Physical had the least amount of variation between synthesis results and SOC Encounter results.




| Dynamic Power (nW) – Design A | | | |
|---|---|---|---|
| Synth Methodology | RC | SOC | Variation (%) |
| Zero | 705,876,554 | 770,393,478 | -8.37 |
| WLM | 702,497,998 | 764,619,809 | -8.12 |
| RC-Physical | 728,521,983 | 710,065,116 | 2.60 |

| Dynamic Power (nW) – Design B | | | |
|---|---|---|---|
| Synth Methodology | RC | SOC | Variation (%) |
| Zero | 166,698,734 | 461,485,386 | -63.88 |
| WLM | 164,720,532 | 480,983,077 | -65.75 |
| RC-Physical | 229,548,980 | 469,328,523 | -51.09 |

**Figure 15 Dynamic Power Prediction Data**

For Design B, RC-Physical's dynamic power estimation was better than WLM and ZWL, however the variation (51%) was much larger than variations for any of the other QoR metrics. We attribute this to the fact that the dynamic power numbers are not accurate. Dynamic power consumption is calculated using RC's default switching probability values, which is a far less accurate way of measuring power consumption than using a switching activity file. Had a switching activity file been used, we believe RC-Physical would produce results consistent with previous QoR metrics – power consumption would be least of the three methodologies and have the smallest variation.

# Net Load Prediction Data

The second part of predictability analysis looks at net capacitive load. Looking at net loading is not only an indicator of timing correlation (since net loading directly effects net delay), it also provides a much larger sample of data to measure prediction against. QoR metrics indicate prediction accuracy. Looking at the load values of hundreds of thousands of nets indicates prediction consistency.

### Generating Histograms

As stated earlier, capacitive load comparison is based on the "set_load" files generated by RTL Compiler and SoC Encounter. A Perl script processes both files, and generates a table consisting of the net name and the difference between the RTL Compiler "set_load" value and the SoC Encounter "set_load" value for each respective net. The output file listed over 200,000 nets for each of the two designs, thus the most practical way of evaluating so much data is to use histograms that show the distribution of RTL Compiler/SoC Encounter load differences across all nets.

The process of generating the histogram is shown in Figure 16. The net load delta values are distributed into bins which represent a range of load values. For the charts that follow, 100 bins representing a 0.001 pico Farad (pF) range were used. The number of nets whose load delta value corresponds to each bin is then calculated and the histogram created as shown in **Error! Reference source not found.** .

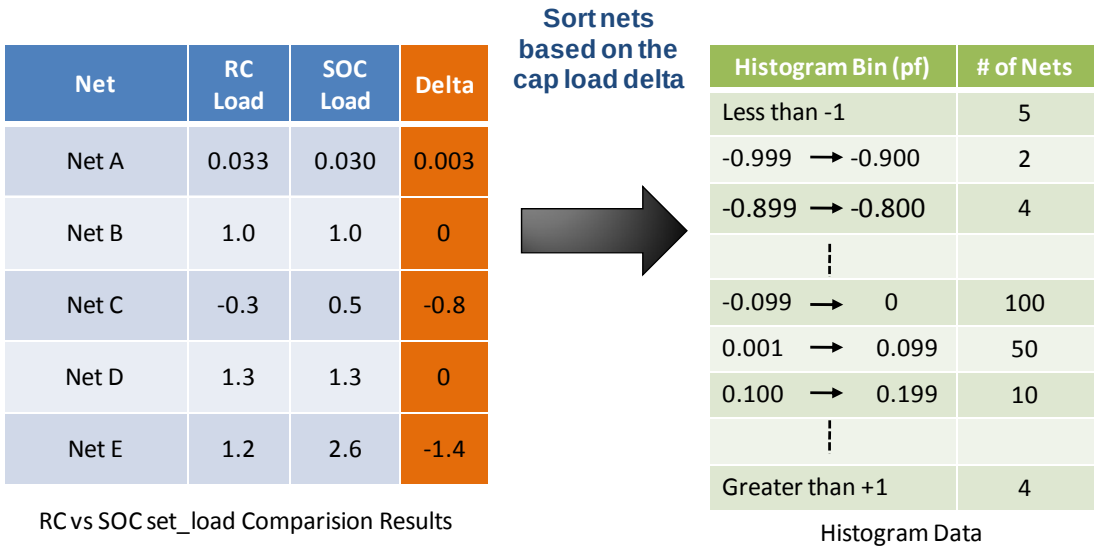



**Figure 16 Generating Net Load Histogram Data**

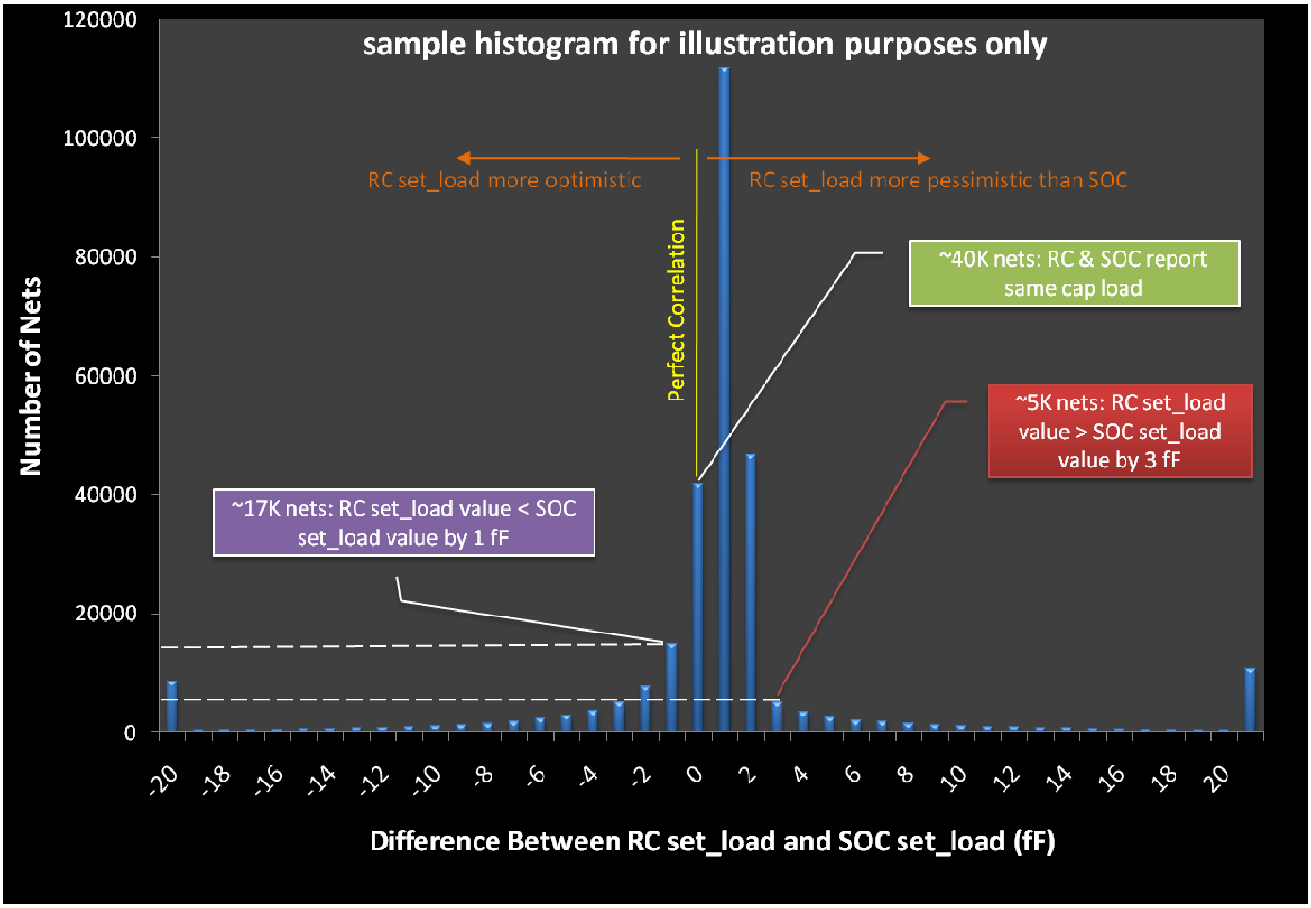



**Figure 17 Net Load Histogram Example**

## Interpreting The Histogram

There are two characteristics of the histogram that are useful for analyzing predictability: how well is the graph centered on a delta of 0pF, and how wide is the delta distribution. Both concepts are shown in . A histogram centered on zero with a tight distribution is one indicator of a predictable synthesis methodology.

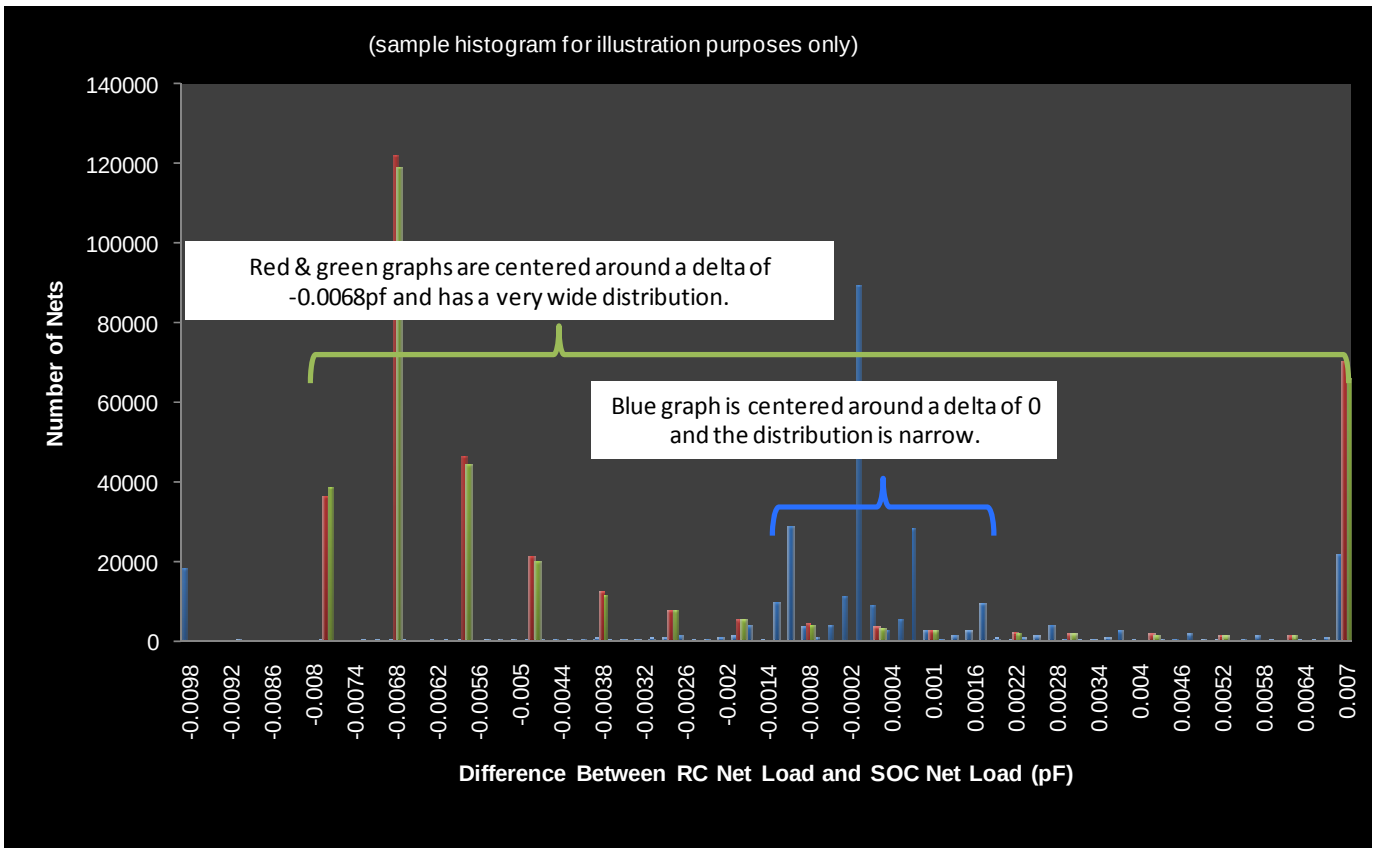



**Figure 18 Interpreting A Histogram**

**Histograms For Designs A & B**

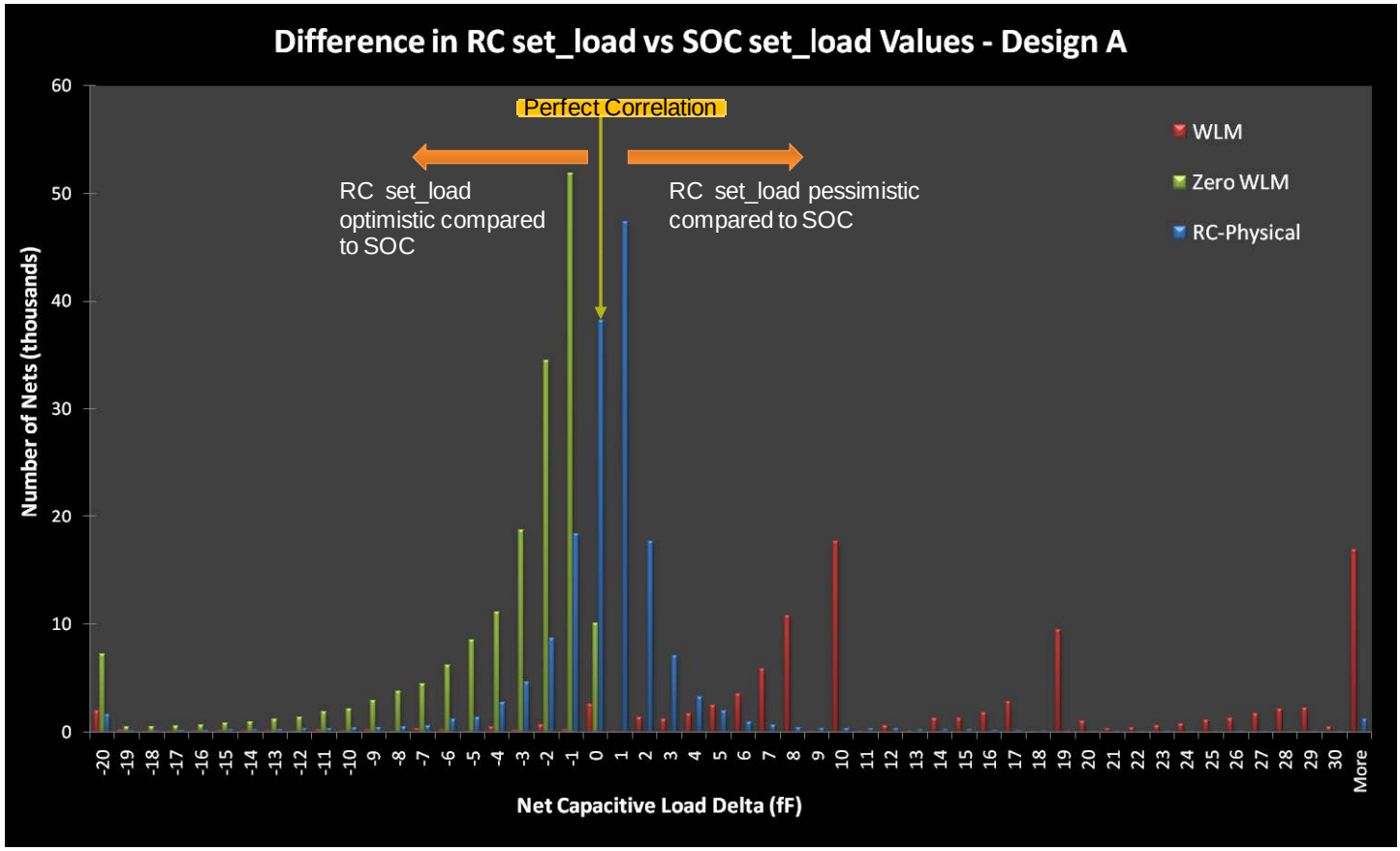



**Figure 19 Design A Net Load Histogram**

**Design A Analysis**

Three histograms are shown in Figure 19, each representing the distribution of net load differences for Design A. The wire load model-based synthesis run clearly produced the worst results as indicated by the wide distribution of histogram values.

The ZWL synthesis histogram is essentially the SOC Encounter load values because any net synthesized with a zero wire load model will have a capacitive load of 0 (after synthesis). This results in better correlation than WLM, however, the distribution of the ZWL graph is quite wide, spanning from 0fF to -20fF, which indicates a large amount of variability. The ZWL run will generate some net loads that correlate very well with SoC Encounter results but it will also generate an even greater number of net loads that vary significantly.
The RC-Physical graph is centered around 0 pF with a much narrower distribution of paths between -3fF and +3fF. There are paths beyond this range, but they are not statistically significant.

## Design B Analysis

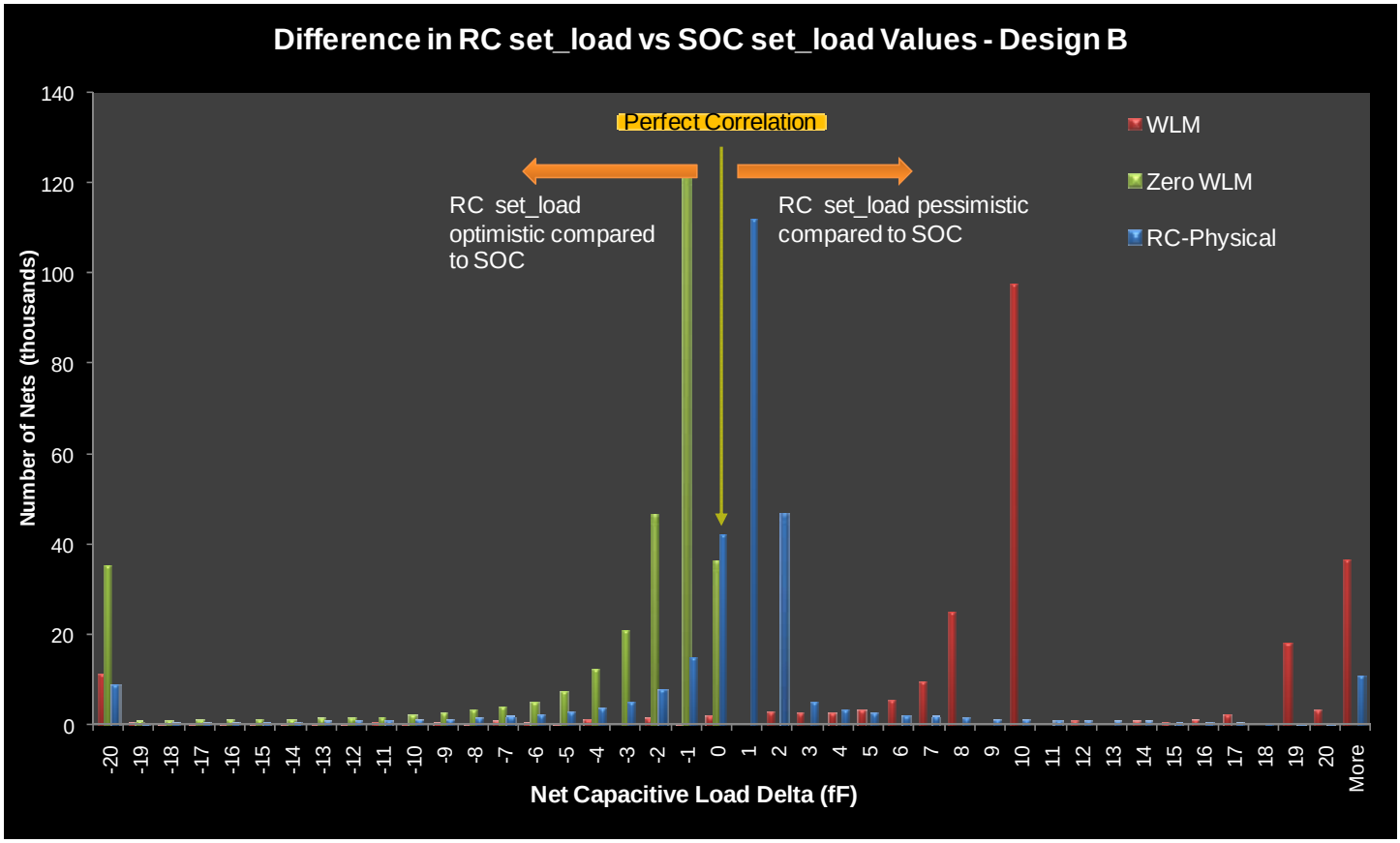



**Figure 20 - Design B Net Load Prediction Data**

Figure 20 shows the three synthesis methodology histograms for Design B. Like Design A, the WLM histogram is clearly the worst with a large distribution centered around 10fF. The ZWL histogram correlated better, but still had a large distribution of values. RC-Physical shows a net load graph centered at 0fF with a narrow distribution between -2pF and +2pF.

## Overall Conclusion

When examining each testcase design AFTER trial route & optDesign:

- RC-Physical produced designs are consistently smaller, faster and consume less leakage current
- The dynamic power number for Design B was the only metric RC-Physical didn't produce better results for. Since no switching activity file was used to calculate dynamic power, the power numbers are likely inaccurate. It is reasonable to expect actual dynamic power consumption to track the area and instance count trends

When examining the variation in RC QoR data versus SOC Encounter QoR data:

- RC-Physical produced far more predictable results than WLM and ZWLM methodologies

- RC-Physical WNS was within 13% of SOC
- RC-Physical TNS was within 35% of SOC
- RC-Physical instance count was within 16% of SOC
- RC-Physical area & leakage were within 5% of SOC

RC-Physical benefits the logic design engineer by generating synthesis results that are significantly closer to their final numbers than what WLM or ZWLM can produce. Logic designers can have a higher degree of confidence that the results they see in synthesis will hold through place and route.

RC-Physical benefits the physical design engineer by creating a netlist that is a better starting point than the WLM or ZWLM netlists. A better start point means timing closure is achieved with less engineering effort and fewer iterations.