Quo Vadis Power?
Early Power Assessment over Development Cycle

Author: Carsten Rau (Infineon Technologies AG)

Infineon Technologies AG

Neubiberg, Germany

ABSTRACT

"Shift left" and "Find issues early" are today's buzzwords. In this presentation we will share some experience on how SpyGlass Power Estimate / Reduce was used during the development of a design to provide early visibility on power trends.

We will spend some time to describe some of the findings we made with the tool, and focus on the continuous tracking, trend analysis and respective enhancements done to the flow

Table of Contents

Table of Figures

Table of Tables

1. Introduction

For most applications still primary focus is timing and area.
Of course there are the dedicated applications where power is the key topic, but also for the other kind of designs the importance of power is increasing and hardly any product these days can neglect power. This is partly due to that fact that high power consumption is making implementation more complex (e.g. Voltage drop and power mesh dimensioning), but partly also due to power finally being something that customers care more and more for and certain limits have to be kept.

Hence also management starts to ask early on about power consumption and if the design will meet the target that we aimed for. Power estimation on RTL cannot be as accurate as power estimation on the final layout netlist, but the main benefit is its early availability and its use as an indicator where power is going to end up.
Making use of this indicator and reporting a trend is an important benefit and the main topic of this paper.

This paper is one result of the application of SpyGlass Power Estimate/Reduction Feature on one of our products from first RTL to final RTL. Its focus is not the actual power saving techniques features used in the tool to reduce power, the focus is on using the given infrastructure to as easy as possible being able to provide a comprehensive trend to management.

1.1 Motivation and Design

The design is a digital controller for power management application. A predecessor was already done in a 130nm technology, but with a reduced feature scope.
For the new design there was following change:

- Design is now on a 65nm node
- Its target use case was widened hence more features requiring new or modified IP
- In those new target use cases, completely new low power targets where given by marketing/customers, up to 10X lower than was required for the previous design

This raises of course several questions:

- Without good incremental power estimates, due to different design features and technology, how do we do early power estimation?
- Can the existing architecture of the predecessor still be adjusted to the newer applications and still meet the new power targets?
- Do we need a major change in the design?

So it was very important to get early on an outlook about dynamic power consumption of the first RTL. Designing for power has impact on schedule as designers have to consider it more carefully during coding and its measures have negative effect on area and/or timing.

SpyGlass RTL Power Estimate/Power Reduction was used to assess design for power on RTL and due to the fact that new design features are getting added to the IPs, to carefully judge and detect their impact on power and this is best done via continuous tracking.

In the end no major re-architecting was needed and the final design had following aspects:

- Digital area approximately 1 million gates
- Multiple clock frequencies in the range 1 – 200 MHz
- Digital area is single 1.2V digital supply domain

- No power gating used,
  modules are only disabled via functional clock and data gating,
  focus was just dynamic power, not leakage power,
- 5 different "power modes" ps0 to ps4, where
  - Ps0 is the highest active mode with full chip operational and fully active
  - Ps4 is the lowest power mode with only a wake up controller active, other modules are functionally disabled (clock gating)

1.2 Tool Introduction

As mentioned the tool used for the RTL power assessment, analysis and tracking is SpyGlass. Respective features are called Power Estimate/Power Reduction.

- Power Estimate: pure power estimation on RTL
- Power Reduction: assessment of RTL for power and thus helping to identify room for improvement in design (like enhancing or correcting clock gating)

SpyGlass is a Rule based tool, and uses specific terminology, which shortly should be explained in this table:

Table 1 SpyGlass Terminology

Terminology	Meaning
Rule	This is one check or analysis to execute. If Rule is selected, SpyGlass will execute whatever this Rule does. This can be a linting check, a CDC check, or a specific Power estimation. Rules may generate reports in all kind of formats and raise INFO/WARNING/ERROR messages/violations
Parameter/options	Each rule is affected by parameters/options which are set to a default. End user can and often should configure the rule to some extend via parameters/options for specific purposes or requirements
constraints	SpyGlass constraints describe usually design aspects and are also a way to affect the tool and its rules. In case of Power e.g. also heuristics about the design are given to be used for RTL power estimation
Project File	Central SpyGlass setup file containing all user customization of SpyGlass setup
Goal	A goal is a collection of rules. SpyGlass will always execute Goals and thus directly run multiple rules in one go. The various aspects of Power estimate/reduction require several goals to be executed as some rules cannot be combined in same run.
Scenario	A scenario is a derivative of a goal. One can simply execute an existing goal, only with slightly different modification to ruleset, parameters or constraints. This way one can easily generate variants and compare them together without overwriting each other
Methodology	Methodology is a collection of goals. GuideWare is what the default methodology is called that is provided with a tool installation, but also end-users can configure and set their own methodology.

2. Reporting, Visualization and Exploration

2.1 Native Tool Tracking

SpyGlass offers already per default following reports relevant/usable for abstract power assessment tracking:

- pe_summary report (ASCII/CSV)
- Power Explorer (GUI/CSV)
- pe_enable_scorecard.csv (GUI/CSV)
- HTML Dashboard

2.1.1 Pe_summary Report

```
[…]
Power Summary
,Leakage,Internal,Switching,Total
Total Power,2.467e-04,2.771e-02,3.644e-03,3.160e-02
Combinational Power,1.760e-06,3.300e-04,1.742e-03,2.073e-03
Sequential Power,9.192e-07,1.991e-04,1.395e-04,3.395e-04
Black Box Power,0.000e+00,0.000e+00,0.000e+00,0.000e+00
Memory Power,2.439e-04,2.469e-02,2.026e-04,2.514e-02
IO PAD Power,0.000e+00,0.000e+00,0.000e+00,0.000e+00
Clock Power,1.143e-07,2.488e-03,1.561e-03,4.049e-03


Report Format
Instance(Module),Leakage,Internal,Switching,Total

top,2.467e-04,2.771e-02,3.644e-03,3.160e-02
top.inst1(modulename1),2.456e-04,2.705e-02,1.064e-03,2.836e-02
[…]
```

(not from this design)

Figure 1 pe_summary Report Example

- Hierarchical power report
- Automatically generated each time.
  Rule PEPWR02 is run which is the power estimate rule
- Contains additional information (like setup and used activity file)
- In older SpyGlass reports this was the basis for the power report in the SpyGlass GUI
- Can be configured to use absolute numbers and csv format, which is configured that way at Infineon to allow better parsing with custom scripts and reuse in Excel

2.1.2 Power Explorer

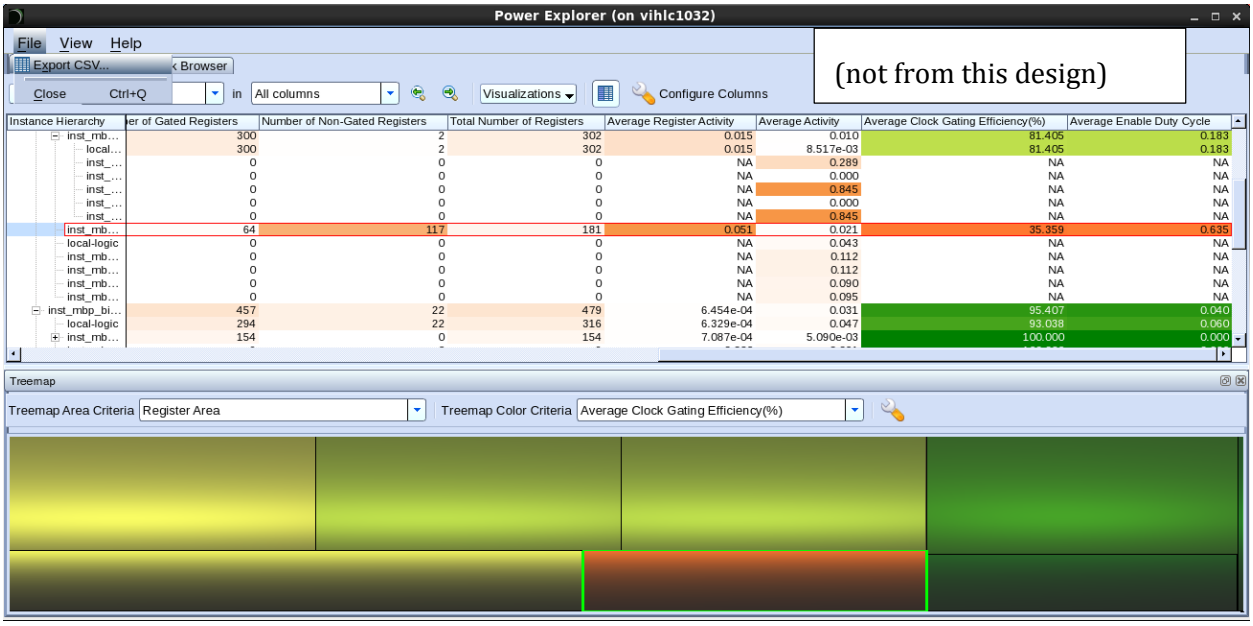

Figure 2 Power Explorer GUI

- Not only hierarchical power report, but all attributes applicable for instances available in one large combined view
- Additional visualization like pie chart and tree map available
- Exportable as CSV format
- Drawback: CSV generation not available per batch mode in current tool version, only from GUI. Hence not usable for any automatic generic tracking.

2.1.3 Pe_enable_scorecard.csv Report

A DesignUnit	B Instance Count	C Total Register Coun	D Registers with Enable	E Unique Enables Count	F Registers with Enable(%)	G Clock Gating Efficiency(%)	H Clock Activity Distribution(%
[redacted]	1	1898	1758	110	92.624	79.284	100
	1	1409	1293	56	91.767	73.94	93.387
	1	185			59	39.459	28.485
	1	306			73	76.308	18.439
	1	306			73	76.309	18.438
	1	306	305	12	99.673	81.977	14.026
	1	306	305	12	99.673	82.012	13.999
	1	481	465	54	96.674	96.258	4.578

(not from this design)

Figure 3 pe_enable_scorecard.csv Example

- Generated automatically as CSV file whenever PESTR06 or PEPWR06 Rule is executed
- Contains clock gating statistics per module/instance

2.1.4 HTML Dashboard

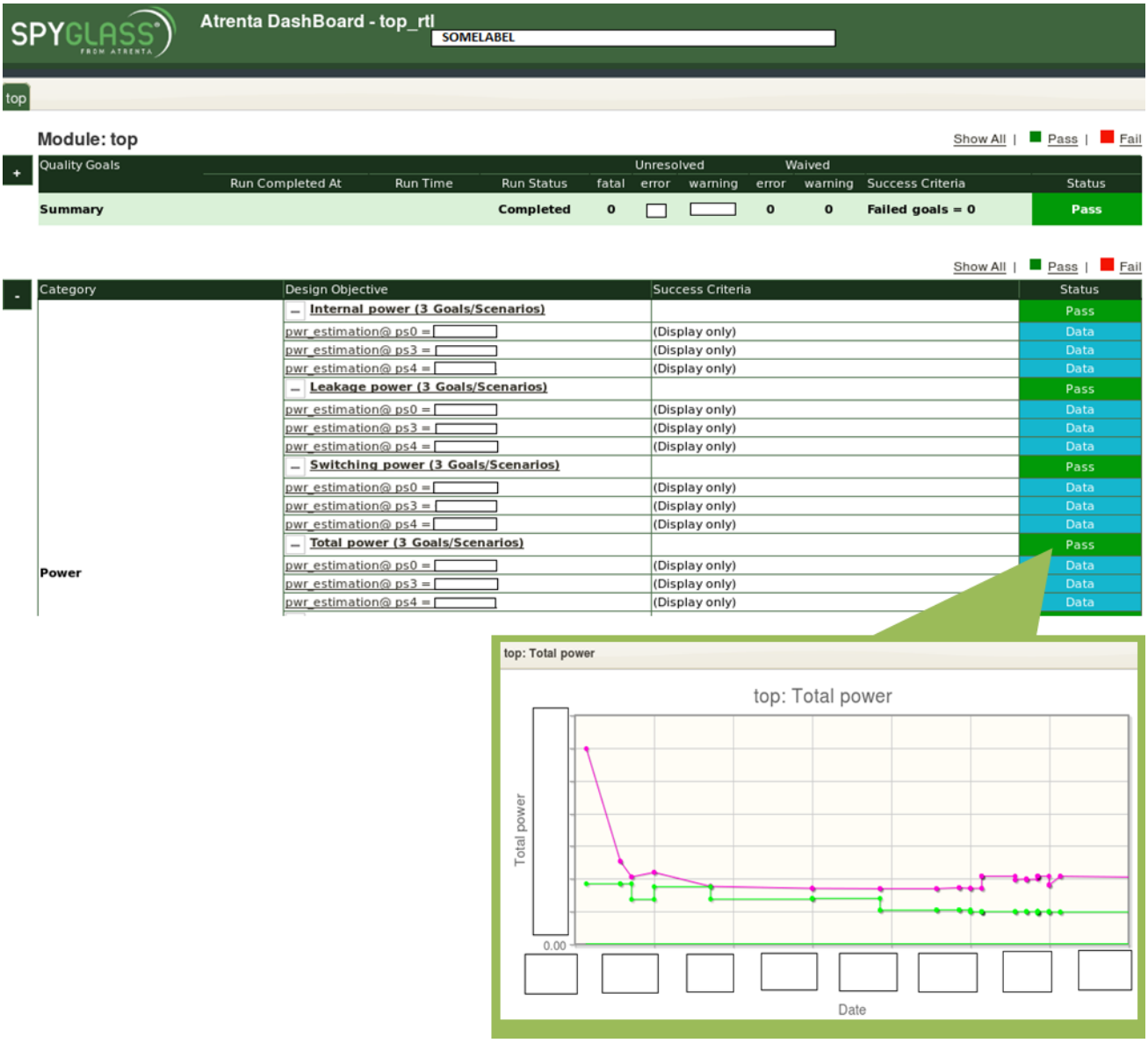

Figure 4 HTML Dashboard

- Configurable "overview" in HTML format
- Automatic datamining of important results (e.g. top level power results, ASCII reports)
- Provision of history and trend over time.
- Drawbacks
    - Only available for top level attributes
    - Only available for attributes defined by Vendor

In Figure 4, you see results of the goal used for power estimation and respective scenario.
Each entry named like this: “goal”@”scenario”.
The scenario is used to differentiate between multiple activities. This enables the user to easily compare and visualize how the design behaves with different activities defined in the sgdc constraint files.
In the SpyGlass setup file this looks like this:

```
[…]
current_goal pwr_estimation -scenario ps0
read_file -type sgdc constraints/top_rtl_power.sgdc
read_file -type sgdc constraints/top_rtl_power_activity_ps0.sgdc

current_goal pwr_estimation -scenario ps1
read_file -type sgdc constraints/top_rtl_power.sgdc
read_file -type sgdc constraints/top_rtl_power_activity_ps1.sgdc

current_goal pwr_estimation -scenario ps2
read_file -type sgdc constraints/top_rtl_power.sgdc
read_file -type sgdc constraints/top_rtl_power_activity_ps2.sgdc

[…]
```

Figure 5 SpyGlass Project File Example

The special constraints file top_rtl_power_activity_ps* only contains the pointer to the respective activity file (in fsdb or VCD format) and is the only difference between scenarios.
Meanwhile the generic setup for power is in top_rtl_power.sgdc and same for all.
Sgdc stands for “SpyGlass design constraints” and is a TCL syntax with tool proprietary commands.

So overall this configurable HTML report allows some built in tracking over time.
But this was not sufficient to our needs.

2.2 Extending and Customizing Tracking

The existing solution in the tool had the following limitations for us, which we wanted to resolve:

- Only vendor defined attributes can be tracked
- Only top level attributes can be tracked
- No possible reuse of tracking information in Excel

This was enabled and customized using two build-in tool features:

- Custom Report API, which allows generation of user defined reports
- HTML report configurations allows to incorporate additional data if provided in specific format

In addition a custom Perl script was created to export data also in Excel for further processing if requested/needed.

2.2.1 HTML Dashboard Configuration

A typical Dashboard generation setup may look like this:

```
CMDLINE:

spyglass -batch -config_file top_rtl_html_report_config \
   -gen_aggregate_report dashboard -reportdir my_spyglass_dashboard
```

```
top_rtl_html_report_config File Content:

top_rtl.prj
SUCCESS_CRITERIA_FILE_PATH top_rtl_html_report_dashboard_criteria
REPORT_TITLE_DASHBOARD top_rtl
REPORT_LABEL_DASHBOARD "${SPYGLASS_LABEL} ${SPYGLASS_DATE}"

REPORT_FILE_NAME_DASHBOARD dashboard
CUSTOM_LOGO_DASHBOARD ifx_logo.png@@@@@@IFX LOGO
```

```
top_rtl_html_report_dashboard_criteria File Content:

[…]
set_design_objective Power -criteria {Internal_power=display_only} -
goal {pwr_estimation}
[…]
```

Figure 6 Dashboard Generation Setup

To elaborate on this setup

- The command line contains a pointer to a generic report configuration file
- The generic report configuration file contains also some generic look and feel information and for which project file(s) to generate the report for, but also a pointer to a dashboard criteria file
- The dashboard criteria file hosts all further html configuration settings, mainly containing the information which attributes to report and if there is a limit for it. The example in Figure 6 in this document contains just one line for this criteria file stating that internal power is to be taken only from the goal "pwr_estimation" and that there is no pass/fail criteria

This setup can be enhanced to include additional custom attributes in the Dashboard. Dashboard generation picks up any data if provided in respective goal result report directory in specific format:

```
ifx_dashboard.rpt content:

SCHEMA@@MYATTRIBUTE1@@inst1_::_module1_::_Total_Power
VALUE@@MYVALUE1@@123e-02
```

```
Environment Variables:

INCLUDE_DASHBOARD_SOURCES := spyglass_reports/ifx_dashboard.rpt
```

Figure 7 Custom Dashboard Report Example

By setting the respective environment variable to the location of the report file, SpyGlass will add the information to the so called "RunSummary" files, which are the basis for the Dashboard report.

See [1] SpyGlass Documentation and search for key word "Customizing Report" for further information on syntax and setup of this format.

Now what is a "RunSummary" File?

```
Date Created     : 01-04-2015 22:43:21
SpyGlass Version : 5.5.0.3
Project Name     : top_rtl
Goal Name        : pwr_estimation
Scenario Name    : ps0_rtl
Top              : top

Message Summary
Severity   Non-Waived  Waived
```

```
FATAL              0          0
ERROR              0          0
WARNING          121          0
INFO               6          0


-dashboard_data total_power 2.341e-02
[…]
-dashboard_data inst1_::_module1_::_Total_Power 2.341e-02
[…]
```

Figure 8 RunSummary File Example

This is where SpyGlass stores all attributes that may be used for tracking and reporting in the HTML dashboard. The attributes in here map exactly to the attributes used in the html dashboard criteria file.
With a file like above, one now can add any custom attribute to the HTML dashboard:

```
top_rtl_html_report_dashboard_criteria File Content:
[…]
set_design_objective IFX_MODULEPOWER \
-criteria { inst1_::_module1_::_Total_Power=display_only} \
-goal { pwr_estimation}
[…]
```

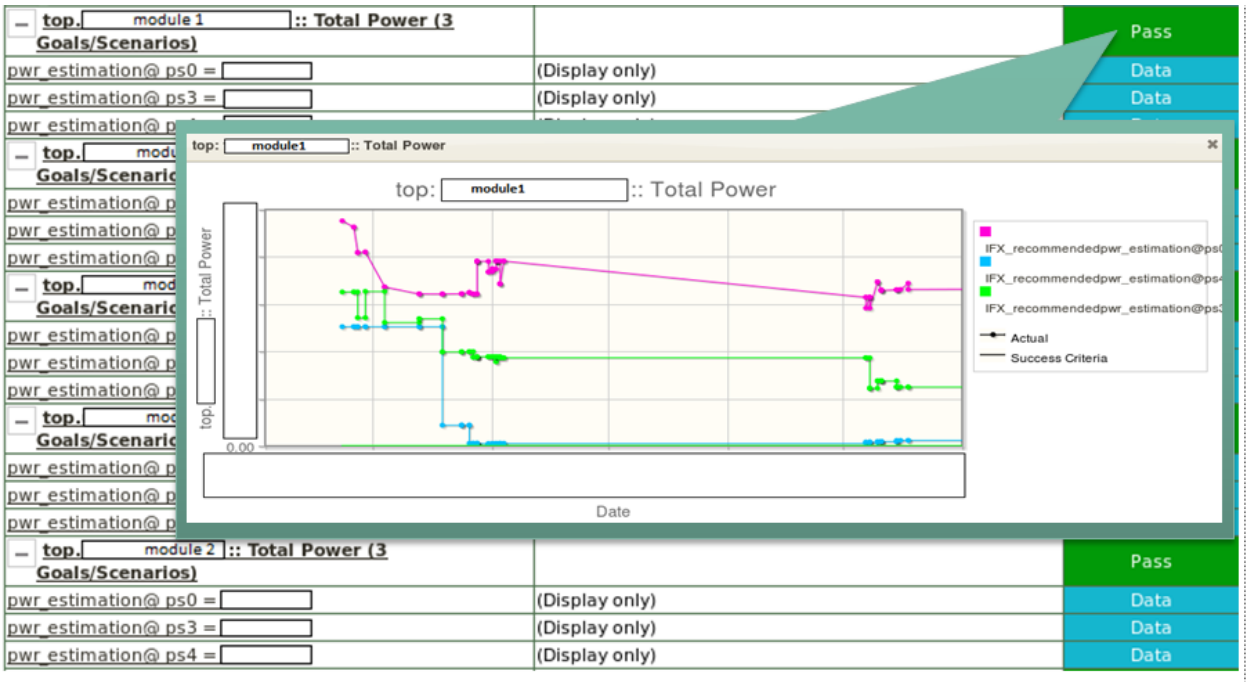

Figure 9 Customized HTML Dashboard

2.2.2 Custom Report Generation

How to generate the report and have its content added to the RunSummary information, which is required for the HTML generator? SpyGlass has a custom report API to generate own reports, and we use it to simply generate the required format.
This offers the advantage that independent of batch or GUI mode and without user interaction, all the important attributes of each run are stored each time it is executed.

```
SpyGlass project file:
[…]
set_option I { custom_reports }
[…]
current_goal pwr_estimation -scenario ps0
read_file -type sgdc constraints/top_rtl_power.sgdc
read_file -type sgdc constraints/top_rtl_power_activity_ps0.sgdc
set_goal_option report { pe_summary ifx_dashboard }
set_parameter ifx_power_modulelist {modulename1,modulename2}
[…]
```

```
custom_reports/ifx_dashboard-sgreport.pl:

use SpyGlass;
use SpyGlass::Objects;
use File::Basename qw(basename dirname);
use Cwd 'abs_path';
require "reports.pl";
my $report_name = "ifx_dashboard";
&spyRegisterReportGenerator($report_name,"", $report_name);
&spyRegisterReportParameter("","$report_name","ifx_power_modulelist","");
[…]
```

Figure 10 Custom Report and Generation Example

See [1] SpyGlass Documentation and search for key word “Creating Custom Reports” for further information on syntax and generation of custom reports.

2.2.3 Export to Excel

When talking about power, in many companies still a lot of discussions and experiments take place using Excel. So an easy way to export to that format would be beneficial.
Also there are so many ways how one can easily represent the data differently, that it would be a waste of time to wait for the EDA industry to re-implement every Excel feature in their EDA tools.

Only in a generic table calculation tool, one can visualize different activities and attributes in any way one can think of using standard pivot tables.

So last but not least a simple Perl script was generated that data mines all RunSummary data files and puts them in one Excel table like this.

	A	B	C	D	E	F	G	H	I	J	K
1	Date	Time	SpyGlass	Project	Goal	Scenario	Top	Module	Instance	Attribute	value
2	2015/03/19	23:00:50	5.5.0.3	top_rtl	pwr_estimation	ps0	top	module1	instname1	Clock_Gating_Efficiency(%)	42.986
3	2015/03/19	23:00:50	5.5.0.3	top_rtl	pwr_estimation	ps0	top	module2	instname2	Clock_Gating_Efficiency(%)	85.888
4	2015/03/19	23:00:50	5.5.0.3	top_rtl	pwr_estimation	ps0	top	module3	instname3	Clock_Gating_Efficiency(%)	48.629
5	...	...	...	...	...	...	...	...	...	...	...

Figure 11 Excel Format

This enables then graph generation with just a few buttons using Pivot tables, whatever correlation the end user wanted to see.
This can be the information of activities for top level over time like in Figure 12, but could also be the reporting of different design hierarchies for identical activity information like in Figure 13.
A common question is, which module was responsible for a change in overall power consumption. This now can be quickly answered.

One might ask: What is the benefit compared to simply dumping the csv file from the power explorer GUI? With that csv file one can also do any kind of number crunching there.
Let's compare the work flows:

CSV from power explorer workflow:

- Project decides to track power or certain attributes in the middle of the project
- Data prior to decision difficult to recreate
- SpyGlass Power user gets from this point onwards the action item for each RTL delivery to open GUI and dump csv at specifically location with date stamp
- Power Concept person (usually a different person than SpyGlass user) uses custom scripts and/or Excel macros to bring the data from each RTL delivery together and combine them in the format he prefers
    - Often interactive work just to adapt format
    - Often not easily interchangeable, as often custom macros and scripts are used instead of defined format
- Power Concept person generates custom graph of trend to management

Presented solution:

- Project decides to track power or certain attributes in the middle of the project
- Key attributes from earlier design states available in RunSummary files
- SpyGlass Power user has nothing to do to enable tracking, just starts excel exporting script, whenever Power Concept person wants to report something
- Power Concept person gets always standard format, only works with Excel to create nice trend graphs

So overall, the presented solution requires less interactive involvement and puts the post processing of the data as close as possible to the generation
Also as basic data is always available, this can even be provided post tape out just for a lessons learned and may result in tracking actively in next project.

On the other side for detailed analysis not all information is stored within the default tracked data, while all would be available in the full csv files from power explorer. But then again, detailed analysis usually requires just the status quo and not the history in that detail.

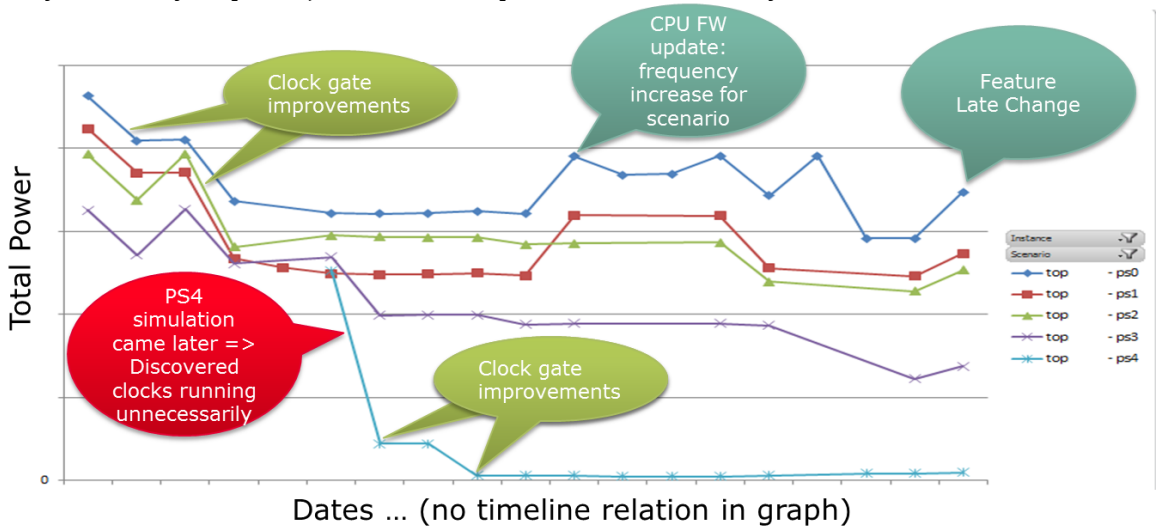

Figure 12 Design Optimization Graph over Time

In Figure 1 one can see that especially in the low power state modes of the new design, that were not part of the predecessor design, clock gating was missing in the early RTL version. Adding clock gating and the impact of clock gating is shown too. Parts of those clocks gating savings were achieved by adding/correcting instantiated central clock gates, other parts of savings were achieved by design changes which lead to further synthesis inferred clock gates or having them more often closed.
As this is a real design, power is not just going down, as multiple factors influence a design and its power consumption during design cycle
In the middle of the graph one can see for ps0 and ps1 a rise in power, which was simply due to the fact that the scenario was redefined to accommodate an updated specification and now ran with a higher frequency. Also at the end of the graph the overall power increased for all power modes due to late feature changes.

Another example is this graph here below in Figure 13.
In the dark blue line one can see the overall power going down in the beginning, while one module in light blue actually did increase in power. Only much later the project saw that this was not just due to added features and when fixing it for power at the end, it also had significant effect on overall power.
This graph helps to assess over time where the power increase or decrease is coming from and one can challenge oneself and ask the question if this was really expected as part of to the implemented feature change.

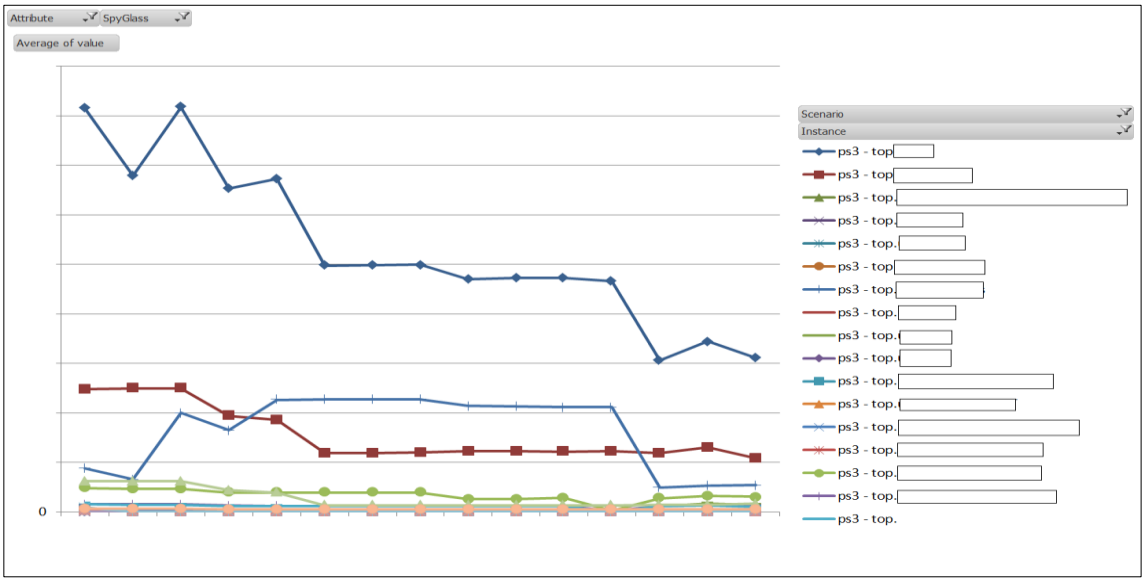

Figure 13 Activity ps3 per Instance Graph

3. Further Findings worth Mentioning

3.1 Clock Gating Efficiency Data Pitfalls

As mentioned earlier some power related design characteristics are basically more or at least equally important as absolute power numbers.
One of those is the clock gating efficiency.

Clock gating efficiency = % of time the clock gate is closed / % of overall simulation time

This information will be valid on RTL as well as final layout netlist, as this is per design and thus correlates per construction.
Still there are some pitfalls with those numbers. As this information does not contain the information if the source clock is active.
If the source clock is inactive, and all clock gates below are open, it still will result in zero clock activity, so everybody is happy.

Another relevant statistic is called ROAD:

```
Register Output Activity Density =
Register Q pin activity / Register CP pin activity)
```

A register that is always receiving a clock, but also has a lot of activity on its Q pin, has probably a good reason for not being gated, while a register whose Q pin never changes, makes you wonder why it is not being clock gated most of the time.

SpyGlass provides such statistics in various reports, though for following reason, sometimes it is misleading:

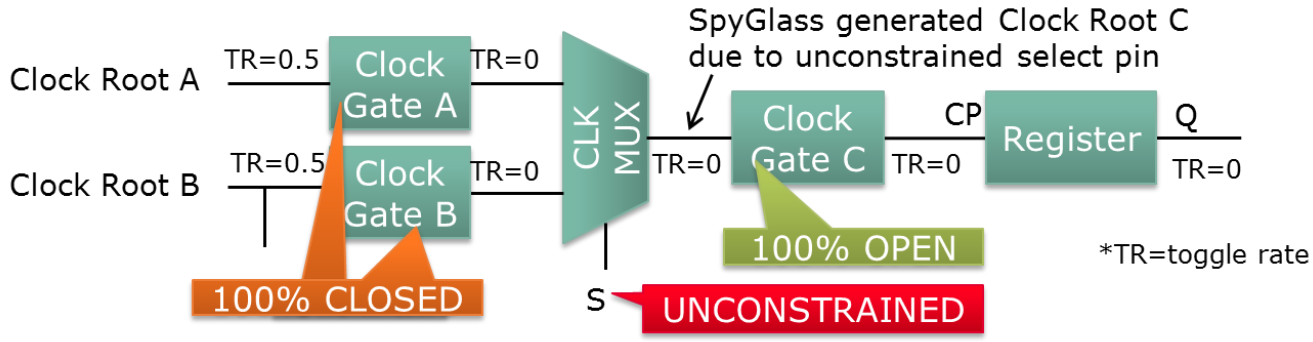

Figure 14 Clock Root Creation Example

SpyGlass automatically infers additional clock roots if it cannot identify a unique clock root. This is only done for creating clock gating statistics.
The reason for this tool behavior is being able to support certain beneficial tool features like e.g. reporting clock gating efficiency or generating a graph how many registers are reached by a specific clock over time.
For those features the tool infrastructure is currently designed in a way that each register may have just one clock root. If e.g. due to clock muxing that is not the case, the tool creates a new clock root object internally at the point of divergence to accommodate its own requirements.
In the above Figure 14, it will create a clock root on the output of the mux, due to an unconstraint select pin.
So for the Power feature, SpyGlass will always trace back from register clock pins until there is no

unique driver net and then creates a clock root to be used for further reports and statistics.

This may result in some bad statistics although actually there is no activity on the clock net at all:

$ClockGateEff_{ClockRoot\_C} = Time_{ClockGateClosed\_C}/Time_{overall} = 0\%$

$ROAD_{ClockRoot\_C} = 0_Q/0_{CP} = N/A = 0\%$ (reported as 0% ROAD in SpyGlass)

Enhancement request that at least this is reported as "N/A" instead of "0%" already filed.

3.2 Correlation

In general we find the VCD based correlation between SpyGlass RTL average power analysis and PrimetimePX gate level power analysis reasonable accurate. We have spent some effort in assessing best practice settings for our kind of designs, and also make use of some design history to tune to some extend the heuristics of SpyGlass via constraints. Even though on module level correlation is simply always more off, as SpyGlass Power Estimate is simply lacking a few things like e.g.:

- No timing driven synthesis engine
- No floorplan information
- Working with wireload models

Still in this specific design, we did get some surprises during PrimetimePX, even though we continuously tracked power on RTL.
Some modules were off by factor 10X due to dynamic power, impacting also the overall correlation and having us left with an optimistic RTL power estimate until we reached first layout power analysis.
We identified the actual activity information from RTL simulation as the root cause. Several modules did have 10X higher activity in the gate level simulation than in the RTL simulation due to race condition induced glitches in some large combinatoric clouds. As power estimation tools take the activity data as golden, power results could not possibly match with that delta in activity.
Further Root cause analysis of this issue and a method to create earlier realistic activity is the topic of another paper for SNUG Germany 2016 [2]

4. Conclusion

Power assessment on RTL is a valid and helpful approach to reach early convergence and keeping schedule if power targets exist for the design. Focus of such an assessment is usually dynamic power, as (other as leakage power) this is what the RTL designer has more control over. Accuracy/correlation is of course important to some extent, as it affects overall architecture decisions, but one important "trust building stone" to have is that a tool is accurate enough to indicate if something is worse or better for power.
Simply

- having a trend
- being able to ask the question, why the trend evolved in the one or other direction and if the behavior fits the expectation
- being able to assess clock gating, where independent of absolute power numbers, the actual gating efficiency always correlate and any reduction in clocked registers is usually beneficial
- being able to visualize and detect this

is very helpful.
This paper showed an application and customization of the default SpyGlass environment to help in visualization and tracking.

5. Acknowledgements

Thanks to
Richard Pierson (Infineon Technologies)
and
Guillaume Boillet (Synopsys)
for reviewing the initial presentation.

6. References

[1] SpyGlass HTML Documentation: $SPYGLASS_HOME/htmlhelp/index.html

[2] SNUG Germany 2016 paper: Generating accurate gate level activity for power analysis, prior to back annotated timing simulation.