ABSTRACT

BHATTACHARYA, SUPRIYA. Early IR-drop Analysis Flow for Hot-spot Pre-emption. (Under the direction of Dr William Rhett Davis).

Power Grid analysis, as a part of sign-off check, had been traditionally done after routing - at the end of design implementation flow due to availability of accurate parasitics etc only at that stage. This posed a threat to the product schedule because any power or voltage drop issue arising at this stage could potentially push-back the design to floorplanning or pre-CTS stage. Highly over-designed power-grids had been the approach to tackle these late surprises. However with exponential increase in gate count in modern SoCs routing resources became very expensive. The over-allocation of tracks to power and ground nets threaten the design closure with unrouteable signal nets. Moreover, below 28nm and with usage of finfets, congestion has become an even more severe issue. Overdesigned power-grids cannot be afforded anymore. Early power-grid analysis methodologies and tools are therefore being developed and employed to address this challenge.

Methodologies that predicts possible issues in design convergence early during the implementation phase can improve both quality-of-result as well as productivity of the design team. There are few early power analysis and power grid analysis tools and methodologies but they are either done at the end of synthesis or placement. The methods with dependency on post-synthesis results cannot be done on the RTL while a clean synthesis requires several weeks of time after RTL release (due to its dependency on clean design constraints – like sdc, etc).
RTL analysis tools like Power Artist can report area, power and connectivity information of the RTL. And tools like Red Hawk have early power grid capability. But these tools are very expensive. So this work shows an alternative approach of early power grid analysis using these backend tools. Design information like area, power, cell count etc are used from the original design to create inputs for the backend tool.

Apparently designs in the industry are becoming more and more (Intellectual Property) IP based and relatively little new RTLs are being written to develop the modern multi-core SoCs (like old wine in a new bottle!). This fact is being leveraged through the current work. Here, a methodology is implemented to provide ‘guidance’ to the design engineer about regions in the def frame that can have static IR-drop hot-spots so that appropriate steps can be taken very early – at pre-synthesis stage, of the design cycle. In this flow it is assumed that there is an incremental change in the original design. A method of automatically modifying the design through TCL commands was devised. Synopsys IC-Compiler was used for implementing this flow. Several TCL procs were used to create a proxy design to represent the design modified through script. ICC does not have such a capability of performing power-grid analysis at pre-placement stage. But, indirectly, this work made that possible. A dummy-cell placement flow is implemented to create combinational and sequential cells and place them over the core-area of the floorplan similar to previous design. This is followed by creating power/ground metal stripes and power-pads uniformly over the frame. Then clock tree synthesis, routing are done like a usual place-route flow to generate the IR-drop maps - using ICC’s basic capabilities. Experiments were done on multiple designs in order to show the correctness of this flow through IR-drop maps and dynamic power numbers of combinational, sequential and clock-
tree elements. The dynamic power numbers on clock tree are, relatively consistent, in range of 0.5X, register power 1.5X, and combinational power varying at an average of about 3X. The voltage drop maps on instances and power-ground stripes are found similar between proposed flow and reference flow. The results are encouraging.
Early IR drop Analysis Flow for Hot-spot Pre-emption

by
Suprio Bhattacharya

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the degree of
Master of Science

Electrical Engineering

Raleigh, North Carolina

2015

APPROVED BY:

________________________________  __________________________________
Dr Eric Rotenberg                    Dr Paul Franzon

________________________________
Dr William Rhett Davis
Chair of Advisory Committee
DEDICATION

To My Parents and Teachers!
BIOGRAPHY

Suprio Bhattacharya completed Bachelors in Engineering in Electronics and Telecommunication Engineering from University of Pune in 2005. He has worked as Design Engineer and Product Engineer in the semiconductor industry with companies like HCL Technologies, Freescale Semiconductors, Wipro Technologies, and Cadence Design Systems. His internship with Intel Corporation while pursuing Masters led to the inception of this work. His research interests include design automation, mathematical modeling, and neural engineering. He is an outdoor person and has strong interests in adventures and music.
ACKNOWLEDGMENTS

Foremost, I am extremely thankful to my advisor Dr Rhett Davis and my former manager at Intel Mr Tal Gubezskis. Without their support I simply could not attempt this work in any form.

I am very thankful to Dr Kinchit Desai for introducing me to this idea of early power analysis which formed the seed of this work. Thanks to Mr Santosh Thouta and Mr Arun Subbiah for their invaluable guidance during the numerous meetings we had during my internship at Intel.

I am very thankful to Dr Paul Franzon and Dr Eric Rotenberg for kindly agreeing to be on my committee, for the invaluable guidance at different steps and for taking out time from their super busy schedule to promptly responding to all my emails.

I am also thankful to Rangeen Roy Chowdhury (currently pursuing his PhD from NCSU) for providing me with the design collaterals using which I could perform some of the experiments, Chaitanya Mantena (my colleague at Intel) who helped me at all times in getting used to design flow, using the EDA tools amongst other things, and Wayne Szeto (my colleague at Intel) for sharing his invaluable insights.

I am also thankful to many other people - who have directly or indirectly, through their work (in diverse fields as arts, sports, and science), have (and continue to) inspire me through thick and thin by instilling enthusiasm, confidence, and courage.

Thank you all for being there!
# TABLE OF CONTENTS

LIST OF TABLES .................................................................................................................. vi

LIST OF FIGURES ................................................................................................................ vii

1. Introduction ....................................................................................................................... 1

2. The Power Calculation paradigm ..................................................................................... 12

3. Power Grid Design .......................................................................................................... 19

4. Early IR drop analysis flow: A Use Case ........................................................................ 29

5. Experiments and Results ................................................................................................. 45

6. Summary ............................................................................................................................ 70

7. Current Limitations and Futurescope ............................................................................. 73

REFERENCES ...................................................................................................................... 75

APPENDIX .............................................................................................................................. 77

APPENDIX A .......................................................................................................................... 78
LIST OF TABLES

Table 1: Correlation between the instance powers for design Cortex. .................................. 43
Table 2: Correlation between the instance powers for design Cortex. ................................. 51
Table 3: Correlation between the instance powers for design Jacobi................................. 54
Table 4: Correlation between the instance powers for design S38584. .............................. 57
Table 5: Correlation between the instance powers for design S38417. ............................... 60
Table 6: Correlation between the instance powers for design S13207. ......................... 63
Table 7: Correlation between the instance powers for design S35932. ............................... 66
Table 8: Correlation between the instance powers for design S15850. .............................. 69
Table 9: Summary of comparison of power numbers over multiple designs for validation... 71
LIST OF FIGURES

Figure 1: Traditional early power grid analysis flow where it is done at post placement stage. .......................................................................................................................... 2

Figure 2: RTL based early power analysis flow using special tools. ......................................................... 3

Figure 3: Illustration of license crunch of expensive tools........................................................................ 4

Figure 4: Proposed Early IR Drop Analysis Flow ....................................................................................... 5

Figure 5: Illustration of information derived from RTL for EIRDA flow.................................................. 6

Figure 6: Illustration of variation in area between RTL and post route design stage. ......................... 7

Figure 7: Illustration of instance power map of a post-placed design....................................................... 8

Figure 8: RTL based early IR drop analysis flow using place-and-route tools. ................................. 9

Figure 9: Power calculation requirements [picture sources 3, 8] ............................................................. 12

Figure 10: Power components of a buffer. [2] ......................................................................................... 13

Figure 11: Power associated to different types of cells, as calculated on Fabscalar design (4.47mW) ...................................................................................................................... 16

Figure 12: Basic Power Grid structure [8] ............................................................................................... 19

Figure 13: Calculation model for hierarchical power grid analysis [11] .............................................. 25

Figure 14: Illustration of the inputs required to run the EIRDA flow......................................................... 29

Figure 15: Stages of EIRDA Flow ............................................................................................................ 31

Figure 16: Contents of reg_distribution.rpt and comb_distribution.rpt files, generated using report_cell_distribution tcl command on the original design session on ICC shell. .......... 32

Figure 17: Contents of loc_for_new_reg.rpt, generated using add_extra_logic command on ICC after selecting a logic portion on the GUI. ................................................................................. 33
Figure 18: The selected/highlighted cells are assumed to be logic to which new cells will be added.

Figure 19: Code snippet to create power grid. Please see appendix for details.

Figure 20: Regions created for hierarchical modules created in ICC for proxy design.

Figure 21: Placing the cells in hierarchical modules. Number of cells in each grid is determined based on reg_distribution.rpt and comb_distribution.rpt and loc_for_new_reg.rpt.

Figure 22: Connecting the different inputs of the combinational and sequential cells with the module’s input ports. The output ports are created to connect output of each register.

Figure 23: Connecting the input ports of all the hierarchical modules with the corresponding primary input ports.

Figure 24: Placement of combinational and sequential cells and their connectivity as seen at the end of routing.

Figure 25: Dividing the floorplan into multiple grids to measure register and combinational cell distribution per grid.

Figure 26: ICC GUI showing voltage drop map.

Figure 27: ICC GUI showing instance voltage map.

Figure 28: Flows run to generate the validation results.

Figure 29: Illustration indicating addition of new register-xor gate pair to an existing register.

Figure 30: Interpreting the IR drop maps and power reports.

Figure 31: ICC GUI showing instance voltage map using EIRDA flow.
Figure 32: ICC GUI showing instance voltage map using reference flow.......................... 49
Figure 33: Voltage drop on power straps on EIRDA flow. ........................................ 50
Figure 34: ICC GUI showing voltage map using reference flow.................................. 50
Figure 35: ICC GUI showing instance voltage map using EIRDA flow. ................. 52
Figure 36: ICC GUI showing instance voltage map using reference flow............... 52
Figure 37: ICC GUI showing voltage map using EIRDA flow. ................................. 53
Figure 38: ICC GUI showing voltage map using reference flow.......................... 53
Figure 39: ICC GUI showing instance voltage map using EIRDA flow. ................. 55
Figure 40: ICC GUI showing instance voltage map using reference flow............... 55
Figure 41: ICC GUI showing voltage map using EIRDA flow. ................................. 56
Figure 42: ICC GUI showing voltage map using reference flow.......................... 56
Figure 43: ICC GUI showing instance voltage map using EIRDA flow. ................. 58
Figure 44: ICC GUI showing instance voltage map using reference flow............... 58
Figure 45: ICC GUI showing voltage map using EIRDA flow. ................................. 59
Figure 46: ICC GUI showing voltage map using reference flow.......................... 59
Figure 47: ICC GUI showing instance voltage map using EIRDA flow. ................. 61
Figure 48: ICC GUI showing instance voltage map using reference flow............... 61
Figure 49: ICC GUI showing voltage map using EIRDA flow. ................................. 62
Figure 50: ICC GUI showing voltage map using reference flow.......................... 62
Figure 51: ICC GUI showing instance voltage map using EIRDA flow. ................. 64
Figure 52: ICC GUI showing instance voltage map using reference flow............... 64
Figure 53: ICC GUI showing voltage map using EIRDA flow. ................................. 65
Figure 54: ICC GUI showing voltage map using reference flow.............................. 65

Figure 55: ICC GUI showing instance voltage map using EIRDA flow..................... 67

Figure 56: ICC GUI showing instance voltage map using reference flow.................. 67

Figure 57: ICC GUI showing voltage map using EIRDA flow............................... 68

Figure 58: ICC GUI showing voltage map using reference flow............................ 68
1. Introduction

The semiconductor technology has been the primary enabler of growth of almost every other industry in modern world. Exponential growth in complexity of the modern SoCs (System on Chip) in terms of device count, functionality, and specifications have fueled IP (Intellectual Property) based designs. In fact to the three major pillars of the semiconductor industry – design houses, fabrication houses, and EDA (Electronic Design Automation) companies, companies providing IPs to the design houses form the forth new pillar. Two primary factors are the reasons for this development: i) using silicon-proven IP increases chances of “first-time-success” – critical for the profit margin of the design houses due to super-expensive, newer process nodes, and ii) “First-to-market” – a critical business strategy in today’s extremely competitive world, reusing proven IPs help reduce product development cycle time. Thus it may be argued that most of the new SoCs – particularly that go into consumer electronics, have relatively little amount of ‘new’ RTL in it. Opportunities thus exist in leveraging this aspect of SoC design – like opportunity of performing early power grid analysis.

1.1 Need of early power grid analysis

Power Grid analysis has been traditionally done at the end of design cycle during sign-off and therefore pose a threat to the product schedule for any issue discovered during the sign-off checks. They can result in a re-spin of the design to floorplanning or pre-CTS stage. Moreover below 28nm and usage of devices like finfets, congestion has become a major issue. Traditional approach of overdesigning a power-grid no longer works. Early power-grid analysis methodologies and tools are therefore being developed to address this. This can also help in early chip-package co-design.
1.2. Existing solutions

Initial approaches of performing early power grid analysis were done using the usual back-end physical implementation tools like Cadence’s Encounter Digital Implementation System, Mentor Graphics’ Olympus, or Synopsys IC Compiler. These could perform the IR drop calculation once the design was synthesized and placed and routed. It takes several weeks for a design to reach the placement stage. There are multiple bottlenecks associated in terms timing constraints, DFT, etc. – that get defined during the initial months. Thus the early power grid analysis has to wait for a few months until the design teams reach post-placement stage.

Newer approach of performing early power grid analysis is to estimate the RTL power and then perform power grid analysis assuming the power – thus estimated, is consumed uniformly across the core-area. Tools like Power Artist™ are used for estimating the RTL power while tools like Red-Hawk™, has this capability of performing early power grid analysis. Besides,
traditional synthesis tools like Synopsys Design Compiler and Cadence RTL-Compiler have RTL power estimation capabilities that can be used for early power analysis.

**Figure 2: RTL based early power analysis flow using special tools.**

Companies have only a few licenses of these tools. And with multiple designs being implemented by multiple teams in parallel, managing license is a challenge. On the other hand, there are relatively larger number of licenses of the physical implementation tool (though it is not they come very cheap!). These tools too have capability of performing basic power grid analysis. Thus a flow would be very useful that can utilize the basic capability of these backend tools to perform early power grid analysis. Any flow that can make them work beyond their normal capabilities can be very valuable both for design team as they probably do not have to use another tool (and its added methodologies) as well as for management for not having to budget for another expensive tool licenses.
The primary goal of this work/flow is to predict the areas of higher static IR drop based on the RTL at the initial phase of design. The assumption is that this design for implementation is a modified version. This is a realistic assumption because designs in the industry are becoming more and more (Intellectual Property) IP based and very little new RTLs are actually written to develop the modern multi-core SoCs in most cases (like old wine in a new bottle!). Majority of today’s billion transistors based designs are not built from scratch but rather have incremental changes.

**1.3 Proposed method**

The primary goal of this work/flow is to predict the areas of higher static IR drop based on the RTL at the initial phase of design. The assumption is that this design for implementation is a modified version. This is a realistic assumption because designs in the industry are becoming more and more (Intellectual Property) IP based and very little new RTLs are actually written to develop the modern multi-core SoCs in most cases (like old wine in a new bottle!). Majority of today’s billion transistors based designs are not built from scratch but rather have incremental changes.

*Figure 3: Illustration of license crunch of expensive tools.*
These changes may be due to some logic enhancement or inclusion of a new logic. Though with technology scaling the final implementation does change but it can still be assumed that the changes at block level are incremental in nature.

The work done as presented through this document, is to leverage these information. It can enable power grid design right during the initial phases after RTL design. The value proposition is in using more information from the RTL and previous design database. The allied innovation are in using a PnR tool – Synopsys IC Compiler (ICC) to perform this early power grid analysis. ICC requires a design to be at least placed to perform early power grid analysis. This work shows an alternative flow to use ICC to perform an early power grid analysis which is otherwise beyond its capability. The idea can easily be extended to other PnR tools like Cadence’s Encounter as well.
Information derived from RTL:

```
RTL can be analyzed by different purpose by different tools. The basic information that needs to be derived in most cases is the connectivity between the logic elements. For example RTL power can be estimated using power analysis tool like Power Artist, or even by synthesis tools like RTL Compiler. RTL can be analyzed to assess the number of sequential cells, combinational cells and any other hard macro that may be present. Accordingly a reliable estimate of the design area can also be inferred. Albeit the inaccuracy of this estimation is limited as the actual area would be different as the design gets optimized during synthesis and place and route in order to meet the design specifications for timing, power, and area. However, by correlation the RTL area numbers with post layout area numbers, a reliable “fudge-factor” can be established.
```

**Figure 5: Illustration of information derived from RTL for EIRDA flow.**
Following figure illustrates this idea:

![Illustration about variation in area between RTL and post route design stage](image)

*Figure 6: Illustration of variation in area between RTL and post route design stage.*

For any design, the number of sequential elements tends to remain same between RTL and post layout stage. It is primarily the combinational cells that change and are restructured to meet the design specifications. Albeit the relative percentage of combinational and sequential cells is design dependent (the illustration – figure 14, just shows a generic case where sequential cells’ count is more). The clock tree that does not exist in RTL, thus, has some area only after clock tree synthesis.

Similarly, the power numbers can be analyzed on the RTL and correlated with post layout design. Moreover, for a given set of constraints for a design, a correlation can be drawn between power and cell count. Different modules in the design with different cell counts have
specific power. This power distribution can be seen as a power map with each region/module region consuming certain amount of power – illustrated as follows:

Tools like Red-Hawk have the early power grid analysis capability in which each region of core-area consuming certain amount of power is treated as a cluster of current sinks. Power grid analysis is then to check whether the grid is strong enough to support current without causing enough voltage drop. However these tools have very expensive license costs.

In order to use the backend tools, the necessary information have to be extracted from the RTL to form inputs in correct format so that the back-tools can process them appropriately.
All the information derived from the RTL – cell count, area, average fanout of the cells etc. are the key information that can be fed to the back-end tool. The other critical information needed is the placement information of the modules or the cell distribution from the previous design database so that cells can be placed realistically in the die-area and current can sink appropriately from the power grid through them.

Based on the kind of IP and its previous implementation data, the standard cell distribution report can be prepared. This distribution is analyzed by dividing the core area into smaller grids and number of sequential and combinational cells in each grid can reported. Considering the new RTL design (modified by designer) would be similar to the earlier implementation, this information can be used to place dummy cells – combinational and sequential cells in a similar fashion. This proxy design is then taken through the place-and-route flow to generate the IR drop maps. Details about this flow is discussed in Chapter 4.

Figure 8: RTL based early IR drop analysis flow using place-and-route tools.
Difference with respect to implementing Engineering Change Order (ECO) flows

ECOs are the quickest way of implementing incremental changes in the design towards the end of the design phase, when final versions of the RTLs get released and the backend team tries incorporating those changes through automatic ECO implementation using the Back-End tools. However this work has a different value proposition than implementing an automatic ECO flow. Firstly, this is based on statistical approach for power-grid analysis without exact need of a cell structure formally equivalent to RTL but ECOs are done for an exact result. Secondly, this approach can be extended to perform power grid analysis right at initial stage of the design phase, even for newer designs with minimal information – i.e. design constraints like sdc etc, whilst ECOs are done at the end of the design phase when all the constraints are available and accurate - logically equivalent, timing closed design implementation is needed.

Following chapters elaborate on the background of types of power consumption in ICs – the physical background, their models in standard cell libraries –Synopsys liberty or .lib files. Next basics of power grid design, types of analysis that are done on them like static dynamic power grid analysis, and the types of algorithms used in calculating voltage-drop/power calculation in the power-grids. In chapter 4, details about the different steps of dummy-cell placement (EIRDA) flow are discussed. Chapter 5 shows the experimental results on the different design done through the EIRDA flow through the instance voltage-drop map, power-rail, voltage-drop map. Maps generated using EIRDA flow and using full placement-and-route flow (of corresponding designs) are showed that show correlation. Also the dynamic power (segregated into clock tree power, combinational cell power, and register/sequential power) numbers
between the two flows are compared, that in general tend to correlate. Finally, limitations of the current flow and future possibilities of extending the idea into an EDA tool is discussed.
2. The Power Calculation paradigm

In this section we will go through the different aspects of power calculation starting with the classification of power into static, dynamic power and how they are modeled in the liberty files and used by the EDA tools.

Requirements for power calculation

![Diagram](source.png)

*Figure 9: Power calculation requirements [picture sources 3, 8]*

Above figure shows inputs needed for power analysis. The power models come from the technology library. The netlist is usually taken from the post-route stage. The signal activities
come from the SAIF (Switching Activity Interchange Format) files whilst net-parasitics are read from the spec.

In older (90nm and above) NLDM (Non-Linear Delay Model) libraries used a look-up table based approach to calculate the power. However with increasing short-channel effects, many assumptions – uniform transition waveform, single value of pin capacitances, etc. no longer hold good. CCS (composite current source) based cell libraries are thus being used for their high accuracy and time resolution [2]. The leakage power is calculated based on the different states of the input pins. The dynamic power is calculated based on the current waveforms – using CCS libraries the current waveforms are estimated very accurately resulting in a more accurate power calculation.

Components of Power

Following figure shows the different current components associated to the total power (dynamic + leakage) calculation.

\[ \text{Figure 10: Power components of a buffer. [2]} \]
Power is normally classified into static and dynamic power [2]:

1. Static/leakage Power: It is the power consumed by the cells when there is no switching at the inputs and only current flow is due to their internal parasitics. The basic equation for the leakage power is $V_{DD} \cdot I_{leak}$. Depending on the flow of current between different regions, the total static power can be classified into different components [6, ppt]:
   a. Subthreshold conduction: The conduction by the device at weak-inversion region when the gate-to-source voltage is less than threshold voltage.
   b. Gate induced drain leakage: is caused due to the band-to-band tunneling between the silicon and gate dielectric when the gate-to-drain bias is large [9].
   c. Drain source punch-through: this short-channel effect happens when the drain is at a much higher potential than the source and the drain current becomes independent of the gate voltage as the depletion region near the drain reaches the source [9].
   d. Gate tunneling: As the dielectric thickness continued reducing, albeit thanks to high-k dielectric that reduced the rate, leakage current due to tunneling through the thin dielectric, continued increasing. This is a significant source of leakage power.

As evident from above list, most of the components are associated to Gate. Thus leakage power varies greatly with number of inputs of a cell.

Each cell can be associated to a particular state depending on the stable potential at its different nodes. Thus number of states in a combinational cell depends only the input
nodes while power in sequential cells have also dependence on potentials at their internal nodes.

2. Dynamic Power:

The dynamic power consumption in the design is categorized into two types based on the external or internal connection of the ‘load’, and can be expressed as $P_{\text{switching}} + P_{\text{internal}}$:

a. Internal power ($P_{\text{internal}}$): It is the power consumed during switching – when both N and PMOSes are on and cause a short-circuit current, and due to charging/recharging of the internal nodes capacitances. Thus larger cmos cell with more nodes and pmos-nmos pair have a greater internal power.

   It is expressed as $-\frac{1}{2} f C_{\text{int}} V^2 + V.I_{\text{sc}}$, (where symbols have their usual meanings)

b. Switching power ($P_{\text{switching}}$): Expressed as $\frac{1}{2} f C_{\text{load}} V^2$, and also referred as short circuit power, is the power loss when both the pmos and nmos are on connecting power and ground.
Power distribution across cell types

**Figure 11**: Power associated to different types of cells, as calculated on Fabscalar design (4.47mW).

Above figure shows how power distribution, in terms of magnitude, is mostly associated to the combinational logic.

With the shrinking technology nodes, the nature of parasitics in the design have also been changing – reduced dimensions are causing reduced capacitance and increased resistances. As a result the assumptions about the parasitic nature of the routes are also changing – earlier they could be treated just as parasitics components but not anymore. The earlier NLDM (Non Linear Delay Model) fails to account for this and CCS (Constant Current Source) based libraries are employed to accurately characterize the delays and power numbers.

**RTL power analysis:**

There are several EDA tools that are used to perform early power analysis on the RTL. For example Apache’s Power Artist. Using them have the advantage of having an early estimate
on the power numbers. By estimating the power numbers, design engineers can try means of accordingly control the power of different modules within their power-budget by applying different techniques.

The basic flow for the RTL power calculation involves elaborating the RTL similar to synthesis tools to infer the sequential and combinational elements – like a ‘rough’ logic synthesis. Other information like clock gating and clock pins, clock period, transition time etc. are specified so that the timing graphs can be created. Then using the power models from libraries like CCS liberty files, power calculations can be done to estimate the total design power.

However, as the design thus represented is not an accurate representation of the final post layout design, the power numbers are also inaccurate. The design teams perform extensive experiments to come up with the correct setting for these tools in order to bring a closer correlation (in the range of +-20%) between the predicted powers of the RTL power analysis tools v/s sign-off power numbers.

While such RTL power estimation methods are useful, they do not provide any idea about the physical location on the floorplan where hot-spots might occur during the power grid analysis because of uniform density. Using some spatial information about the cell distribution, however, can add more value to this approach.

**Backend power calculation:**

After the post-route stage, accurate power analysis and power integrity checks can be done on the design. The power analysis is done in a vectored (using SAIF) or vectorless form, each method having its own pros and cons. While SAIF based approach gives a more realistic picture of the power consumption, they become available at later stage of the design phase when the verification engineers have done significant work of bug finding and RTL fixing.
Normally this happens after few weeks of the RTL release. Also, from reliability perspective, in which the worst case power analysis is done, it is very hard, if not impossible, to create the realistic vectors associated to the maximum power consuming state of the design. Particularly with super complex designs of today – multi-voltage, power-gating, multi-core, (beyond the earlier days of just multi-corner and multi-mode) added with more complex devices like FinFETs at latest technology nodes, identifying conditions for the maximum power consumption of the design gets even more difficult. Vectorless power analysis comes to rescue in this situation by providing maximum power numbers that can ever be possible on the design. It is done by applying certain switching factors on the nodes in the design – inputs, registers, etc. and then calculating the power, voltage drop, electromigration etc. Although even this is pessimistic because the design may not actually be running such modes, it is the best worst case calculation. Another advantage of the vectorless approach is that it can be done at any stage of the design phase as it does not have a dependency on other teams – the way SAIF has on verification team. Next chapter has more details on the overall aspect about power grids and power calculation.
3. Power Grid Design

![Diagram of a power grid structure]

*Figure 12: Basic Power Grid structure [8]*

**Power Grid design:**

Traditionally a power grid has been a highly overdesigned structure and is “carried forward” over generations of ICs simply because they have not failed in past. A new designer dare not question the extent of overdesign for he/she may have to bear the responsibility of failure of the chip in future. Apparently fault-finding on chip is a terribly difficult fault-testing task in itself, and so is checking root-cause to be a power-grid failure.

**Power grid design consideration**

Power grid essentially comprises of a power/ground-ring, power/ground straps, vias, and power nets to supply power and connection to each devices.

*set_fp_rail_constraints -add_layer -layer metal9 -min Strap 8 -max Strap 8*
However with increasing number of IOs in the chips, flip-chip based packages have become very common. These packages have the pads connected to the package through C4 bumps distributed across the surface (shown in previous figure).

A power grid has the following features that determine the effectiveness of the power distribution:

- Metal layer: The top-metal layers are most suitable to form the power ring and the power due to their larger dimensions that reduces voltage drop.

- Width and spacing: depending on the electromigration limits, power, and current requirements the number of power and ground stripes is determined. But the other critical factor to consider is the signal congestion. With total number of routing tracks being constant, there has to be a compromise between power routes and signal routes. So although it would be nice to have maximum number of power routes, design congestion alleviation may need to create more space between power routes for more signal tracks. On the other hand, IR drop issues discovered at the post-route stage may lead to removal of signal tracks to create space for more power tracks.

- Types of vias: Vias are dropped from the top-most metal layers to layers underneath to supply the PG pins of standard cells.

- Power Pads
Power pads are associated to the package but in order to perform realistic analysis, the backend tools like ICC provide feature of adding virtual pads. The types and number of pads play a significant roles in the power distribution and integrity of the chip.

- UPF and other complexities:
  The other added complexities in the analysis are accommodating the constraints set by the low power standards like UPF. Moreover the dynamic behavior of the design, which is otherwise beyond the scope of current work, also plays a key role in causing a power need at certain location at certain time – thereby adding a whole new dimension to this problem and its solution space. In this work, these extra constraints are not handled.

**Power grid analysis types:**

Static and dynamic are two types of analysis done for power grid integrity check. Static analysis checks on degree of current flow through the routes and associated voltage drop, power consumption, and reliability checks like electromigration. Dynamic analysis is done to check the transient/temporal behavior of routes due to the parasitics – inductances, capacitances and resistances associated to power distribution network (PDN). Another perspective to look at these two types of power grid analysis is in terms of performance and reliability:

**Power grid analysis for performance:** The proper operation of the devices require them to be in their proper state of operation – linear, saturation etc. However, as the current flows from
the pads to the devices via the metal routes and via stacks, there is a voltage drop. This drop must be within certain limit so that the cmos devices are in their correct state of operation. The situation becomes complicated as the switching behavior of the devices change with time depending on the functionality. Different functional modes activate different modules of the design – that are spread over different parts on the frame. Consequently the voltage drops also change. Some regions may now have more voltage drop thereby slowing down the cells while some may have lesser drop thereby fastening the cells – either can cause timing violations if the variations exceed designed limits. Moreover, parasitics in the routes greatly effect these changes causing voltage droops.

Modern designs with multiple voltage and power domains add another degree of complexity. As different regions of the design can now turn on/off, the sudden surge in current requirement becomes more uncertainty and at higher magnitude. If the power grid is not designed to accommodate such needs, there may be too many functional failures.

**Power grid analysis for reliability:** Increasing device density is associated more current per unit area that leads to more thermal heating per unit area leading to temperature hot-spots. Moreover, with reducing device sizes, current density in the wires are causing them to operate at their electromigration limits. These factors affect the reliability of the power grid/power delivery network leading to early device/system failure. Let us take a closer look on some of these critical factors:

- **Electromigration:** shifting of ions in the metallic routes over time due to collision with electron clouds can cause cracks and electrical discontinuities in them. At corners of the routes, where the electron density is higher, electromigration is more prone to occur.
- Oxide breakdown: Gate Oxide breakdown (BD) – diffusion – thermal stress – greater electric field strength at advanced nodes – flow of hydrogen through the oxide that actually causes the BD – less regulated power supply makes oxide DB more likely.

- Hot carrier injection: The electric field strength gets stronger with shrinking device sizes causing impact-ionization and carrier multiplication. The gate-control can thus get lost resulting in a very high leakage power consumption [9].

Following is a little elaboration on these two types of power rail analysis:

- Static Analysis: As the name suggests, timing behavior of the power grid or PDN are not considered but the current flow through the different nodes of the network are analyzed in this method. All the PG pins of all the standard cells in the design and network parasitics are considered to calculate the resistance of the entire network, and based on the power models of the cells from the libraries, the leakage and dynamic (internal and switching) powers are calculated. The switching probabilities of each cell and applied switching probability of the inputs and sequential cells in the design are used to calculate the power and IR drop for the design. Similar to static-timing-analysis (STA), switching factors are propagated through cells. And this is done at different operating corner and functional working modes of the design. In the current work, static power analysis is done. Based on availability of SAIF/VCD files, the switching factors can derived from the functional waveforms to analyze function specific power numbers. In this work, vectorless approach of signal activity (or toggle count) is approached.

- Dynamic analysis: In this method, the PDN is modelled as a network of impedances and a time-domain analysis is done. Dynamic analysis is not in the scope of this work.
Estimation models for IR drop analysis in power grids

A resistive model is used for the power-ground interconnect network to perform static IR drop analysis (whilst a RLC model is used to perform dynamic IR drop analysis). Each unit cell on the frame is represented by a power-ground wire grid of impedance and algorithms are employed to calculate the effective impedance between different points. All such models are then combined together based on the network topology to be analyzed together using circuit simulation algorithms/tools [11].

Considering the billions of transistors in today’s designs and many associated with multiple power ground pins, further complicated by the multi-corner/multi-mode analysis, the sheer volume of data handled for power grid analysis is daunting. Practically, besides requiring extreme accuracy of result, runtime and memory requirements of the algorithms employed can make or break the power-grid analysis EDA tool’s market prospect! Therefore it will be interesting to take a quick look on the different approaches being used to solve this huge computational problem. Hierarchical methods are generally becoming commonplace in tackling in the super-sized database of nodes, pins, and other design information needed to perform the power grid analysis These help in managing the peak-memory usage so that machines running the analysis does not crash.

The following three steps are primarily needed to solve this problem [11]:

- Divide and conquer: the entire floorplan is divided into local grids called macromodelling [11]. These can be based on number of ports in the design wherein each local grid can also be associated to all the internal interconnect nodes of pins and nets.
Each local partition was modeled through $I = A.V + S$ form, wherein $A$ and $S$ are constant (usually dense) matrix and constant vector, and $I$ and $V$ are the port current and voltage vectors.

- Solving the global grid: all the local grids are then considered together to form a global equation. This form is simplified by removing all internal nodes and sparsifying the matrices to limit the error.

- Local grid computation: Finally the solutions obtained from global grid computations are then used to compute the internal node voltages and currents.

The number of grids and size – number of nodes in each grid, have significant role in speed of computation. Grid reduction techniques can be employed to make the above steps faster and more efficient.
Challenges of early power prediction

The main challenge in power prediction is due to the fact that the initial graph – the common data structure used to represent design, derived from the RTL is quite different from the final graph of post-routed design. Due to optimizations happening at different stages to meet the PPA (power performance and area) requirements there is a significant change in the design. Moreover extra logic insertion for different requirements of testability (like BIST etc) or power-cells further transforms the original design by huge proportions. Thus it is important to understand these key factors around which design transforms so that a better power prediction method can be devised. Following section takes a closer look in these steps and how they modify the design.

Following sections will detail on the kind of changes takes place at each of these stages.

a) Changes during synthesis:

There broadly two major kinds of transformation during synthesis associated to logic optimization and optimization to meet design constraints. Logic optimization may be thought of ironing out the inherent redundancies in the RTL and comprises of constant propagation dead-logic removal, etc. The next level of optimization takes place to meet the PPA constraints and comprises of the following:

- Grouping/ungrouping – sometimes multiple submodules in their parent module may have to be ‘dissolved’ to allow better logic optimization by the tool. This also effects the placement of this logic eventually during the place-and-route flow because they may be moved away from the surrounding logic depending on their timing slacks with other logic groups.
- Clock gating – Clock gating reduces the dynamic power consumption by about 30%. Much of the clock gating is done during the physical implementation phase being automatically inserted by the tools. It is therefore pertinent to account for the clock-gates quite accurately during the RTL power analysis.

- Scan insertion – Inserting BIST for memories (MBIST) and logic (LBIST) can add to the design significantly. Thus it may be necessary to account for these insertion during power analysis.

b) Changes during Placement and clock tree synthesis optimization

During placement, the net delays become more accurate than been approximated in synthesis (although the physical synthesis tools have reduced these mismatches considerably with challenges of miscorrelation lying mostly with the long nets). Also depending on the congestion, the cell placement may change considerably depending on the modules and their interconnectivity with other modules. All these factors lead to another degree of change of the design that can spoil the predicted placement of the modules. Further during clock tree synthesis, with tighter skew and latency constraints, the design increases significantly – like by 10% or more, during clock tree buffer insertion. And these clock tree buffers, always toggling at, possibly the highest, clock frequency, contribute to significant portion of the dynamic power. Following are some of the factors that significantly transforms the design:

- Restructuring of logic structure to optimize PPA.
- Cloning/decloning of clock gates and registers
- Multi-Bit Registers
c) Changes during routing optimization:
- Long net buffering to minimize signal integrity issues

However, as discussed in introduction, IP reuse is opening up opportunities of leveraging already existing information about the blocks of the SoCs that can pave the way for more reliable early power grid analysis methodologies, as has been attempted in this work.
4. Early IR drop analysis flow: A Use Case

In this chapter, a proxy design will be created and taken through the place-and-route flow to perform early IR drop analysis. As discussed in previous chapters, the primary contribution of this work is in performing early IR drop estimation on the grid even before synthesis has been performed. As soon as the RTL is available, this flow can begin. The only caveat is that this design should be an incremental version such that most of the logic can be assumed to remain same. Access to placed and routed database of the original design is also assumed to be available.

Figure 14: Illustration of the inputs required to run the EIRDA flow.

Synopsys IC Compiler (ICC) was used for the implementation of the flow for this work. The different designs used as testcases were used either in their RTL form or in their mapped netlist.
form. In order to run the flow, first step is to prepare the necessary inputs needed to run the flow. These include the following:

- Milkyway library: all the necessary views – CELL, FRAM etc

- Previous Design Database: the cell distribution from the previous design database (at its post place stage) are needed. This information is collected in two files: reg_distribution.rpt and comb_distribution.rpt. They have TCL arrays reg_dist and comb_dist defined. Their size depend on the number grids into which the core-area has been divided. new_reg_loc.rpt is the third file needed in the flow which has the location of registers in previous design to which new logic has been added. The generation of these three files will be detailed in following sections.

- run_icc.tcl is the top level run file for ICC which is sourced as a single file to run the flow. It defines all the different global variables used in different stages of the flow.

After preparing all the necessary inputs needed to run the flow, run_icc.tcl is sourced in ICC. Following figure illustrates the different stages of the flow:
<table>
<thead>
<tr>
<th>Stage</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input data preparation</td>
<td>- find register distribution,</td>
</tr>
<tr>
<td></td>
<td>- find combinational cell distribution</td>
</tr>
<tr>
<td></td>
<td>- get location of new-registers</td>
</tr>
<tr>
<td>Tool, flow variable configuration</td>
<td>- prepare dummy netlist</td>
</tr>
<tr>
<td>Generate floorplan</td>
<td>- specify core height and width</td>
</tr>
<tr>
<td></td>
<td>- read in pin placement def</td>
</tr>
<tr>
<td>Create power grid</td>
<td>- specify power and ground straps in top metal layers</td>
</tr>
<tr>
<td></td>
<td>- create and place virtual power and ground pads</td>
</tr>
<tr>
<td>Create and place dummy cells</td>
<td>- Update the reg distribution and combinational cells' distribution based on loc_for_new_reg.rpt</td>
</tr>
<tr>
<td></td>
<td>- create hierarchical plan groups to divide the entire floorplan as per cell distribution</td>
</tr>
<tr>
<td>Connect dummy cells</td>
<td>- create nets to connect the cells with the i/p,o/p ports, clock ports and reset port.</td>
</tr>
<tr>
<td>Perform CTS</td>
<td>- specify clock ports, clock buffers</td>
</tr>
<tr>
<td></td>
<td>- compile clock tree</td>
</tr>
<tr>
<td></td>
<td>- perform post cts optimization</td>
</tr>
<tr>
<td>Perform routing</td>
<td>- global route</td>
</tr>
<tr>
<td></td>
<td>- track route</td>
</tr>
<tr>
<td></td>
<td>- detail route</td>
</tr>
<tr>
<td>IR drop analysis</td>
<td>- generate instance voltage drop map</td>
</tr>
<tr>
<td></td>
<td>- generate voltage drop map</td>
</tr>
</tbody>
</table>

Figure 15: Stages of EIRDA Flow
Let us now take a closer look in each of these stages:

a) Input Data preparation:

   i) Area estimation: The area of the design - modified by designer, is estimated based on the synthesis of the original RTL. Then based on the general starting utilization of about 70%, the area of the floorplan frame was determined.

   ii) Cell distribution statistics: As most of the power and hot-spots are associated to the clock tree, the registers are considered to be the pivot points that guide the final implementation of the design. Thus the register distribution is was measured on the placed and routed design of the original RTL. The register distribution was measured by dividing the floorplan into grids and then checking register count in each grid. For the experiments, the floorplan comprised of 16 such grids.

   iii) New register location: The register locations are reported by ICC in form of

   ```
   set reg_count { 101 100 86 101 126 114 100 91 65 81 71 107 97 118 83 118}
   set comb_count {917 902 998 882 829 865 946 968 1131 972 1070 875 924 848 980 799}
   ```

   Figure 16: Contents of reg_distribution.rpt and comb_distribution.rpt files, generated using report_cell_distribution tcl command on the original design session on ICC shell.

   Thus when the selected registers are reported into a file, it is in the format as shown in the following figure:
For this example, mapped netlist of the original design was used. Using ICC, assuming new logic to be located at the particular location in the core-area, a region was selected from the placed design using the ICC GUI. The selected region would look as shown in following figure as the cells in the region get highlighted on selection.

![Image of selected cells](image.png)

**Figure 18:** The selected/highlighted cells are assumed to be logic to which new cells will be added.

---

```plaintext
set new_reg_loc {{22.2300 179.2000} {25.8400 180.6000}} ...
{{19.1900 170.8000} {22.8000 172.2000}}
```

**Figure 17:** Contents of loc_for_new_reg.rpt, generated using add_extra_logic command on ICC after selecting a logic portion on the GUI.
*add_extra_logic* (a TCL proc defined in the util.tcl) is then used to create the *new_reg_loc.rpt* file that defines *new_reg_loc* tcl array for the \( \{x,y\} \) locations of all the sequential elements selected from the region.

This information (reported in a file “reg_distribution.rpt”) is used as a reference for placing the dummy cells in the EIRDA flow.

A large cell density of the combinational cells can also cause IR drop issues due to large amount of switching. Assuming modified design has similar cell distribution, the combinational cell count in each grid was also reported into a file (comb_distribution.rpt).

Assuming similarity in the placement of the cells between the original design and the modified design, register locations of the placed, original design are reported into a file (using ICC command *get_attr [all_registers] bbox*) *reg_loc.rpt*. This file is later used to implement the dummy flow.

iv) Pin placement: In order to reduce variation of the placement, the primary ports – that guide the cell placement, should be kept in same location. A pin def is therefore created from the placed database of the original design. This pin.def is then used in the reference flow, (and also in the dummy-cell flow) to keep the placement similar.

a) Tool configuration

The primary requirement to run ICC is availability of the Milkyway database of the technology library. These are generated using the Synopsys Milkyway tool.
Milkyway database: Milkyway is a binary format of storing the design information to be used by Synopsys tools in more efficient way – requiring less memory and faster runtime. The technology information from the `tech.lef` and `.lib` files – cell geometry, parasitics, and other characteristics of cells and the technology are stored in form of views. Depending on the type of operation to be done, certain views are more suitable to benefit runtime and memory footprint.

Once the Milkyway library has been created for a technology, it can be used for implementing different designs.

b) **Floorplan generation**

A square floorplan is generated based on the design area.

Following command is used to create the floorplan in ICC:

```
create_floorplan -control_type width_and_height -core_width [expr $offset + $core_edge] -
core_height [expr $offset + $core_edge]
```

Although current implementation is not affected by pin-placement, the same pin placement is kept by reading in the pins.def from the original design.

c) **Create Power Grid**

The essential elements of a power grid, as discussed in previous chapters are the two upper layers that have the stripes and the ring defined. The two layers needed to create horizontal and vertical stripes. The width and spacing are the key specifications. The ring is then created using the two layers.
These created using *place_pads* (TCL command defined in the util.tcl, detailed in appendix) used to create the power grid. It creates the power and ground stripes, and place the virtual pads (power and ground) across the floorplan.

**d) Create Dummy Cells**

There are two stages of cell insertion, as done using the proc `createPlace_reg_comb`, to account for the combinational cells and sequential cells. The combinational logic of design is accounted by instantiating nand cells in the design and the sequential logic is accounting by instantiating registers.

The core area is divided into a grid of hierarchical modules. ICC allows doing this through `plan_group`. In each of these grids, register-xor pair and nand gates are created using native ICC command *create_cell*. The number of register-xor pairs to be placed in each grid is determined from the *reg_distribution.rpt* and *comb_distribution.rpt* files.

Figure 19: Code snippet to create power grid. Please see appendix for details.

```tcl
... set_fp_rail_constraints -add_layer -layer metal9 -min_strap 8 -max_strap 8 set_fp_rail_constraints -add_layer -layer metal10 -min_strap 8 -max_strap 8 ... create_fp_virtual_pad -nets vdd -point “x y” ... synthesize_fp_rings -nets {vdd gnd} ... apply_switching ... synthesize_fp_rail -synthesize_power_plan -nets {vdd gnd} -analyze_power ...
```
Following figure illustrates the instantiation of these cells in the different hierarchical modules.

Figure 21: Placing the cells in hierarchical modules. Number of cells in each grid is determined based on reg_distribution.rpt and comb_distribution.rpt and loc_for_new_reg.rpt.

The loc_for_new_reg.rpt has the coordinates of register to which the pipelined registers were added. These locations are then mapped to a particular grid.
e) Connect Dummy Cells

Following figure shows the connectivity of the different cells in each hierarchical instances.

*Figure 22: Connecting the different inputs of the combinational and sequential cells with the module's input ports. The output ports are created to connect output of each register.*

created in ICC. Each hierarchical module has four input pins created on them – clock, and reset pins for all the registers and two other input pins to act as data pins. As shown in the above figure, the combinational cells – the nand gates are connected to one input port while the registers are connected to another input pin. This is done to control the switching numbers of the combinational logic and the sequential logic separately. This provision can allow adding a more sophisticated mechanism of controlling the switching mechanism for a future work to improve the accuracy. Also, as shown, the registers are connected with xor-gates in a loop back mode. This is done to allow maximum toggling and switching power in the sequential logic for a conservative analysis.
Figure 23: Connecting the input ports of all the hierarchical modules with the corresponding primary input ports

After instantiation all the cells and connecting them in the hierarchical instances, each of the pins in each of the hierarchical instances are connected to the four primary pins. This is illustrated in the figure below.

f) **Perform Placement, CTS, and Routing**

Once all the cell creation and all the needed connectivity is done, the design is taken through the basic steps of placement, clock tree synthesis and routing.
Figure 24: Placement of combinational and sequential cells and their connectivity as seen at the end of routing.
Following figure shows the placed and routed design at the end of the flow:

Figure 25: Diving the floorplan into multiple grids to measure register and combinational cell distribution per grid.

g) Generate IR drop map and power reports

The voltage drop maps show a qualitative idea about the regions of maximum voltage drop. Design engineers can then accordingly take pre-emptive measures to minimize the drop in those regions right from floorplanning stage. They can try beefing up the power grid in regions of larger voltage drop and possibly allocating more signal tracks to regions with less voltage drop there by leading to a more optimal design.
Comprehending different power and voltage maps (on Synopsys ICC GUI window)

a) Voltage drop map: The following figure shows the voltage drop along the power (vdd) straps. Red indicates maximum drop – the color coding is indicated on the right.

![Figure 26: ICC GUI showing voltage drop map.](image)

b) Instance Voltage drop map:

Following figure shows the voltage drop map on the instances.
Figure 27: ICC GUI showing instance voltage map.

Table 1: Correlation between the instance powers for design Cortex.

<table>
<thead>
<tr>
<th>Cortex</th>
<th>Reference Flow</th>
<th>Dummy Flow</th>
<th>Dummy/Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock Tree</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal Power</td>
<td>1.48E-02</td>
<td>2.71E-03</td>
<td>1.83E-01</td>
</tr>
<tr>
<td>Clock Tree</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Net Power</td>
<td>9.16E-03</td>
<td>9.36E-03</td>
<td>1.02E+00</td>
</tr>
<tr>
<td>Total clock tree power</td>
<td>2.40E-02</td>
<td>1.21E-02</td>
<td>5.04E-01</td>
</tr>
<tr>
<td>Register</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal power</td>
<td>0.1284</td>
<td>0.1809</td>
<td>1.41</td>
</tr>
<tr>
<td>Register</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Net Power</td>
<td>1.38E-02</td>
<td>5.32E-03</td>
<td>3.86E-01</td>
</tr>
<tr>
<td>Total Register power</td>
<td>1.42E-01</td>
<td>1.86E-01</td>
<td>1.31E+00</td>
</tr>
<tr>
<td>Combinational Internal power</td>
<td>0.1029</td>
<td>0.9242</td>
<td>8.98</td>
</tr>
<tr>
<td>Combinational</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Net power</td>
<td>8.23E-02</td>
<td>1.62E-02</td>
<td>1.97E-01</td>
</tr>
<tr>
<td>Total combinational</td>
<td>1.85E-01</td>
<td>9.40E-01</td>
<td>5.08E+00</td>
</tr>
</tbody>
</table>
In order to improve the correlation between the reference and EIRDA flows, the input to the combinational cells was tuned for a non-default switching factor and static probability:

* `set_switching_activity -period $CLK_PER -toggle_rate 0.00001 -static_probability 0.01 NMI`

In general, the early IR drop analysis flow (EIRDA flow) tends to give higher power numbers because it is inherently pessimistic in the current approach due to the continual switching of all the cells. This is further evident from the larger difference in combinational power.
5. Experiments and Results

Following figure shows the three flows that were run on each design:

![Diagram of flows]

- **Original Design**
  - Run through PnR
  - Cell statistics reported

- **Design altered by script**
  - Run through PnR
  - IR drop maps and power numbers generated

- **Proxy Design**
  - Run through PnR
  - IR drop maps and power numbers generated.

- **Validation**
  - Compare IR drop maps and power numbers from the proxy design and design altered by script.

*Figure 28: Flows run to generate the validation results.*

i) ‘Original’ flow: The ‘original’ design (in the form of RTL or netlist) is taken through the place-and-route flow to generate the inputs for the reference flow and the dummy cell placement flow. Depending on the area as estimated using design compiler, the core-area and the floorplan was created. This area can be reported at pre-compilation stage. The modified netlist and the def with pin positions were generated for the reference flow. The cell distribution – both combinational cells
and registers were generated, and the location of the registers – to which register – xor pair are added, are also reported.

ii) Design altered by script: For experiments, as set of original RTL and RTL modified by designers are not available, an automated way of modifying the original design using scripts is done. This modified netlist generated from the original flow is taken through the place-and-route flow to generate the reference IR drop results. These results are used to compare with results obtained on dummy –cell flow.

iii) EIRDA flow: A dummy netlist is created that has only the top-level interface definition. Then based on the TCL procs, a proxy design is created in ICC. The pin positions from the def generated from ‘Original’ flow are used to create the floorplan. Power stripes and power pads are created followed hierarchical modules using plan_group. The registers and other logic gates are created, placed and then connected. These are then taken through remaining stages of the place-and-route flow – clock-tree-synthesis and routing.

Netlist modification:
In absence of set of original and modified version of RTLS, a netlist based version was used for validation experiments. The original design was modified to create an updated version as “altered by script” design. This was done by selecting a set of instances in a region on the routed database.

All the registers from this region were then connected with a register-xor cell pair as shown in the following figure:
New Register locations: These new registers added in the design would change the register distribution on the grid. Considering they will be placed based on their connection to the original registers, it is assumed that placement position of these new registers should be in the same grid as the original register. So a `loc_for_new_reg.rpt` is generated which is used by the dummy-cell flow in placing the dummy registers accordingly.

Following pages show the results of implementing the flow on different designs. For qualitative check, the voltage drop maps on power straps and instances are shown (from the ICC GUI) and for quantitative comparison, dynamic power numbers are compared.
Interpreting the maps and power table:
- Figures on the left correspond to results from new flow implementation while figures from right show results from the normal flow done on ‘reference’ design.
- The power calculation was done on 0.95V.
- Each color ‘pixel’ represents an instance.
- The entire range of drop is binned into 10 groups (which can be changed ICC)
- The IR drop maps are plotted in mV
- If an instance has a drop less than the minimum threshold drop, it is not colored. Thus is evident from some of the uncolored regions in the maps.
- The power numbers are in mW units

Figure 30: Interpreting the IR drop maps and power reports.
Figure 31: ICC GUI showing instance voltage map using EIRDA flow.

Figure 32: ICC GUI showing instance voltage map using reference flow.
Figure 33: Voltage drop on power straps on EIRDA flow.

Figure 34: ICC GUI showing voltage map using reference flow
Power Number comparison:

Table 2: Correlation between the instance powers for design Cortex.

<table>
<thead>
<tr>
<th></th>
<th>Reference Flow</th>
<th>Dummy Flow</th>
<th>Dummy/Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock Tree</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal Power</td>
<td>1.48E-02</td>
<td>2.71E-03</td>
<td>1.83E-01</td>
</tr>
<tr>
<td>Clock Tree</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Net Power</td>
<td>9.16E-03</td>
<td>9.36E-03</td>
<td>1.02E+00</td>
</tr>
<tr>
<td>Total clock tree</td>
<td>2.40E-02</td>
<td>1.21E-02</td>
<td>5.04E-01</td>
</tr>
<tr>
<td>Register</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal power</td>
<td>0.1284</td>
<td>0.1809</td>
<td>1.41</td>
</tr>
<tr>
<td>Register</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Net Power</td>
<td>1.38E-02</td>
<td>5.32E-03</td>
<td>3.86E-01</td>
</tr>
<tr>
<td>Total Register</td>
<td>1.42E-01</td>
<td>1.86E-01</td>
<td>1.31E+00</td>
</tr>
<tr>
<td>Combinational</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal power</td>
<td>0.1029</td>
<td>0.9242</td>
<td>8.98</td>
</tr>
<tr>
<td>Combinational</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Net power</td>
<td>8.23E-02</td>
<td>1.62E-02</td>
<td>1.97E-01</td>
</tr>
<tr>
<td>Total combinational</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>power</td>
<td>1.85E-01</td>
<td>9.40E-01</td>
<td>5.08E+00</td>
</tr>
</tbody>
</table>
Figure 35: ICC GUI showing instance voltage map using EIRDA flow.

Figure 36: ICC GUI showing instance voltage map using reference flow.
Figure 37: ICC GUI showing voltage map using EIRDA flow.

Figure 38: ICC GUI showing voltage map using reference flow.
Table 3: Correlation between the instance powers for design Jacobi.

<table>
<thead>
<tr>
<th>Jacobi</th>
<th>Reference Flow</th>
<th>Dummy Flow</th>
<th>Dummy/Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock Tree Internal Power</td>
<td>4.71E-02</td>
<td>4.84E-03</td>
<td>1.03E-01</td>
</tr>
<tr>
<td>Clock Tree Net Power</td>
<td>2.91E-02</td>
<td>2.56E-02</td>
<td>8.81E-01</td>
</tr>
<tr>
<td>Total clock tree power</td>
<td>7.62E-02</td>
<td>3.05E-02</td>
<td>4.00E-01</td>
</tr>
<tr>
<td>Register Internal power</td>
<td>0.2323</td>
<td>0.3968</td>
<td>1.708136031</td>
</tr>
<tr>
<td>Register Net Power</td>
<td>5.71E-02</td>
<td>1.26E-02</td>
<td>2.21E-01</td>
</tr>
<tr>
<td>Total Register power</td>
<td>2.89E-01</td>
<td>4.09E-01</td>
<td>1.41E+00</td>
</tr>
<tr>
<td>Combinational Internal power</td>
<td>0.3209</td>
<td>0.2851</td>
<td>0.888438766</td>
</tr>
<tr>
<td>Combinational Net power</td>
<td>0.3237</td>
<td>0.2217</td>
<td>0.6848934198</td>
</tr>
<tr>
<td>Total combinational power</td>
<td>0.6446</td>
<td>0.5068</td>
<td>0.7862240149</td>
</tr>
</tbody>
</table>
Design: s38584

Figure 39: ICC GUI showing instance voltage map using EIRDA flow.

Figure 40: ICC GUI showing instance voltage map using reference flow.
Figure 41: ICC GUI showing voltage map using EIRDA flow.

Figure 42: ICC GUI showing voltage map using reference flow.
Table 4: Correlation between the instance powers for design S38584.

<table>
<thead>
<tr>
<th>S38584</th>
<th>Reference Flow</th>
<th>Dummy Flow</th>
<th>Dummy/Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock Tree Internal Power</td>
<td>2.33E-02</td>
<td>4.07E-06</td>
<td>1.75E-04</td>
</tr>
<tr>
<td>Clock Tree Net Power</td>
<td>1.67E-02</td>
<td>1.69E-05</td>
<td>1.01E-03</td>
</tr>
<tr>
<td>Total clock tree power</td>
<td>3.99E-02</td>
<td>2.10E-05</td>
<td>5.25E-04</td>
</tr>
<tr>
<td>Register Internal power</td>
<td>0.1398</td>
<td>0.2611</td>
<td>1.867668097</td>
</tr>
<tr>
<td>Register Net Power</td>
<td>6.30E-03</td>
<td>8.59E-03</td>
<td>1.36E+00</td>
</tr>
<tr>
<td>Total Register power</td>
<td>1.46E-01</td>
<td>2.70E-01</td>
<td>1.85E+00</td>
</tr>
<tr>
<td>Combinational Internal power</td>
<td>0.1638</td>
<td>0.2021</td>
<td>1.233821734</td>
</tr>
<tr>
<td>Combinational Net power</td>
<td>6.69E-02</td>
<td>0.1702</td>
<td>2.54E+00</td>
</tr>
<tr>
<td>Total combinational power</td>
<td>2.31E-01</td>
<td>0.3723</td>
<td>1.61E+00</td>
</tr>
</tbody>
</table>
Figure 43: ICC GUI showing instance voltage map using EIRDA flow.

Figure 44: ICC GUI showing instance voltage map using reference flow.
Figure 45: ICC GUI showing voltage map using EIRDA flow.

Figure 46: ICC GUI showing voltage map using reference flow.
Table 5: Correlation between the instance powers for design S38417.

<table>
<thead>
<tr>
<th>S38417</th>
<th>Reference Flow</th>
<th>Dummy Flow</th>
<th>Dummy/Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock Tree Internal Power</td>
<td>2.74E-02</td>
<td>3.55E-03</td>
<td>1.29E-01</td>
</tr>
<tr>
<td>Clock Tree Net Power</td>
<td>1.84E-02</td>
<td>1.68E-02</td>
<td>9.13E-01</td>
</tr>
<tr>
<td>Total clock tree power</td>
<td>4.58E-02</td>
<td>2.03E-02</td>
<td>4.44E-01</td>
</tr>
<tr>
<td>Register Internal power</td>
<td>0.1763</td>
<td>0.2585</td>
<td>1.466250709</td>
</tr>
<tr>
<td>Register Net Power</td>
<td>8.01E-03</td>
<td>8.29E-03</td>
<td>1.03E+00</td>
</tr>
<tr>
<td>Total Register power</td>
<td>1.84E-01</td>
<td>2.67E-01</td>
<td>1.45E+00</td>
</tr>
<tr>
<td>Combinational Internal power</td>
<td>0.1321</td>
<td>0.2404</td>
<td>1.81983346</td>
</tr>
<tr>
<td>Combinational Net power</td>
<td>7.02E-02</td>
<td>0.1894</td>
<td>2.70E+00</td>
</tr>
<tr>
<td>Total combinational power</td>
<td>2.02E-01</td>
<td>0.4298</td>
<td>2.12E+00</td>
</tr>
</tbody>
</table>
Design: s13207

Figure 47: ICC GUI showing instance voltage map using EIRDA flow.

Figure 48: ICC GUI showing instance voltage map using reference flow.
Figure 49: ICC GUI showing voltage map using EIRDA flow.

Figure 50: ICC GUI showing voltage map using reference flow.
Table 6: Correlation between the instance powers for design S13207.

<table>
<thead>
<tr>
<th>S13207</th>
<th>Reference Flow</th>
<th>Dummy Flow</th>
<th>Dummy/Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock Tree</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal Power</td>
<td>1.15E-02</td>
<td>1.79E-03</td>
<td>1.56E-01</td>
</tr>
<tr>
<td>Clock Tree</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Net Power</td>
<td>8.50E-03</td>
<td>7.78E-03</td>
<td>9.16E-01</td>
</tr>
<tr>
<td>Total clock tree power</td>
<td>2.00E-02</td>
<td>9.57E-03</td>
<td>4.79E-01</td>
</tr>
<tr>
<td>Register</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal power</td>
<td>7.53E-02</td>
<td>0.1276</td>
<td>1.70E+00</td>
</tr>
<tr>
<td>Register</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Net Power</td>
<td>4.34E-03</td>
<td>0</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>Total Register power</td>
<td>7.96E-02</td>
<td>0.1276</td>
<td>1.60E+00</td>
</tr>
<tr>
<td>Combinational</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal power</td>
<td>4.79E-02</td>
<td>6.03E-02</td>
<td>1.26E+00</td>
</tr>
<tr>
<td>Combinational</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Net power</td>
<td>2.29E-02</td>
<td>1.77E-02</td>
<td>7.71E-01</td>
</tr>
<tr>
<td>Total combinational power</td>
<td>7.09E-02</td>
<td>7.80E-02</td>
<td>1.10E+00</td>
</tr>
</tbody>
</table>
Design: s35932

Figure 51: ICC GUI showing instance voltage map using EIRDA flow.

Figure 52: ICC GUI showing instance voltage map using reference flow.
Figure 53: ICC GUI showing voltage map using EIRDA flow.

Figure 54: ICC GUI showing voltage map using reference flow.
Table 7: Correlation between the instance powers for design S35932.

<table>
<thead>
<tr>
<th>S35932</th>
<th>Reference Flow</th>
<th>Dummy Flow</th>
<th>Dummy/Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock Tree Internal Power</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.90E-02</td>
<td>3.71E-03</td>
<td>1.28E-01</td>
</tr>
<tr>
<td>Clock Tree Net Power</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.96E-02</td>
<td>1.74E-02</td>
<td>8.89E-01</td>
</tr>
<tr>
<td>Total clock tree power</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.86E-02</td>
<td>2.11E-02</td>
<td>4.35E-01</td>
</tr>
<tr>
<td>Register Internal power</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1462</td>
<td>0.2684</td>
<td>1.835841313</td>
</tr>
<tr>
<td>Register Net Power</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3.20E-03</td>
<td>8.65E-03</td>
<td>2.70E+00</td>
</tr>
<tr>
<td>Total Register power</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.49E-01</td>
<td>2.77E-01</td>
<td>1.85E+00</td>
</tr>
<tr>
<td>Combinational Internal power</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>7.21E-02</td>
<td>0.2381</td>
<td>3.30E+00</td>
</tr>
<tr>
<td>Combinational Net power</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8.28E-02</td>
<td>0.1891</td>
<td>2.28E+00</td>
</tr>
<tr>
<td>Total combinational power</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.55E-01</td>
<td>0.4272</td>
<td>2.76E+00</td>
</tr>
</tbody>
</table>
Design: s15850

Figure 55: ICC GUI showing instance voltage map using EIRDA flow

Figure 56: ICC GUI showing instance voltage map using reference flow.
Figure 57: ICC GUI showing voltage map using EIRDA flow.

Figure 58: ICC GUI showing voltage map using reference flow.
Table 8: Correlation between the instance powers for design S15850.

<table>
<thead>
<tr>
<th>S15850</th>
<th>Reference Flow</th>
<th>EIRDA Flow</th>
<th>EIRDA/Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock Tree Internal Power</td>
<td>1.72E-03</td>
<td>1.45E-03</td>
<td>8.43E-01</td>
</tr>
<tr>
<td>Clock Tree Net Power</td>
<td>5.94E-03</td>
<td>5.44E-03</td>
<td>9.16E-01</td>
</tr>
<tr>
<td>Total clock tree power</td>
<td>7.66E-03</td>
<td>6.89E-03</td>
<td>9.00E-01</td>
</tr>
<tr>
<td>Register Internal power</td>
<td>4.82E-02</td>
<td>8.38E-02</td>
<td>1.74E+00</td>
</tr>
<tr>
<td>Register Net Power</td>
<td>1.52E-03</td>
<td>2.86E-03</td>
<td>1.88E+00</td>
</tr>
<tr>
<td>Total Register power</td>
<td>4.97E-02</td>
<td>8.67E-02</td>
<td>1.74E+00</td>
</tr>
<tr>
<td>Combinational Internal power</td>
<td>3.09E-02</td>
<td>0.1067</td>
<td>3.45E+00</td>
</tr>
<tr>
<td>Combinational Net power</td>
<td>1.74E-02</td>
<td>6.52E-02</td>
<td>3.75E+00</td>
</tr>
<tr>
<td>Total combinational power</td>
<td>4.83E-02</td>
<td>1.72E-01</td>
<td>3.56E+00</td>
</tr>
</tbody>
</table>

Factors of miscorrelation:
As seen across the designs, there is order of magnitude of difference in the power numbers for the combinational cells. And also the IR drop contour – which are in mV, differ in magnitude by huge factor despite similar number of cells. This illustrates not just cell-count but order of connection is also key in having a more reliable quantitative correlation between the proxy design and the design altered through script. By utilizing the other statistical information from the RTL – fanout, types of cells, etc, perhaps it can assumed that a better correlation will yield.
6. Summary

Key Contributions:

- An alternative early power grid voltage drop analysis flow:

The existing methodologies of early power grid analysis usually begins at post synthesis stage that usually takes several months after the release of the RTL. However this method makes it possible to start the flow right after the RTL is releases – thus giving a head start by several weeks. The other approach is to use the RTL power number - generating those numbers using a separate tool but that comes with extra license cost.

- Using Synopsys ICC beyond its native capability

The native capability of ICC to perform early IR drop analysis is possible on a post placed design, which is bit obvious because the starting input for place-and-route is a synthesized netlist. However, using this method, a dummy netlist is sufficient to meet the basic requirement and then the key method of creating the dummy cells suffice to form a dummy design for the power grid analysis.

- Creating TCL procs that can be used in general outside the flow

The proc that have been created to support the flow can be otherwise used even outside the flow for analysis of the design. For example, generating a cell distribution report, or generating the average fanout in different grid.
Results & issues:

Table 9: Summary of comparison of power numbers over multiple designs for validation.

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Design</th>
<th>Switching power combinational</th>
<th>Switching Power Register</th>
<th>Switching power Clock Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cortex</td>
<td>5</td>
<td>1.31</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>Jacobi</td>
<td>0.78</td>
<td>1.41</td>
<td>0.4</td>
</tr>
<tr>
<td>3</td>
<td>S83584</td>
<td>1.61</td>
<td>1.36</td>
<td>0.001</td>
</tr>
<tr>
<td>4</td>
<td>S38417</td>
<td>2.12</td>
<td>1.45</td>
<td>0.44</td>
</tr>
<tr>
<td>5</td>
<td>S13207</td>
<td>1.1</td>
<td>1.6</td>
<td>0.479</td>
</tr>
<tr>
<td>6</td>
<td>S35932</td>
<td>2.76</td>
<td>1.85</td>
<td>0.435</td>
</tr>
<tr>
<td>7</td>
<td>S15850</td>
<td>3.56</td>
<td>0.867</td>
<td>0.9</td>
</tr>
</tbody>
</table>

The above table summarizes the ratio of power numbers between the EIRDA flow and design modified through script across multiple designs. As seen in above table, the ratio of register/sequential power is and clock tree varies in a smaller range but there is a high variation in the power number correlation particularly for the combinational logic. The IR-drop maps too have a large degree of variation in the absolute value of IR drops in the instances. So it is mainly the similarities in the regions of hot-spots that may be considered to be predicted by the EIRDA flow (albeit with outliers in designs like Cortex).

Key value propositions:
- Design houses can save on license fees of different tools. This alternative flow using the place-and-route tools – that any design team has, allows to save license fee off using a separate tool like Red Hawk for early power analysis. For sign-off power grid analysis however such specialized tools are still required.

- This approach can be used on place-and-route tools from different companies like Cadence’s Encounter or Mentor Graphic’s Olympus.
7. Current Limitations and Futurescope

The primary contribution of current work is in showing viability of an alternative early power grid analysis flow using commonly used place-and-route tool (instead of using a special tool for the purpose). The experiments have been done, to present a proof of concept, on simple designs in single clock domain. However, the work is limited in two main aspects – greater correlation of power numbers with reference designs and types of design constraints it can handle. The switching of combinational cells, for example, needs to be automated – may be built around a function dependent on timing or average fanout statistics of the design.

Inaccuracy in the correlation is an issue that can attempted to be fixed in next level of work by incorporating other information – as mentioned earlier, from the RTL designed by designer (beyond using only the cell distribution information)

The plan_group used for placing the standard cells do allow any new cell that does not belong to the module to be placed inside its boundary. Therefore during clock tree synthesis the clock tree buffers are placed outside the hierarchical modules. The gap between the modules as shown in the figures is deliberately kept to let the tool place the clock tree buffers in those gaps.

Most designs have – multiple clocks, complicated clock structure, clock-gating, multiple modes, memories etc. Also the analysis have been done at a single corner. Most designs of today are UPF based. Thus the logical next step would be perform experiments to extend the flow to include such types of designs.

In the current flow, the register and combinational logic have been created and placed without any direct association to the modules. However it can be more valuable to also incorporate the module placement information by using previous results.
Early estimation is always a challenging problem considering the lack of information and the many assumptions that have to be made in predicting reliable results. Unavailability of actual gates, nets makes the early RC extraction a very wild problem. However considering most of the designs are practically built through releases, there are ample scopes of using placement and other design information of previously closed designs in a newer implementation. This work opens up the possibility of employing statistical properties of designs to approximate the post-routed state of a design to its RTL representation. By correlating more parameters to recognize relations between RTL (post-elaboration graphs) and its final/post-routed implemented state to develop a more reliable EIRDA methodology in the future.
REFERENCES


APPENDIX A

TCL procs used in the flow:

proc find_pin_loc {} { 
    echo "" > pin_loc.rpt 
    foreach_in_collection cel [all_inputs] { 
        echo "[get_attribute $cel name] [get_attribute $cel bbox]" >> pin_loc.rpt 
    } 
    foreach_in_collection cel [all_outputs] { 
        echo "[get_attribute $cel name] [get_attribute $cel bbox]" >> pin_loc.rpt 
    } 
}

proc find_reg_loc {} { 
    echo "" > reg_loc.rpt 
    # cat regl.rpt | awk '{print $2,$3} | sed 's/\{/g' | sed 's/ /,g' | sed 's/}/;g' 
    foreach_in_collection cel [all_registers] { 
        echo "[get_attribute $cel name] [get_attribute $cel bbox]" >> reg_loc.rpt 
    } 
}

proc rpt_power {stage} { 
    global PER 
    report_power -analysis_effort high > ${stage}_DC_vs_power_PM_period${PER}.rpt 
    for {set i 0} {$i < 1.1} {set i [expr $i + 0.1]} { 

#set_switching_activity -period 5 -toggle_rate $i -static_probability 0.5
[remove_from_collection [get_nets -hier *] [get_nets -hier *clk*]]

set_switching_activity -period 3 -toggle_rate 0.5 -static_probability $i [get_pins */*_reg/CK]
report_power -analysis_effort high >> ${stage}_DC_vs_power_PM_period$PER.rpt
reset_switching_activity
}
report_power -analysis_effort high > ${stage}_SF_vs_power_PM_period$PER.rpt
for {set i 0} {$i < 1.1} {set i [expr $i + 0.1]} {
    #set_switching_activity -period 5 -toggle_rate $i -static_probability 0.5
[remove_from_collection [get_nets -hier *] [get_nets -hier *clk*]]
    set_switching_activity -period $PER -toggle_rate $i -static_probability 0.5 [get_pins */*_reg/CK]
    report_power -analysis_effort high >> ${stage}_SF_vs_power_PM_period$PER.rpt
    reset_switching_activity
}
}

proc get_avg_slack {stage} {
    global PER
    set avg_slack 0
    set slack_list ""
    foreach_in_collection P [get_pins */*_reg/D] {

set i [get_timing_paths -to $P]

lappend slack_list [get_attr $i slack]

set avg_slack [expr $avg_slack + [get_attr $i slack]]

}

set avg_slack [expr $avg_slack/llength $slack_list]]

puts "$avg_slack"

echo "$avg_slack" >> ${stage}_slack_list_at_${PER}.list

echo "$slack_list" >> ${stage}_slack_list_at_${PER}.list

}

set avg_fanout 0

proc get_avg_fanout {} {

set avg_fanout 0

echo "" > avg_fo.rpt

foreach_in_collection fp [all_registers] {

set fp [get_attr $fp name]

set pin_list [get_pins -of_objects */$fp -filter "@direction==out"]

set avg_fanout_i 0

foreach_in_collection pin $pin_list {

set avg_fanout_i [expr [sizeof_collection [all_connectivity_fanout -from $pin]] + $avg_fanout_i]

}

set avg_fanout_i [expr $avg_fanout_i/[sizeof_collection $pin_list]]

echo "$fp $avg_fanout_i" >> avg_fo.rpt
set avg_fanout [expr $avg_fanout + $avg_fanout_i]
}
set $avg_fanout [expr $avg_fanout/sizeof_collection [all_registers]]
puts "$avg_fanout"
}

##### createModuleRegion creates a ICC command file to create different
##### regions for different modules by reading another file that has this information.
##### this module placement is based on previous placement results of the design.
proc createModuleRegion {moduleplan} {
# read a text file - "prev_placement.rpt", to collect "<module> <region>" information
set fp [open $moduleplan]
set module_loc [read $fp]
close $fp

# create a command file - "createModuleRegion.tcl" based on these relations
# create region for each module
for {set i 0} {$i < [llength $module_loc]} {incr i} {
set m_loc [lindex $module_loc $i]
set m_name [lindex $module_loc [incr i]]
create_plan_groups -coordinate $m_loc $m_name
}
}
proc createModule {} {
    set buf_type "BUF_X4"
    global moduleName
    global area
    set moduleArea $area
    global core_edge
    global offset
    global clk_name

    #set moduleName_xys "${offset} ${offset} [expr $core_edge - $offset] [expr $core_edge - $offset]"
    set moduleName_xys "0 0 $core_edge $core_edge"
    create_cell -hierarchical $moduleName
    create_plan_groups -coordinate "$moduleName_xys" $moduleName
}

proc connectTopNets {} {
    global clk_name
    global moduleName
    global rst_name
    create_net $clk_name
    create_net $rst_name
    connect_net $clk_name HCLK
connect_net $clk_name $moduleName/$clk_name
connect_net $rst_name $rst_name
connect_net $rst_name $moduleName/$rst_name
}

proc createCellsNets {} {
    set buf_type "BUF_X4"
    global moduleName
    global area
    set moduleArea $area
    global core_edge
    global offset
    createModule
    # create the cells and the nets
    for {set i 0} {$i<$count} {incr i} {
      create_cell $moduleName/$i
      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm/$buf_type
    }
    for {set i 0} {$i<$count} {incr i} {
      create_net $moduleName/net_${i}[expr ${i} +1]
      if {~$i} {
        # the first net is connected to $module_input_pin
      }
    }
}
#

connect_net $moduleName/net_${i}_[expr ${i} +1] $module_input_pin

}

if {$i == [expr $count - 1]} {

# the last net is connected to $module_output_pin

# connect_net $moduleName/net_${i}_[expr ${i} +1] $module_output_pin

}

# connect the nets to the previous and next stages

connect_net $moduleName/net_${i}_[expr ${i} +1] [get_pins $moduleName/[expr $i - 1]/Z]

connect_net $moduleName/net_${i}_[expr ${i} +1] [get_pins $moduleName/$i/A]

}
}

###########################

## include registers

proc createPlaceReg {} {

puts "Adding registers and XOR2 - one input set high to toggle at every clock"

global reg_count

global moduleName

global reg_type

global clk_name

global rst_name
global avgFanout
global xor_type
global ref_reg_loc
global place_err

#--------------------------------------------------------------------------------
cREATEMODULE

set fp [open "$ref_reg_loc" r]
set loc_list [read $fp]
close $fp

create_port -direction "in" "$moduleName/$clk_name"
create_port -direction "in" "$moduleName/$rst_name"
create_net ${moduleName}/${clk_name}
create_net ${moduleName}/${rst_name}
connect_net ${moduleName}/${rst_name} ${moduleName}/${rst_name}
connect_net ${moduleName}/${clk_name} ${moduleName}/${clk_name}

for {set i 0} {$i < [expr [llength $loc_list]/2]} {set i [expr $i + 2]} {
    set name "$i"
    create_cell ${moduleName}/${name}_reg
    NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm/$reg_type
create_cell \$\{moduleName\}/$\{name\}_xor

NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm/$xor_type

create_net \$\{moduleName\}/n$\{name\}_reg

create_net \$\{moduleName\}/n$\{name\}_xor

connect_net \$\{moduleName\}/n$\{name\}_reg \$\{moduleName\}/$\{name\}_xor/A

connect_net \$\{moduleName\}/n$\{name\}_reg \$\{moduleName\}/$\{name\}_reg/Q

connect_net \$\{moduleName\}/n$\{name\}_xor \$\{moduleName\}/$\{name\}_xor/Z

connect_net \$\{moduleName\}/n$\{name\}_xor \$\{moduleName\}/$\{name\}_reg/D

set x [lindex $loc_list $i]

set y [lindex $loc_list [expr 1 + $i]]

set_cell_location -coordinates "$x $y" \$moduleName/$\{name\}_reg -fixed

}

connect_net \$\{moduleName\}/$\{clk_name\} \$\{moduleName\}/*_reg/CK

connect_net \$\{moduleName\}/$\{rst_name\} \$\{moduleName\}/*_reg/RN

# add the extra registers

}

proc load_registers {} {

global avg_fanout
global buf_type

global moduleName

global inv_type

set buf_type $inv_type

# load Q of each flop with buffers equal to avg fanout of the flops

foreach_in_collection fp [all_registers] {
    set regName [get_attribute $fp name]
    #
    create_net $moduleName/net_buf_${regName}
    connect_net $moduleName/net_buf_${regName} $moduleName/${regName}/Q
    for {set i 0} {$i<$avg_fanout} {incr i} {
        #create_cell $moduleName/${i}_buf_${regName}
        NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm/$buf_type
        create_cell $moduleName/${i}_buf_${regName}
        NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm/$inv_type
        connect_net $moduleName/net_buf_${regName} $moduleName/${i}_buf_${regName}/A
    }
}
}

proc createBufRegNets {} {
    global clk_name
    global reg_type

global reg_count

global offset

global areaa

global buf_type

global reg_type

set moduleName "dummy2"

set moduleArea $area

set reg_count 1000

set moduleName_xys {
$offset $offset [expr $core_edge - $offset] [expr $core_edge - $offset]}

set clk_name "sysclk"

set avgFanout 50

set reg_area [expr $reg_count*[get_attr 
NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm/$reg_type area ]]

set buf_count [expr ($moduleArea - $reg_area)/[get_attribute 
NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm/BUF_X4 area]/1.0]

createModule

# create the cells and the nets

## ADD BUFFERS

for {set i 0} {$i<$buf_count} {incr i} {

create_cell $moduleName/${i}_buf
NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm/${buf_type}
}

### ADD REGS
for {set i 0} {$i<$reg_count} {incr i} {
    create_cell $moduleName/${i}_reg
NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm/${reg_type}
}

### ADD NETS FOR BUFFERS
for {set i 0} {$i<$buf_count} {incr i} {
    create_net $moduleName/net_${i}_buf_[expr ${i} +1]
    if {!$i} {
        # the first net is connected to $module_input_pin
        #
        connect_net  $moduleName/net_${i}_[expr ${i} +1] $module_input_pin
    }
    if {$i == [expr $buf_count-1]} {
        # the last net is connected to $module_output_pin
        #
        connect_net  $moduleName/net_${i}_[expr ${i} +1] $module_output_pin
    }
    # connect the nets to the previous and next stages
}
connect_net $moduleName/net_${i}_buf_[expr ${i} +1] [get_pins $moduleName/[expr $i - 1]_buf/Z]

    connect_net $moduleName/net_${i}_buf_[expr ${i} +1] [get_pins $moduleName/$ModuleName/${i}_buf/A]
}

### ADD NETS FOR REGS
create_net "$moduleName/$clk_name"
for {set i 0} {$i<$reg_count} {incr i} {
## pins in this REG "$moduleName/${i}_reg" are:
## inputs   CK, D,
## outputs  Q, QN
create_net $moduleName/net_${i}_reg_[expr ${i} +1]
if {!$i} {
# the first net is connected to $module_input_pin
#    connect_net $moduleName/net_${i}_[expr ${i} +1] $module_input_pin
}

if {$i == [expr $buf_count-1]} {
# the last net is connected to $module_output_pin
#    connect_net $moduleName/net_${i}_[expr ${i} +1] $module_output_pin
}

# connect the nets to the inputs (D,CK) to previous and outputs (Q,QN) next stages
# connect clk pins to same clk
connect_net $moduleName/$clk_name [get_pins $moduleName/${i}_reg/CK]

# connect D input to
}

create_plan_groups -coordinate $moduleName_xys $moduleName
}

proc apply_switching {} {
    global input_switching_factor
    global input_static_probability
    global reg_switching_factor
    global reg_static_probability
    global CLK_PER
    global moduleName
    foreach_in_collection ip [all_registers] {
        set regName [get_attribute $ip name]
        #set_switching_activity -period $CLK_PER -toggle_rate $reg_switching_factor -static_probability $reg_static_probability [get_pin $moduleName/$regName/D]
        set_switching_activity -period $CLK_PER -toggle_rate $reg_switching_factor -static_probability $reg_static_probability [get_pin */$regName/D]
    }
    set_switching_activity -period $CLK_PER -toggle_rate $input_switching_factor -static_probability $input_static_probability [all_inputs]
    report_power
proc create_reglist_at_input {} {

set ip_flop_collection ""
set pin_list ""

foreach ipName [get_object_name [remove_from_collection [all_inputs] "HCLK HRESETn"]] {
lappend pin_list $ipName
set cells [all_fanout -flat -endpoints_only -from $ipName]
#set flops [get_object_name [filter_collection $cells "@is_sequential==true"]]
set flops [get_object_name $cells ]
set ip_flop_collection [concat $ip_flop_collection $flops]
set ip_flop_collection [lsort -unique $ip_flop_collection]
}
echo "$pin_list" > ip_ports_with_flop.rpt
echo "$ip_flop_collection" > flops_at_inputs.rpt
}

proc create_reglist_at_output {} {

set op_flop_collection ""
set pin_list ""
}


```bash
foreach opName [get_object_name [all_outputs]] {
    lappend pin_list $opName

    set flops [get_object_name [all_fanin -flat -startpoints_only -to $opName]]
    #set flops [get_object_name [filter_collection $cells "@is_sequential==true"]]
    set op_flop_collection [concat $op_flop_collection $flops]
    set op_flop_collection [lsort -unique $op_flop_collection]
}

echo "$pin_list" > op_ports_with_flop.rpt
echo "$op_flop_collection" > flops_at_outputs.rpt
}

proc connect_regs_at_io_ports {} {
    global flops_at_inputs_list
    global ip_ports_with_flop
    global moduleName

    set fp [open "$ip_ports_with_flop" r]
    set pin_list [read $fp]
    close $fp

    set fp [open "$flops_at_inputs_list" r]
    set flop_list [read $fp]
    close $fp

    foreach pin $pin_list {
        # create_port $moduleName/$pin -direction in
```
# create_net $pin

    connect_net $pin $pin
    connect_net $pin $moduleName/$pin

} for {set i 0} {$i < [llength $pin_list]} { incr i} {

    create_net n_{[lindex $pin_list $i]}_{[lindex $flop_list $i]}
    connect_net n_{[lindex $pin_list $i]}_{[lindex $flop_list $i]} $moduleName/[lindex $pin_list $i]
    connect_net n_{[lindex $pin_list $i]}_{[lindex $flop_list $i]} [lindex $flop_list $i]
}

proc place_pads {} {

    global power
    derive_pg_connection -power_net vdd -ground_net gnd
    set_fp_rail_strategy -use_tluplus false
    set_fp_rail_constraints -set_ring -extend_strap boundary
    set_fp_rail_constraints -add_layer -layer metal9 -min_strap 8 -max_strap 8
    set_fp_rail_constraints -add_layer -layer metal10 -min_strap 8 -max_strap 8

    ## power pad placement
    global core_width
    global core_height
    #global core_edge
    global space
global offset

global grid_factor

set core_edge [lindex [get_placement_area] 3]

set grid_size [expr $core_edge*$grid_factor]

set core_height [expr $core_edge + $offset]

set core_width [expr $core_edge + $offset]

# virtual PAD placement

set BOX 10

for {set j 1} {$j < $BOX} {incr j} {
    for {set i 1} {[expr (2*$i+1)*$core_edge/$BOX] < $core_edge} {incr i} {
        create_fp_virtual_pad -nets vdd -point 
        [expr 2*$i*$core_edge/$BOX] [expr $j*$core_edge/$BOX]
        create_fp_virtual_pad -nets gnd -point 
        [expr (2*$i+1)*$core_edge/$BOX] [expr $j*$core_edge/$BOX]
    }
}

synthesize_fp_rings -nets {vdd gnd}

apply_switching

synthesize_fp_rail -synthesize_power_plan -nets {vdd gnd} -analyze_power

#synthesize_fp_rail -create_virtual_rails

}

proc report_cell_distribution {} {

#global core_edge
#global offset
#global area

global grid_factor

set core_edge [lindex [get_placement_area] 3]

set grid_size [expr $core_edge*$grid_factor]

set comb_count ""

set reg_count ""

for {set j 0} {[expr ($j)*$grid_size] < $core_edge} {incr j} {
    for {set i 0} {[expr ($i)*$grid_size] < $core_edge} {incr i} {
        set llx [expr $i*$grid_size]
        set lly [expr $j*$grid_size]
        set urx [expr ($i+1)*$grid_size]
        set ury [expr ($j+1)*$grid_size]

        lappend comb_count [sizeof_collection [filter_collection [get_cells -within "$llx $lly $urx $ury"] "@is_sequential==false"]]

        lappend reg_count [sizeof_collection [filter_collection [get_cells -within "$llx $lly $urx $ury"] "@is_sequential==true"]]
    }
}

echo "set comb_count {$comb_count}" > comb_distribution.rpt

echo "set reg_count {$reg_count}" > reg_distribution.rpt
proc report_cell_power {var} {
    #global core_edge
    global offset
    global area
    global grid_factor
    #set grid_size [expr $core_edge*$grid_factor]
    set core_edge [lindex [get_placement_area] 3]
    set grid_size [expr $core_edge*$grid_factor]
    
    echo "" > power_distribution.rpt
    for {set j 0} {[expr ($j)*$grid_size] < $core_edge} {incr j} {
        for {set i 0} {[expr ($i)*$grid_size] < $core_edge} {incr i} {
            set llx [expr $i*$grid_size]
            set lly [expr $j*$grid_size]
            set urx [expr ($i+1)*$grid_size]
            set ury [expr ($j+1)*$grid_size]
            #        puts "$llx $lly $urx $ury"
            report_power -cell -only [get_cells -within "$llx $lly $urx $ury"] >> power_distribution.rpt
        }
    }
    set power_map "[ sh grep "Totals (" power_distribution.rpt | sed 's/uW//g' | sed 's/nW//g' | sed 's/mW//g' | awk '{print \$6}']"
    echo "$var = \" > power_map.csv
set row_size [expr $core_edge/$grid_size]

for {set i 0} {$i < [expr $row_size*$row_size]} {set i [expr int($i + $row_size)]} {
    echo "[lrange $power_map $i [expr int ($i + $row_size - 1)]];" >> power_map.csv
}

echo "\"" >> power_map.csv
}

proc add_extra_logic {} {
    # connect N new registers to N registers
    #global new_reg_count
    global xor_type
    #global regset
    set reg_type DFF_X2
    set xor_type XOR2_X2
    #set new_reg_count 30
    # following file is created by selecting cells in a particular region on the placed original
    design
    # or by selecting registers of hierarchical modules/subdesigns
    # the registers' locations are then reported in the loc_for_new_reg.rpt
    set regset [get_object_name [filter_collection [get_selection] "@is_sequential==true"]]
    #***** set regset [filter_collection [get_cells $hier_name/*] "@is_sequential==true"]
    #***** echo "set regset {[get_object_name $regset]}" > reg_for_new_reg.rpt
    #source reg_for_new_reg.rpt
# report register location

set new_reg_loc ""

foreach rs $regset {
    puts "[sh dirname $rs]"
    lappend new_reg_loc [get_attribute $rs bbox]
}

echo "set new_reg_loc {$new_reg_loc}" > loc_for_new_reg.rpt

for {set i 0} { $i < [llength $regset]} {incr i} {
    set moduleName "[sh dirname [lindex $regset $i]]"
    create_cell ${moduleName}/${i}_reg
    NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm/$reg_type
    create_cell ${moduleName}/${i}_xor
    NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm/$xor_type
    create_net ${moduleName}/${i}_reg_xor
    connect_net ${moduleName}/${i}_reg_xor ${moduleData}/$i_reg/D
    connect_net ${moduleName}/${i}_reg_xor ${moduleData}/$i_xor/Z
    set nn [lindex $regset $i]
    set nn_net [all_connected $nn/Q ]
    disconnect_net $nn_net $nn/Q
    connect_net $nn_net ${moduleData}/$i_reg/Q
    connect_net $nn_net ${moduleData}/$i_xor/A
    create_net ${moduleData}/n__${i}
connect_net ${moduleName}/n__${i} ${nn}/Q
connect_net ${moduleName}/n__${i} ${moduleName}/${i}_xor/B
connect_net [all_connected $nn/CK] ${moduleName}/${i}_reg/CK
}
write_verilog [get_object_name [current_design]]_modified.v
}

proc createPlace_reg_comb {} {
  global offset
  global area
  global grid_factor
  global reg_type
  global nand_type
  global xor_type
  global moduleName
  global clk_name
  global rst_name
  global ip_port
  global ip_comb_port
  global op_port
  global new_reg_loc
  global new_reg_count
  global new_reg_grid_loc
global comb_count

global reg_count

global gap

source comb_distribution.rpt

source reg_distribution.rpt

source loc_for_new_reg.rpt

##########################################################################3
3

set core_edge [lindex [get_placement_area] 3]

set grid_size [expr $core_edge*$grid_factor]

# update reg_count based on new_reg_count and new_reg_grid_loc

#foreach loc $new_reg_grid_loc {
  # set reg_count [lreplace $reg_count $loc $loc [expr int([lindex $reg_count $loc] +
  $new_reg_count/[llength $new_reg_grid_loc])]]
#
# new_reg_loc has the location of reference registers in which the new registers are to be
# added
#
# find the grid of each ref. reg and incr the reg count for that grid by that amount

#set reg_count_n $reg_count
for {set i 0} {$i < [llength $new_reg_loc]} {incr i} {
    set x [lindex $new_reg_loc $i 0 0]
    set y [lindex $new_reg_loc $i 0 1]
    set grid_x_pos [expr int($x/$grid_size)]
    set grid_y_pos [expr int($y/$grid_size)]

    if {$grid_y_pos} {
        set ind [expr int((1/$grid_factor)*$grid_y_pos + $grid_x_pos)]
    } else {
        set ind $grid_x_pos
    }
    puts "init count @ $ind: [lindex $reg_count $ind]"
    set reg_count [lreplace $reg_count $ind $ind [expr [lindex $reg_count $ind] + 1]]
    puts "added count @ $ind: [lindex $reg_count $ind]"
}

set index 0
for {set j 0} {[expr ($j)*$grid_size] < $core_edge} {incr j} {
    for {set i 0} {[expr ($i)*$grid_size] < $core_edge} {incr i} {
        set llx [expr $i*$grid_size + $gap]
        set lly [expr $j*$grid_size + $gap]
        set urx [expr ($i+1)*$grid_size]
        set ury [expr ($j+1)*$grid_size]
    }
}
set comb_cell_cnt [lindex $comb_count $index]

#set levels [expr int(log([lindex $comb_count $index]/log(2)))]

set levels [expr int(log($comb_cell_cnt)/log(2))]

    # create a hierarchical module to place the comb-cells and regs in it

    set moduleName "module_${i}_${j}"

    set moduleName_xys "$llx $lly $urx $ury"

    create_cell -hierarchical $moduleName

    create_plan_groups -coordinate "$moduleName_xys" $moduleName

    create_port -direction "in" "$moduleName/$clk_name"

    create_port -direction "in" "$moduleName/$rst_name"

    create_port -direction "in" "$moduleName/$ip_port"

    create_port -direction "in" "$moduleName/$ip_comb_port"

    create_port -direction "out" "$moduleName/$op_port"

    create_net ${moduleName}/${clk_name}

    create_net ${moduleName}/${rst_name}

    create_net ${moduleName}/${ip_port}

    create_net ${moduleName}/${ip_comb_port}

    create_net ${moduleName}/${op_port}

    connect_net ${moduleName}/${clk_name} $moduleName/$clk_name

    connect_net ${moduleName}/${rst_name} $moduleName/$rst_name
connect_net ${moduleName}/${ip_port} $moduleName/$ip_port
connect_net ${moduleName}/${ip_comb_port} $moduleName/$ip_comb_port
connect_net ${moduleName}/${op_port} $moduleName/$op_port

# create and place comb in this grid-window

for {set p 0} { $p < $comb_cell_cnt} {incr p} {
    set cell_name
    ${moduleName}/${index}_${p}_nand
    create_cell $cell_name
    NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm/$nand_type
}
puts "$p"

# for {set p 0} {$p < $levels} {incr p} {
#    for {set q 0} {$q < [expr 2**$p]} {incr q} {
#        if {$comb_cell_cnt > 0} {
#            set cell_name
#            ${moduleName}/${index}_${p}_${q}_nand
#            create_cell $cell_name
#            NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm/$nand_type
#        }
#    }
#    #create_net ${cell_name}_net_
#} else {
}
puts "Added [lindex $comb_count $index] cells in grid index $index"

break

set comb_cell_cnt [expr $comb_cell_cnt - $q]

connect_net ${moduleName}/${ip_comb_port} ${moduleName}/*_nand/A*

# create and place reg in this grid-window

# reg_count

connect_net ${moduleName}/${rst_name} ${moduleName}/${rst_name}
connect_net ${moduleName}/${clk_name} ${moduleName}/${clk_name}

for {set p 0} {$p < [lindex $reg_count $index]} {incr p} {
    create_cell ${moduleName}/${index}_${p}_reg NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm/$reg_type
    create_cell ${moduleName}/${index}_${p}_xor NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm/$xor_type
    create_net ${moduleName}/n${index}_${p}_reg
    create_net ${moduleName}/n${index}_${p}_xor
    connect_net ${moduleName}/${clk_name} ${moduleName}/${index}_${p}_reg/CK
}
connect_net ${moduleName}/${rst_name} ${moduleName}/${index}_${p}_reg/RN
connect_net ${moduleName}/${ip_port}  ${moduleName}/${index}_${p}_xor/B
connect_net ${moduleName}/n${index}_${p}_reg ${moduleName}/${index}_${p}_xor/A
connect_net ${moduleName}/n${index}_${p}_reg ${moduleName}/${index}_${p}_reg/Q
connect_net ${moduleName}/n${index}_${p}_xor ${moduleName}/${index}_${p}_xor/Z
connect_net ${moduleName}/n${index}_${p}_xor ${moduleName}/${index}_${p}_reg/D
connect_net ${moduleName}/n${index}_${p}_reg ${moduleName}/${op_port}

puts "[lindex $reg_count $index]"

incr index

}

connect_net $clk_name module*/${clk_name}
connect_net $rst_name module*/${rst_name}
connect_net $ip_port module*/${ip_port}
connect_net $ip_comb_port module*/${ip_comb_port}
connect_net $op_port module*/${op_port}

}

run_ice.tcl

source utils.tcl

global core_width
global core_height

global offset

global area

global PER

global power

set cache_read "/local/home/sbhatt10"
set cache_write "/local/home/sbhatt10"
set area [expr 25000/0.7]
set power 10.00
set clk_name "clock"
set rst_name "reset"
set ip_port "updateY"
set op_port "Y_wr_en"
set top "jacobi"
set CLK_PER 15
set PER 15
set offset 4
set space 20
set regCount 10
set moduleName "dummy1"
set gap 2
set buf_type "BUF_X2"
set reg_type "DFFR_X2"
set inv_type "INV_X2"
set xor_type "XOR2_X2"
set nand_type "NAND2_X2"
set avg_fanout 5
set reg_switching_factor 0.5
set input_switching_factor 0.5
set input_static_probability 0.5
set reg_static_probability 0.5
set ref_reg_loc "reg_loc.rpt"
set place_err 0.15
set grid_factor 0.25
set new_reg_count 20
set flops_at_inputs_list "flops_at_inputs.rpt"
set ip_ports_with_flop "ip_ports_with_flop.rpt"
set flops_at_outputs_list "flops_at_outputs.rpt"
set op_ports_with_flop "op_ports_with_flop.rpt"
set link_library_name
{../temp/NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_conditional_nldm_X-2005.09-SP3.db dw_foundation.sldb}
set target_library_name
{../temp/NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_conditional_nldm_X-2005.09-SP3.db}
set link_library $link_library_name
set target_library $target_library_name

set_mw_lib_reference -mw_reference_library

{NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm} counter

open_mw_lib ../../temp/counter
check_library

read_verilog jacobi_dummy.v

source ./org/clk.sdc

set core_edge [expr floor(sqrt($area))]

create_floorplan -control_type width_and_height -core_width $core_width -core_height $core_height

create_floorplan -control_type width_and_height -core_width [expr $offset + $core_edge] -core_height [expr $core_edge + $offset]

legalize_placement

read_def ./../org/pins.def

#set_dont_touch [current_design] false

createPlace_reg_comb

#load_registers

#connectTopNets

#connect_regs_at_io_ports

create_placement
refine_placement

place_pads

set_attr [get_cells -hier *] dont_touch true

## Clock Tree Synthesis

set_clock_tree_references -references

{NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm/CLKBUF_X1
NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm/CLKBUF_X2
NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm/CLKBUF_X3 }

set_clock_tree_options -clock_trees [get_ports $clk_name]

compile_clock_tree -clock_trees $clk_name

set_propagated_clock $clk_name

clock_opt

route_zrt_global

route_zrt_track

route_zrt_detail