

Achieve Low Power Design with Power Compiler

Wu Qifa, Ting Yujing, Xu Wei, Masayuki Tomisawa

OKI TECHNO CENTRE (SINGAPORE) PTE LTD

20 Science Park Road, #02-06/10, Teletech Park,
Singapore Science Park II, Singapore 117674

ABSTRACT

Low power consumption is a vital feature for many kinds of IC chips. Usually good architecture design and advanced technology process can make power consumption as expected. The power consumption can be further optimized with Power Compiler from Synopsys. In our design case, power optimization with RTL clock gating can reduce about 21%, and gate-level power optimization can reduce about 15% power again. In our design case, the power optimization is not affecting the worst negative slack of timing. The area is improved slightly and the fault coverage of DFT is close to the one without the optimization.

The following topics will be discussed in this paper.

- Power analysis and accuracy
- RTL clock gating and its effect
- Gate-level power optimization and its effect
- DFT and timing issue related with power optimization
- Pre-layout vs. post-layout power analysis

1. Introduction

1.1. The necessity of low power design

Power Consumption was paid more and more attention to by IC designers. The motive of low power design comes from two reasons:

- For those chips used in products supplied by battery, such as portable computers and hand-held devices, lower power consumption is one of the key features surpassing their competitors.
- With the steadily increasing of chip's capacity and density, low power consumption become a vital feature for chip's functionality and reliability. High power density will make chip's temperature increasing, thus cause path delay increasing and problem of metal immigration.

Figure 1(a)-(c) shows the tendency of IC design.

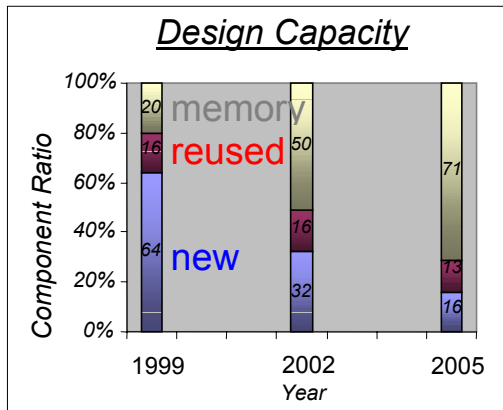


Figure 1(a)

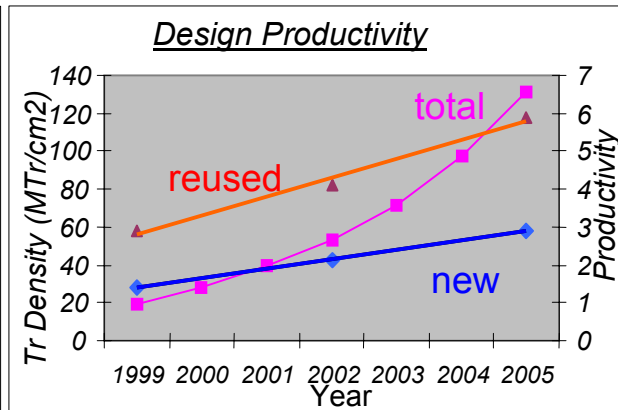


Figure 1(b)

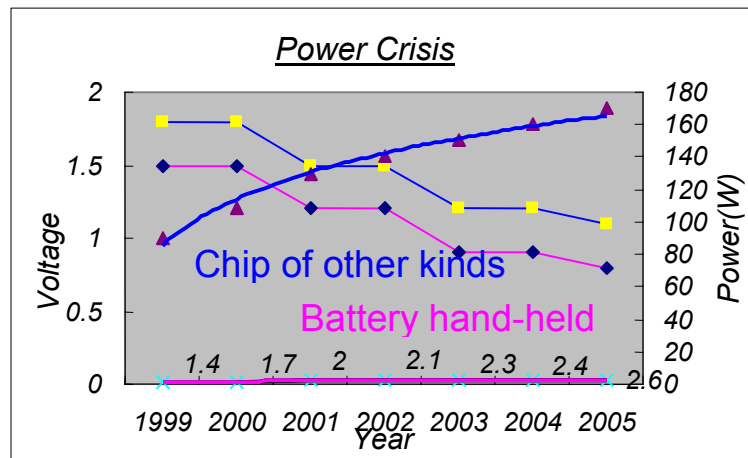


Figure 1(c)

Figure 1: Design Crisis Forecast (Data source: ITRS'00)

Figure 1(a) shows that from year 1999 to year 2005 51% logic area will decrease, while Figure 1(b) shows transistor density will increase to 6.5 times for whole chip. Figure 1(c) shows that power consumption will increase to 2 times during this period even though the supply voltage is decreasing. Obviously, power density of the chip will increase greatly.

1.2. Overview of low power design flow with Power Compiler

Usually good architecture design and advanced technology process can reduce power consumption significantly. But if these factors are fixed, power consumption can be further optimized with Power Compiler from Synopsys. Power Compiler offers analysis and optimization technology from RTL to the gate level.

To reduce design iteration, it is important to be aware of whether the power consumption meets the specification or not in all stages of the whole design cycle. The power analysis technology of Power Compiler analyzes the design for power consumption. Power analysis can be performed at the register transfer level using RTL simulation or at gate level using gate-level simulation.

Working in conjunction with Design Compiler, the power optimization technology of Power Compiler optimizes the design for power consumption, simultaneously for timing and area. Power optimization can be performed at the register transfer level with clock gating and at gate level through back-annotation of switching activity coming from simulation.

Figure 2 is the whole low power design flow with Power Compiler applied in our project. The design flow comprises of three times of power analysis and two kinds of power optimization technology. The three kinds of power analysis: RTL power analysis, pre-layout gate-level power analysis (before power optimization and after power optimization) and post-layout gate-level are based on different kinds of simulation: RTL simulation, pre-layout gate-level simulation and post-layout gate-level simulation. The two kinds of power optimization technology: clock gating and gate level power optimization, one is focused on optimization for timing and area, another is focused for power as well as timing and area.

The first power analysis is called as RTL power analysis, which uses switching activity (SAIF) file coming from RTL simulation. Although power analysis at this stage has only relatively good accuracy, from which we can obtain quick power estimation numbers early in the design flow. This is helpful to estimate packaging and battery requirement, as well as aiding in power grid design. It can also help to perform architectural tradeoff early in the design flow, identify hotspots in a large, complex design. For example, if the power estimation is too high, we can focus our efforts on reducing it on the modules that actually consume the largest amounts of power in the circuit.

Power optimization achieved at RTL has an increasingly important impact on power reduction. Gating clocks of register banks is the main power optimization step.

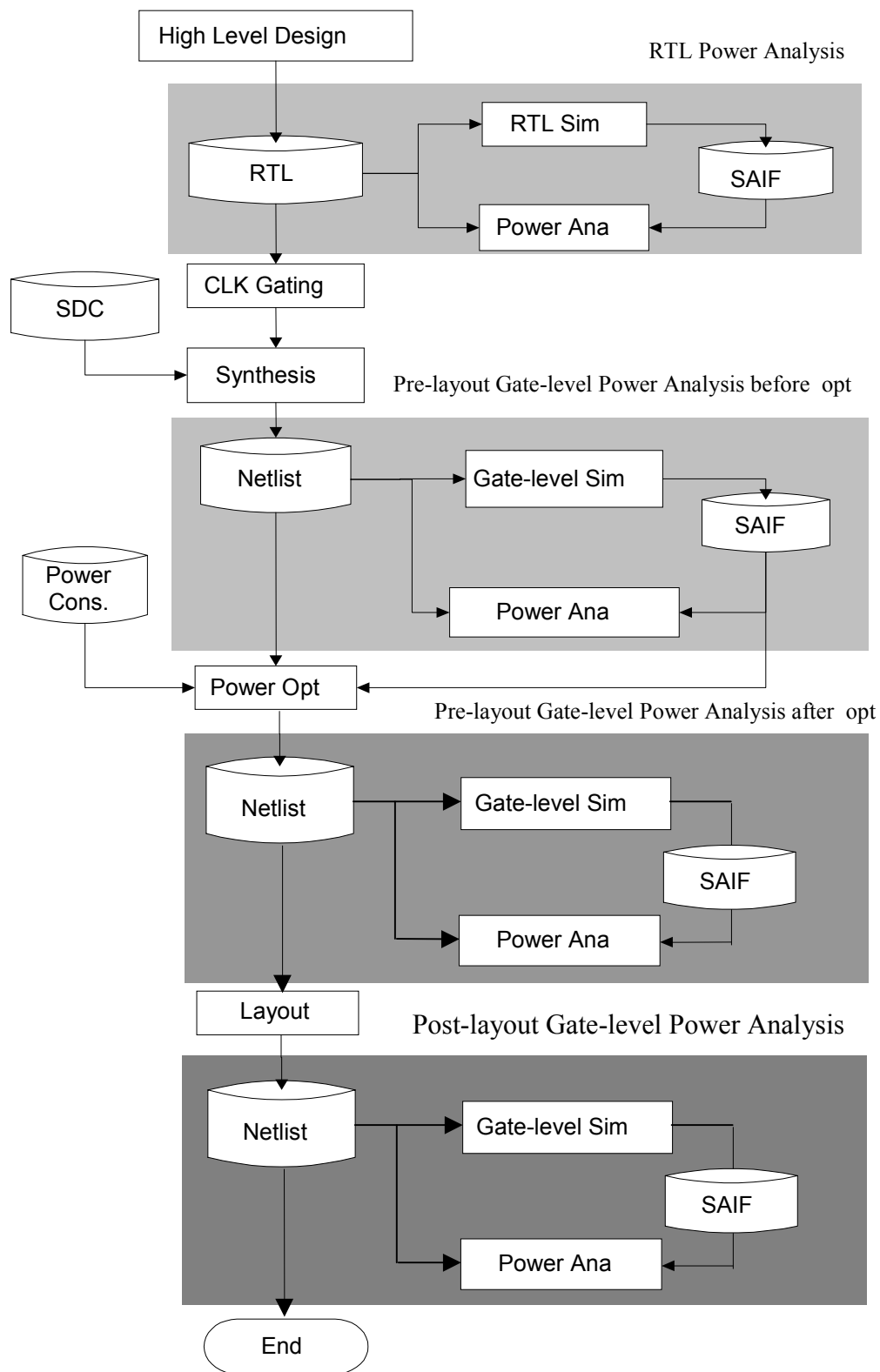


Figure 2: Low power design flow with Power Compiler

After clock gating, the synthesis for timing and area optimization is performed with design constraints applied in Design Compiler. The SAIF file can be generated from the simulation of this synthesized netlist. With this SAIF file and power constraints, the design can be synthesized to gate-level netlist optimized for power with Design Compiler

and Power Compiler working together, while won't make design's timing constraints and DRC worse.

Power analysis of gate-level design can be performed at several points in the methodology flow. For example, after annotating the switching activity coming from gate-level simulation, the power can be analyzed before optimizing the design with Power Compiler. This analysis provides an optional reference for the comparison after power optimization. A detailed power report of the power-optimized netlist will be obtained after power optimization with Power Compiler. After layout, power analysis can be performed with SAIF file coming from post-layout full-timing gate-level simulation. Thus the most accurate power analysis result can be obtained.

Our project is a chip used in wireless hand-held device, so low power consumption is an important target of our design. To achieve lower power design, we used Power Compiler in the following aspects:

- Performing power analysis at RTL and gate-level(pre-layout and post-layout)
- As guide architecture design for saving power
- Achieving power reduction by RTL clock-gating and gate-level power optimization

In this paper, based on the application of Power Compiler in our project, we will discuss the method and effects of power analysis, clock gating, gate-level power optimization in detailed.

2. Power analysis and accuracy

2.1.The calculation of power

To be aware of how accuracy of power analysis is, we should learn first how power consumption is calculated. The power dissipated in a circuit falls into two broad categories:

1. Static power
2. Dynamic power

Static Power Calculation

Static power is the power dissipated by a gate when it is not switching, that is, when it is inactive or static. Static power is dissipated in several ways. The largest percentage of static power results from source-to-drain sub-threshold leakage, which is caused by reduced threshold voltages that prevent the gate from completely turning off. Static power is also dissipated when current leaks between the diffusion layers and the substrate. For this reason, static power is often called leakage power. Power Compiler analysis computes the total leakage power of a design by summing the leakage power of the design's library cells, as shown in the following equation:

$$P_{\text{LeakageTotal}} = \sum_{\forall \text{cells}(i)} P_{\text{CellLeakage}_i}$$

Library developers annotate the library cells with the approximate total leakage power dissipated by each library cell—one leakage power number per library cell. For designs that are active most of the time, leakage power is less than 1 percent of the total power.

Dynamic power is the power dissipated when the circuit is active. A circuit is active anytime the voltage on a net changes due to some stimulus applied to the circuit. The dynamic power of a circuit is composed of two kinds of power:

1. Switching power
2. Internal power

Switching Power Calculation

The switching power of a driving cell is the power dissipated by the charging and discharging of the load capacitance at the output of the cell. Because such charging and discharging are the result of the logic transitions at the output of the cell, switching power increases as logic transitions increase. Therefore, the switching power of a cell is a function of both the total load capacitance at the cell output and the rate of logic transitions. Switching power comprises a large percent of the power dissipation of an active CMOS circuit. Power Compiler analysis calculates switching power (P_c) in the following way:

$$P_c = \frac{V_{dd}^2}{2} \sum_{\forall \text{nets}(i)} (C_{\text{Load}_i} \times TR_i)$$

C_{Load_i} is the total capacitive load of net i , including parasitic capacitance, gate capacitance, and drain capacitance of all the pins connected to net i . Power Compiler obtains C_{Load_i} from the wire load model for the net and from the technology library information for the gates connected to the net. We can also back-annotate capacitance information after physical design. TR_i is Toggle rate of net i , transitions per second. V_{dd} is Supply voltage.

Internal Power Calculation

Internal power is any power dissipated within the boundary of a cell. During switching, a circuit dissipates internal power by the charging or discharging of any existing capacitance internal to the cell. Internal power includes power dissipated by a momentary short circuit between the P and N transistors of a gate, called short-circuit power. When computing internal power, power analysis uses information characterized in the technology library. Power dissipation is dependent upon the power-supply voltage, frequency of operation, internal capacitance, and output load. In the library we used the power dissipated by each cell is:

$$P_{avg} = \sum_{n=1}^x (E_{in} \cdot f_{in}) + \sum_{n=1}^y \left(C_{on} \cdot V_{dd}^2 \cdot \frac{1}{2} f_{on} \right) + E_{os} \cdot f_{o1}$$

where:

P_{avg} = average power;

x = number of input pins;

E_{in} = energy per MHz associated with the n th input pin;

f_{in} = frequency at which the n th input pin changes state during the normal operation of the design (MHz);

y = number of output pins;

C_{on} = external capacitive loading on the n th output pin, including the capacitance of each input pin connected to the output driver, plus the route wire capacitance, actual or estimated (pF);

V_{dd} = operating voltage = 1.8V;

f_{on} = frequency at which the n th output pin changes state during the normal operation of the design (MHz);

E_{os} = energy per MHz associated with the output pin for sequential cells only.

The total average power for the design can be computed by adding the average power for each cell.

To illustrate the cause of power dissipated in a circuit, consider the simple gate shown in Figure 3. A rising signal is applied at IN. As the signal transitions from low to high, the N type transistor turns on and the P type transistor turns off. However, for a short time during signal transition, both the P and N type transistors can be on simultaneously. During this time, current I_{sc} flows from V_{dd} to GND, causing the dissipation of short-circuit power (P_{sc}). Other kinds of power, that is, leakage power and switching power is relatively easily understood from this figure.

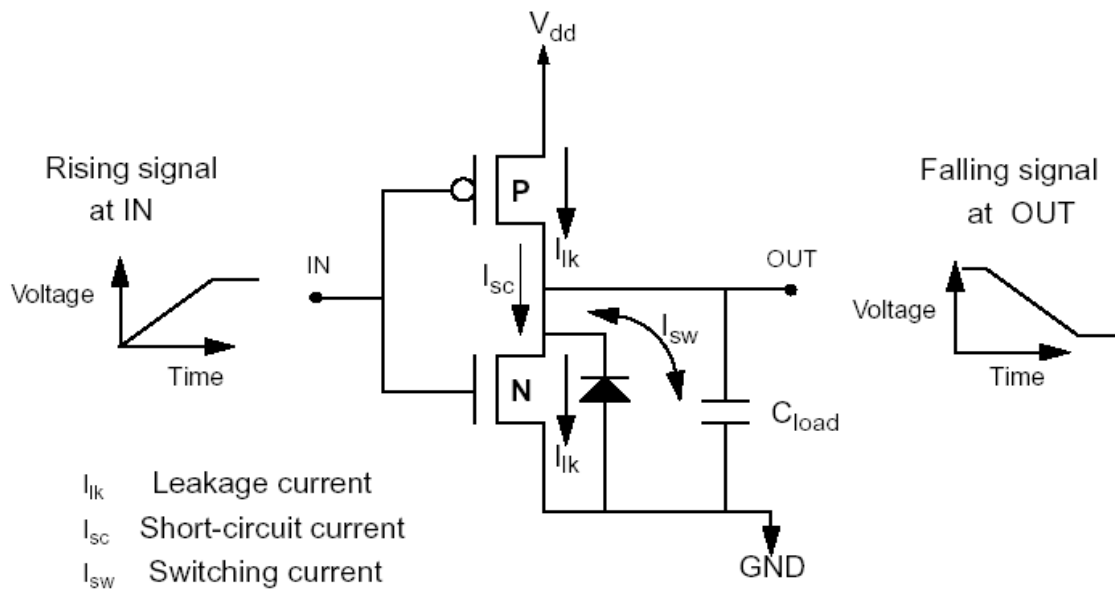


Figure 3: Components of Power Dissipation

2.2.The factors effecting the accuracy of power analysis

Switching Activity

Considering the equation for dynamic power, gate and net toggling clearly have direct effect on the dynamic power of a design. To report or optimize dynamic power, Synopsys tools need toggle information for the design. This information is called switching activity. The power tools of Synopsys model switching activity in terms of static probability and toggle rate (TR).

In our design flow shown in figure 2, there are three kinds of power analysis: RTL power analysis, gate-level power analysis before layout and gate-level power analysis after layout. RTL power analysis is performed using the Switching Activity file from RTL simulation, which just captures switching activity for synthesis invariant signals. When performing power analysis, Power Compiler analysis runs zero-delay simulation to propagate computed switching activity through the rest of the design before completing power analysis. A clock must be specified for this power analysis. So RTL power accuracy just has relatively good accuracy. For gate-level power analysis before layout, the SAIF file from zero-delay gate-level simulation is used. This SAIF file can contain switching activity for all nets in the design and considers state and path dependencies, so the accuracy is increased. For gate-level power analysis after layout, the design is completely annotated with switching activity from full-timing gate-level simulation (gate-level simulation with annotated SDF file from layout). Annotation of full-timing gate-level switching activity can include state- and path-dependent switching activity for cells that have such dependencies modeled in the library as well as glitch information. So Power Compiler analysis at this stage has the highest accuracy. The following table compares the switching activity captured in three kinds of simulation.

Table 1: Comparing Methods of Capturing Switching Activity

	RTL Simulation	Zero-time Gate-level Simulation	Full-time Gate-Level Simulation
Synthesis Invariant	Yes	Yes	Yes
Internal Node	No	Yes	Yes
State- & Path- Dependency	No	Yes	Yes
Glitch	No	No	Yes

Wire Load

From the equations for dynamic power, we can see net capacitance also has direct influence on power calculation. Power analysis uses any back-annotated net loads during the power calculation. For nets that do not have back-annotated capacitance, Power Compiler estimates the net load from the appropriate wire load model from the technology library. So for pre-layout power analysis, the more accurate the WLM is, the higher accuracy of power analysis is. Just as for timing and area optimization, CWLM is better than Non-Customed WLM for power analysis. After layout, we can increase accuracy further by annotating wire loads from layout.

Testbench

Because switching activity comes from RTL or gate-level simulation, so testbench has indirect influence on power analysis. The following figure shows the relationship of calculated power consumption value and simulation run time. To verify our design thoroughly, we may develop many test cases to verify all aspects of the design's function. Many of these test cases may not mimic the real working situation of the design. So get switching activity from running all of test cases in verification plan may be not most accurate and will cost too long run time. We suggest strongly develop dedicated test case specially for power analysis which models the average behavior of the design as closely as possible.

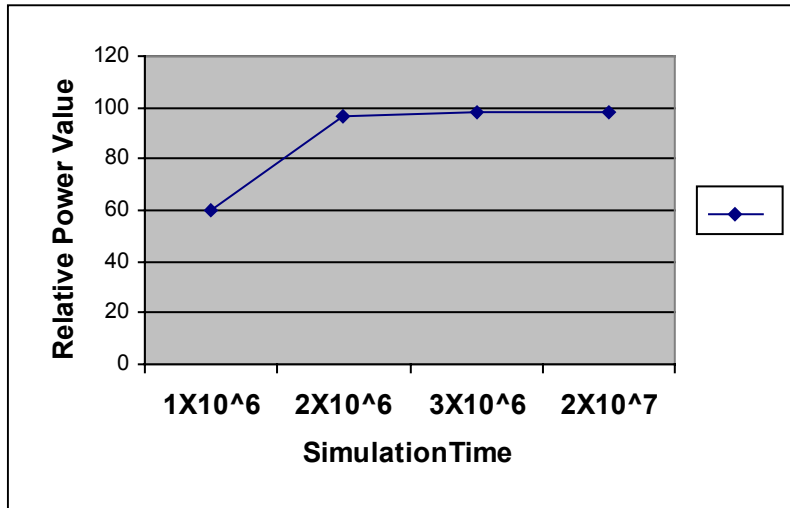


Figure 4: The influence of simulation run time on power analysis

Power consumption by CTS

Clock tree is expanded only after layout. In pre-layout synthesis, clock is considered as an ideal signal. So the power dissipated by buffers in clock tree is not included in pre-layout gate-level power analysis. In our design after layout, clock tree will consume about 14% of total power.

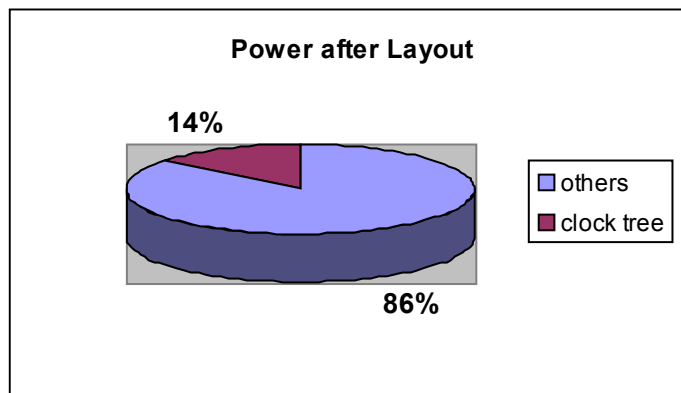


Figure 5: Power consumed by clock tree

2.3.Results of power analysis

Figure 6 shows the relevancy of power consumption of all modules in our design. This information is helpful for us to refine design architecture to reduce power dissipation at high level. Figure 7 shows the design architecture and detail of power consumption. The 89% of total power are consumed by Mod_a, Mod_b1, Mod_b2, Mod_c and Mod_d. Mod_c consumes the most of power in the design. From this information, we can focus our effort on reduce the frequency and bit-width of signals entering these modules. When perform design architecture tradeoff, Mod_c should be ranked No.1. We should reduce power consumed by Mod_c firstly.

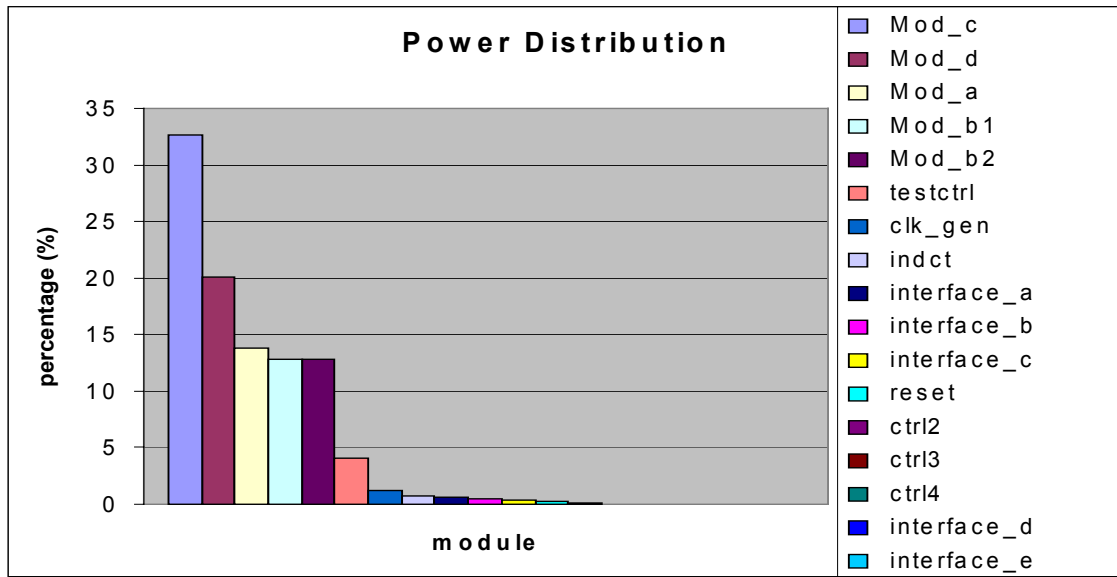


Figure 6: Result of RTL power analysis

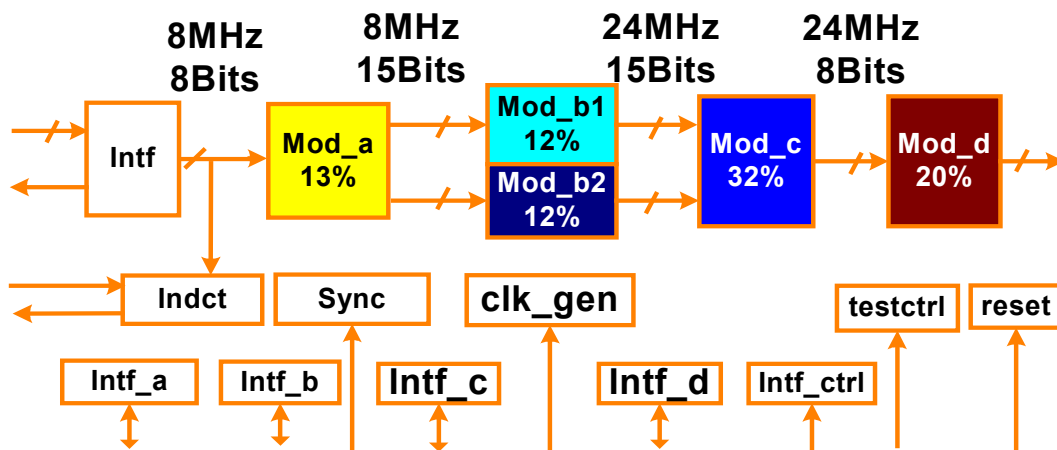


Figure 7: Design architecture and detail of power consumption

Figure 8 is block diagram of the design after architecture change. The frequency and bit-width of signals entering Mod_c are reduced. Mod_b2 is unnecessary and is deleted. The inserted module Mod_e is a small one, just consume a little power.

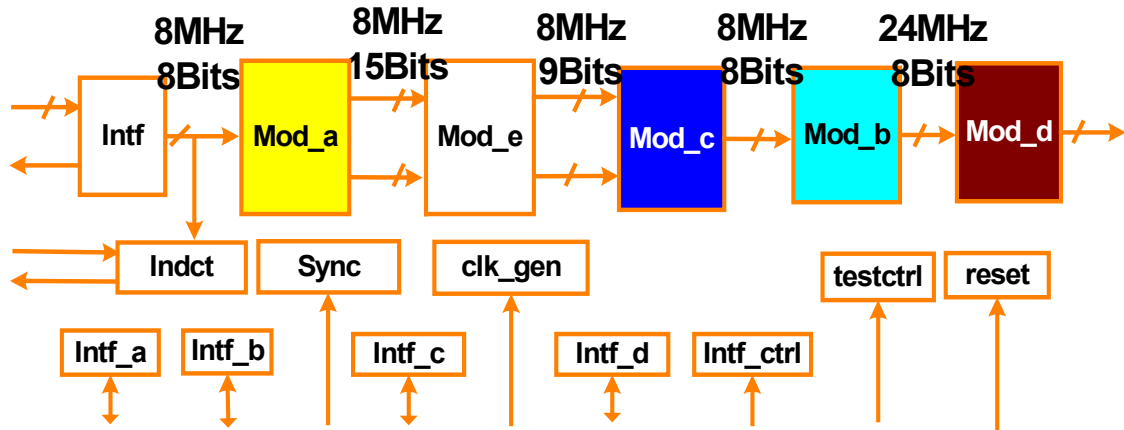


Figure 8: Design architecture change for power reduction

Figure 9 compares the power consumption of RTL power analysis and gate-level power analysis, from which we can see the power of RTL power analysis is about two times of that of gate-level power analysis, the difference is big. So we may not believe the absolute value of RTL power analysis, but as stated above, quickly capturing information of RTL power analysis is helpful.

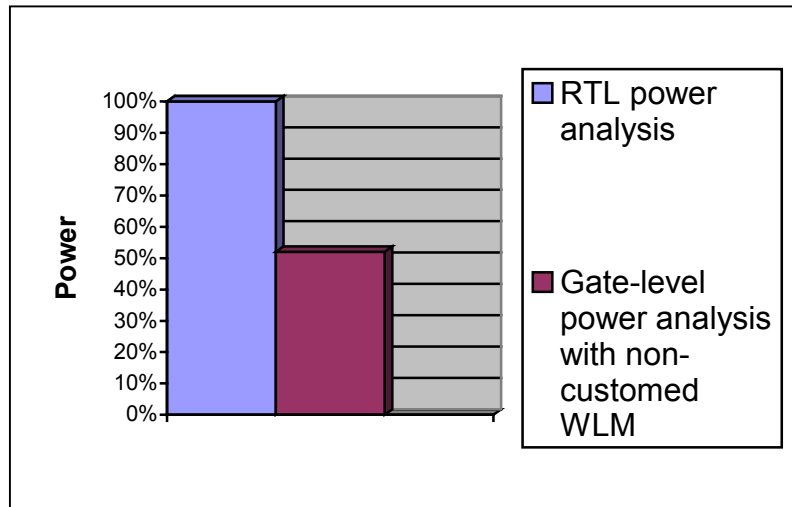


Figure 9: Comparison of RTL power analysis and gate-level power analysis

Figure 10 illustrates the difference resulted by non-Customed WLM, Customed WLM and back-annotated net capacitance from layout. Power analysis of the three kinds uses the same netlist – netlist after layout. As described in above section, power value after layout is most accurate. From this figure we can see power value from gate-level power analysis with CWLM is 95% of that after layout, while gate-level power analysis with non-Customed WLM is 133% of that after layout. CWLM is more accurate for power analysis.

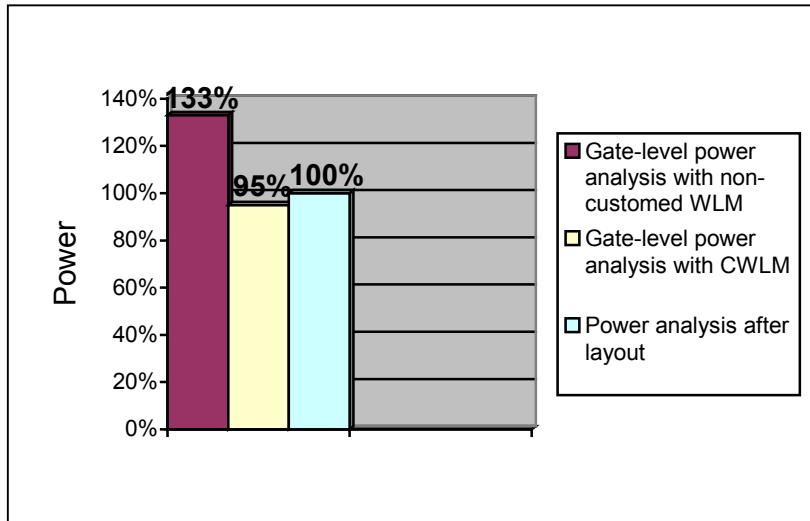


Figure 10: Comparison of power consumption

Figure 11 Compares power of pre-layout using pre-layout netlist and power of post-layout. Pre-layout power analysis uses non-customed WLM. The total difference is 20%. There are three main factors cause the difference between pre-layout power analysis and post-layout power analysis: expansion of clock tree, back-annotated wire load, accuracy of switching activity.

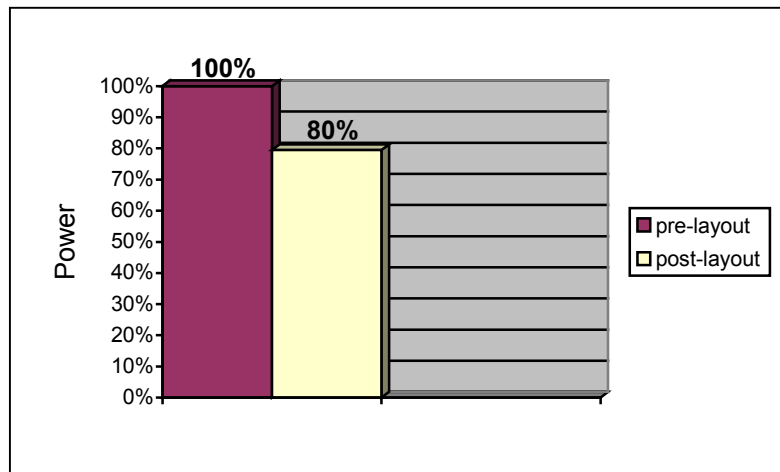


Figure 11: pre-layout power analysis vs. post-layout power analysis

3. RTL clock gating

Because for cells in the circuit, voltage on an input net can change without necessarily resulting in a logic transition on the output, dynamic power can be dissipated even when an output net doesn't change its logic state. In such condition, by gating clock of register banks, power can be saved.

3.1. Power Compiler vs. Design Compiler

With the capability of gating clock, Power Compiler can implement circuit in a different way with Design Compiler. For such kinds of HDL code as following:

```
@posedge (CLK)
begin
  if(EN == 1)
    Q = Data_in;
  end
end
```

Without clock gating, Design Compiler implements register banks by using a feedback loop and a multiplexer. When such registers maintain the same value through multiple cycles, they unnecessarily use power. Figure 12 shows the circuit implemented by Design Compiler.

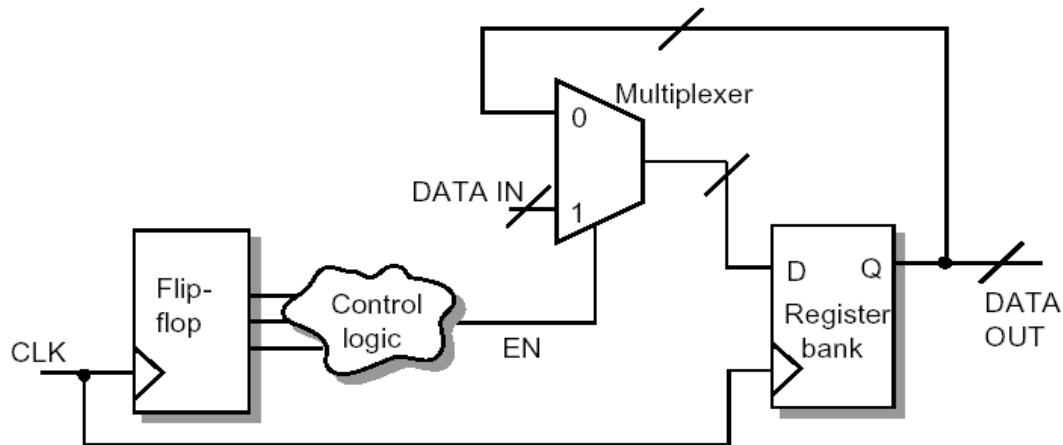


Figure 12: Design Compiler synthesis result

By controlling the clock signal for the register bank, we can eliminate the need for reloading the same value in the register through multiple clock cycles. Clock gating inserts clock-gating circuitry into the register bank's clock network, creating the control to eliminate unnecessary register activity. Figure 13 shows the implementation by Power Compiler. When elaborating the design by using the `-gate_clock` option, several kinds of clock gating circuitry can automatically be created and inserted in clock network. Clock gating reduces the clock network power dissipation, relaxes the data path timing, and reduces routing congestion by eliminating feedback multiplexer loops. For designs that have large multi-bit registers, clock gating can save power and reduce the number of

gates in the design. However, for smaller register banks, the overhead of adding logic to the clock tree might not compare favorably to the power saved by eliminating a few of feedback nets and multiplexers. The latch prevents glitches on the EN signal from propagating to the register's clock pin. When the CLK input of the 2-input AND gate is at logic state 1, any glitching of the EN signal could, without the latch, propagate and corrupt the register clock signal.

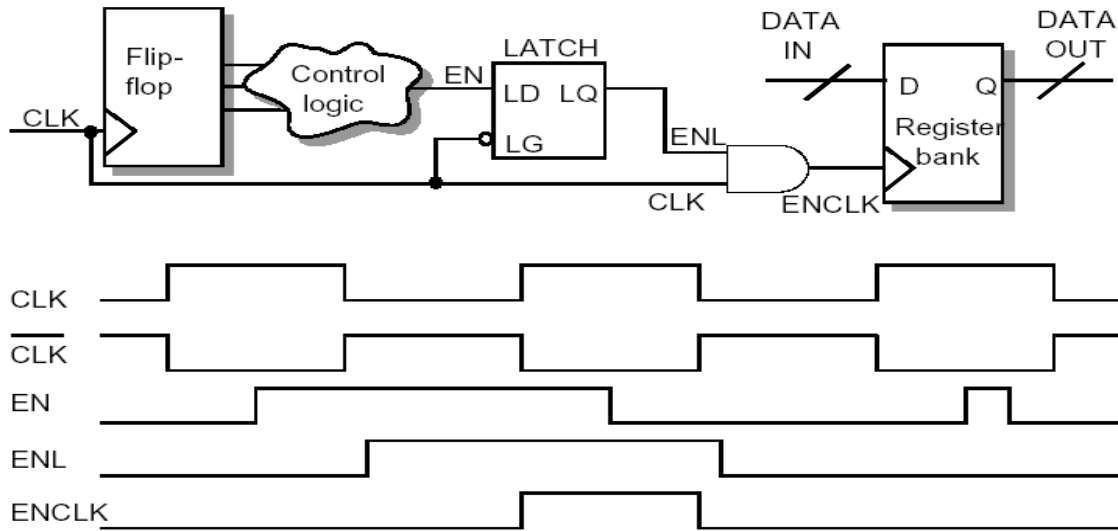


Figure 13: Latch-Based Clock Gating circuitry implemented by Power Compiler

3.2. DFT issue related with clock gating

Gating the register's clock makes it uncontrollable for test (if there is no dedicated scan clock). A control point can be added by Power Compiler to increase the testability of the design by restoring the clock signal to its non-gated form during test. Figure 14 shows a kind of solution for latch-based clock gating circuitry. The control point is a scan enable signal implemented using an OR gate that eliminates the function of the clock gate during

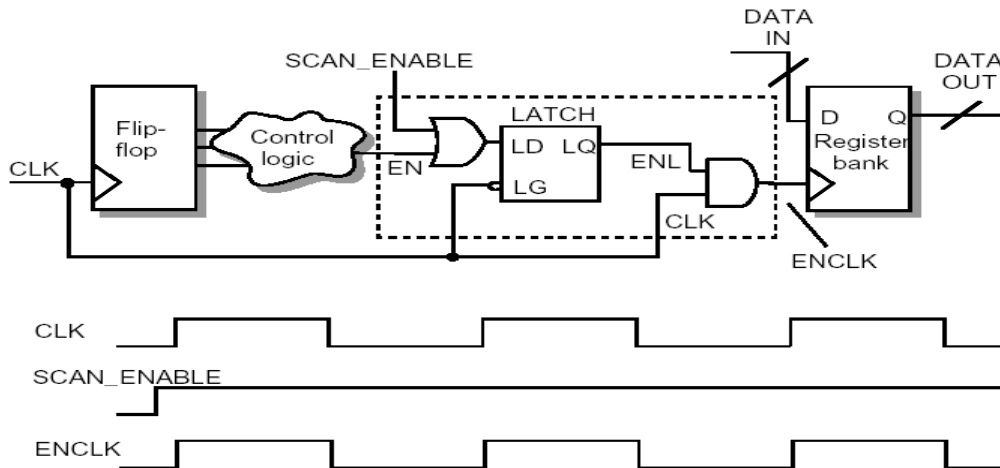


Figure 14: Inserting scan_enable signal as test control point to improve testability

test, which restores the controllability of the clock signal. This kind of test solution has fault coverage comparable to that of a design without gating clock.

To connect the test ports through all the levels of hierarchy in a design and also to the test pin on the clock-gating cells, use the `hookup_testports` command. This command connects the test ports on all levels of the design hierarchy to the `scan_enable` pins of the OR gate in the clock-gating logic. This command should be used on the top level of the design after all lower levels have been elaborated. If a `scan_enable` port does not exist at a higher level of hierarchy, it will be created. If it exists, it can be used. To designate an existing test port we can use the `clock_gate_test_pin` attribute. In this case, it must be labeled as a `scan_enable` signal using the `set_signal_type` and `set_attribute` commands. Whenever inserting scan chain in any hierarchical, this `scan_en` signal must be identified by command `set_scan_signal -type scan_enable`. Thus this `scan_en` signal will be connected to the scan enable pin of scan registers (red line in figure 15). Figure 15 elaborates this scene.

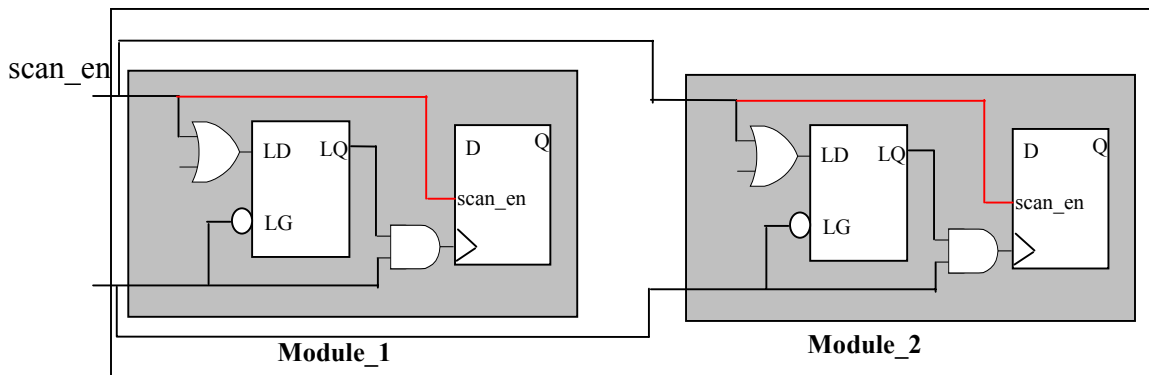


Figure 15: Inserting a Control Point for Testability in a Hierarchical Design

3.3. Timing issue related with clock gating

To maintain the integrity of gated clock signal, gating signal can't change during active clock pulse. When using `set_clock_gating_style` command, we can use the following options:

`-setup setup_value`

`-hold hold_value`

to set the setup and hold time constraints on the clock-gating element. Use `propagate_constraints` command to propagate constraints from the clock-gating element to the current design before compiling the design. The `propagate_constraints` command traverses the hierarchy of the current design, searching for setup- and hold-time constraints on clock-gated sub-designs. The command propagates the setup- and hold-time constraints from the newly created clock-gated sub-designs upward in the design hierarchy. Figure 16 shows the meaning of setup time and hold time for clock gating element. When using PrimeTime for static timing-analysis, use the command `set_clock_gating_check -setup -hold`

to change the setup and hold values for the gating check. PrimeTime performs clock-gating checks on all gated clocks using 0.0 as the default for setup and hold.

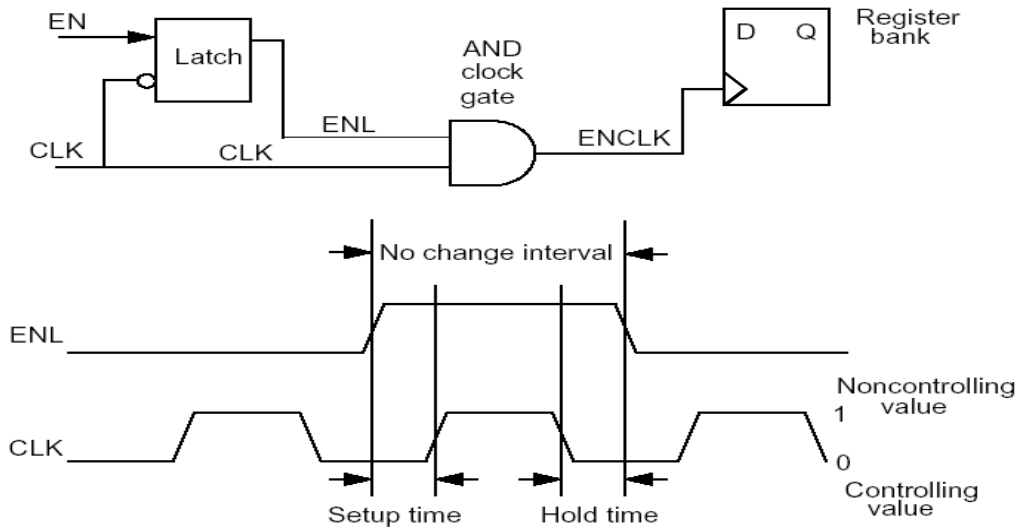
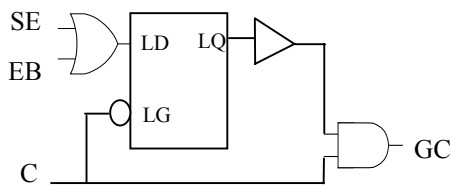


Figure 16: setup time and hold time for clock gating element

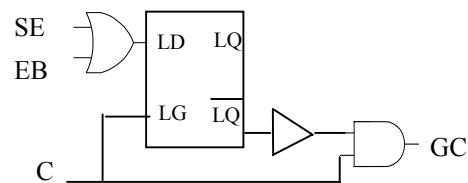
Integrated Clock-Gating (ICG) Cell

An integrated clock-gating cell integrates the various combinational and sequential elements of a clock gate into a single cell located in the technology library. Using an integrated clock-gating cell can alleviate timing and DFT problems. The integrated clock-gating cell setup time and hold time are specified in the technology library and cannot be overridden by `-setup` and `-hold` options of the `set_clock_gating_style` command. By special layout for these ICG cells, we can eliminate the possibility of generating glitching at gated clock. For ICG cells `clock_gate` attributes on both the cell and on the pins are also specified in the technology library. When perform clock gating and `check_test` these attributes can be recognized. ICG cell in the technology library include the following attributes:

- `clock_gating_integrated_cell`
- `clock_gate_test_pin`
- `clock_gate_enable_pin`
- `clock_gate_out_pin`
- `clock_gate_clock_pin`



(a)



(b)

Figure 17: Integrated Clock-Gating (ICG) Cell

Figure 17 is two kinds of ICG cell used in our design. Figure 17(a) is used for posedge clock gating, Figure 17(b) is used for negedge clock gating. In Figure 17(a), the delay of timing arc C-LG-LQ-GC is bigger than that of timing arc C-GC, thus no glitch will be generated at GC. This is the same for Figure 17(b).

3.4. Results of clock gating

In our design case, we use two kinds of ICG cells shown in figure 14, set `minium_bitwidth` as default value — 3, set `max_fanout` as 12. Totally 78 integrated clock gating cells are inserted in our design and 643 registers are clock-gated.

Figure 18 shows the result of clock-gating method. With clock gating, power consumption of 23% is reduced, design area is reduced by 8%, while ATPG fault coverage just decrease by 0.8%. The reason of area reduction is that many multiplexers are replaced by 78 ICG cells. The additional runtime for gating clock buffer insertion is about 1 minutes per 10K gates for our design case.

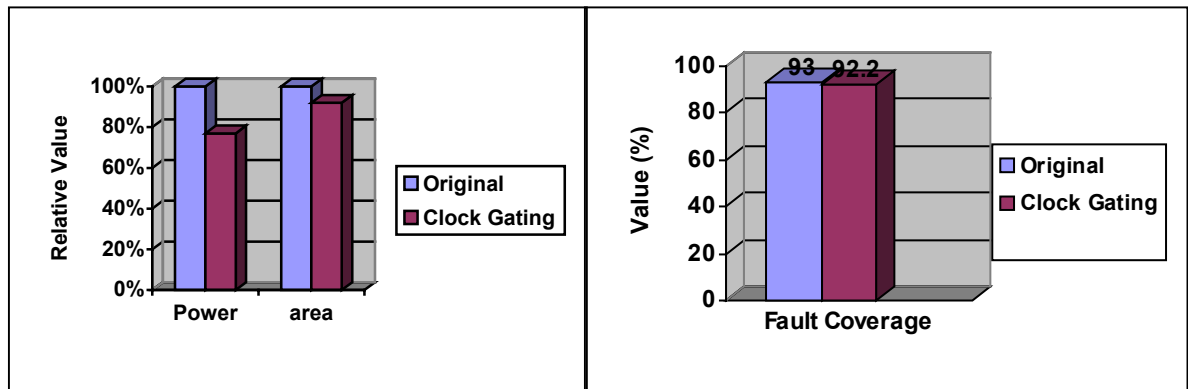


Figure 18: Effect of clock gating

4. Gate-level power optimization

By back-annotating switching activity and setting power constraints using commands `set_max_dynamic_power` and `set_max_leakage_power`, Power Compiler can perform additional steps to optimize the design for dynamic and static power in `dc_shell` environment. The techniques used by gate-level power optimization are technology mapping, cell sizing, buffer insertion, phase assignment, pin swapping, factoring, etc.

4.1. Factors effecting gate-level power optimization

The quality of results obtained with Power Compiler depend on the following factors:

- Optimization priority
- Accuracy of back-annotated SAIF
- Diversity of cell libraries
- Ratio of sequential to combinational logic

Working within the Design Compiler shell, Power Compiler can optimize simultaneously for timing, power, and area. Optimization occurs according to the following default priorities:

1. Design rule constraints
2. Timing
3. Dynamic power
4. Leakage power
5. Area

The optimization priority ensures that Power Compiler, for example, never violates a timing constraint in order to optimize for power.

Power compiler's optimization for power is based on back-annotating switching activity information. Simulation vectors that model the average behavior of the design as closely as possible will increase the accuracy of switching activity and allows Power Compiler to calculate an accurate cost function, thus generate better optimized result. So although back-annotated switching activity can come from gate-level simulation or RTL simulation, switching activity coming from gate-level simulation will generate better result. Of course, testbench is also important because switching activity coming from simulation of testbench.

Power Compiler can use positive timing slack to decrease the design power. For example, Power Compiler can size-down cells to reduce power with path delay increased. The more positive slack that exists, the more delay Power Compiler can accept in making choices for low-power cells. Designs with excessively restrictive timing constraints have little or no positive slack to trade for power reduction. So don't over-constrain the design.

For most of standard cell libraries, different sizes of sequential cells often have less variation in power than different sizes of combinational cells. Table 2 shows power information of two kinds of cell in our library. One is combinational cell And-Invert-Or gate(AIO), the other is sequential cell D-type Flip Flop(DFF). We can see that power consumption of DFF with 4X drive ability is about 3 times of that of the cell with XL drive ability. While for AOI, the difference is 4.5 time. Therefore, designs with large amounts of sequential logic might experience smaller power reductions than designs with less sequential logic. This can be seen from our optimization report in figure 19. The decrease of power in sequential logic is much less than that in combinational logic.

Table 2: Power information of library cells

AOI22						DFF					
Pin	Driver ability				X4/XL	Pin	Driver ability				X4/XL
	XL	X1	X2	X4			XL	X1	X2	X4	
A0	0.0174	0.0233	0.0441	0.0821	4.72	D	0.0327	0.0288	0.0371	0.0583	1.78
A1	0.0217	0.0285	0.0551	0.1008	4.65	CK	0.0309	0.0302	0.0363	0.0516	1.67
B0	0.0235	0.0317	0.0594	0.1108	4.71	Q	0.0328	0.0387	0.0622	0.0989	3.01
B1	0.0280	0.0367	0.0708	0.1303	4.63						

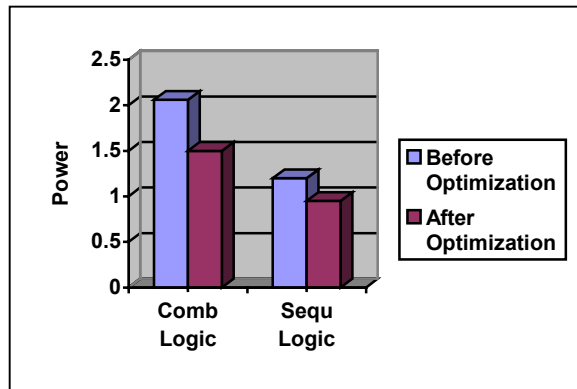


Figure 19: Power reduction for combinational logic and sequential logic

Some gate-level power optimization techniques, such as technology mapping, cell sizing and so on, need change cells with the same function but different physical information. If there are many cells with different power information to choose from the library, Power Compiler can generate good optimized result. Usually in order to have a good result for timing and area, some cells with low driver ability will be set don't_use attribute, which makes these cells excluded from synthesis. With a smaller range of power choices, Power Compiler has less flexibility in reducing power. So for the purpose of lower power design, we should avoid to set don't_use attribute on library cells, especially for those low power cells.

4.2. Results of gate-level power optimization

Figure 17 shows the effect of gate-level power optimization. Comparing with pre-optimization, power consumption is reduced by 15%, at the same time area is reduced by 6%. Timing and ATPG fault coverage are not effected. The run time of power calculation and report is about 30 minutes per 10K gates for our design case.

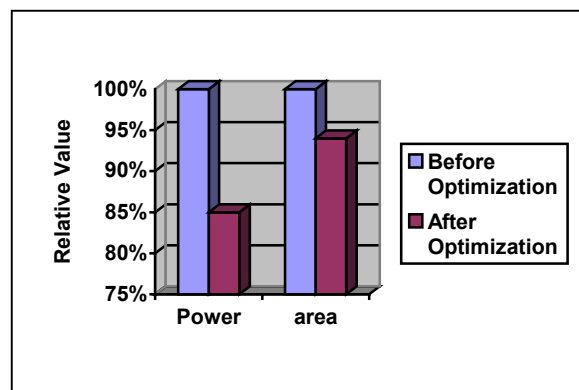


Figure 20: Effect of gate-level power optimization

5. Summary

Figure 21 shows the total effect of RTL clock gating and gate-level power optimization.

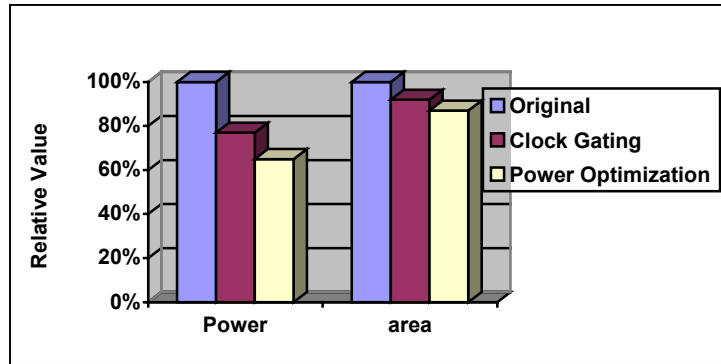


Figure 18: Total effect of power optimization

For our design case, with Power Compiler we can achieve totally 36 % power reduction and 12% area reduction. By using Integrated Clock Gating cells, no timing impact is needed to worry about. The design methodology using Power Compiler is Seamless: RTL modification is not needed. The overhead of run time is acceptable.

6. References

- [1] SYNOPSYS, "Power Compiler Reference Manual".
- [2] ARTISAN, "Chartered Semiconductor Standard Cell Library Databook".