Power Reduction Through RTL Clock Gating

By
Frank Emnett and Mark Biegel

Automotive Integrated Electronics Corporation

ABSTRACT

This paper describes a design methodology for reducing ASIC power consumption through use of the RTL clock gating feature in Synopsys Power Compiler. This feature causes inactive clocked elements to have clock gating logic automatically inserted which reduces power consumption on those elements to zero when the values stored by those elements are not changing. The RTL clock gating feature allows easily configurable, automatically implemented clock gating which allows maximal reduction in power requirements with minimal designer involvement and no software involvement. The paper also discusses the integration of RTL clock gating with full scan techniques, allowing designs to be both low-power and fully testable. The methodology was proven in a 200K-gate ASIC, which implemented full scan testing and used RTL clock gating to reduce its power consumption by two-thirds.

1.0 Introduction

This paper describes our experience with the RTL clock gating feature of Synopsys Power Compiler. The chip in question is a specialized microcontroller peripheral ASIC designed for performing real-time control of an internal combustion engine. The design was originally scoped at around 60K gates with a 50mA dynamic current specification. Over the course of the program development, features and capacity were added to the ASIC until the gate count increased to 200K gates. The technology for the design was restricted to a 0.5µm technology due to a 5 V power supply and I/O voltage specification. The power budget for the chip could be increased to only 100 mA without a redesign of the power supply. In addition, the chip was to be in an engine control module mounted in the engine compartment of a vehicle, with an ambient temperature specification of 125°C and no active cooling. The materials used in the chip packaging required a junction temperature of no more than 150°C. The module specification required that we assume that no heat was dissipated from the device through convection; only through conduction through the package pins. This further limited our temperature rise above the board temperature to approximately 10°C, which required that we dissipate no more than 500mW under normal operating conditions. With no clock gating and limited power management on the internal dual port RAM's, the first pass of silicon had a dynamic IDD consumption of 280 mA, which resulted in a junction temperature of approximately 155°C. We needed to find a way to reduce the dynamic current consumption of the part to less than 100 mA. After investigation of possible solutions, we settled on two approaches: active power management of the dual port RAM cells, and use of the RTL clock gating feature in Synopsys Power Compiler.

2.0 First Steps

After receiving the dynamic IDD consumptions reports on the first pass of the design, we performed a detailed analysis of the design's power consumption (see Table 1). This design incorporated seven dual ported RAM cells. These cells consume considerable power while enabled. It turned out that one of the data ports on four of the RAM cells (which all performed a similar function) were enabled all the time, while there were actually long periods of time during which these ports could be disabled. Adding logic to capture the read data from these ports and then disable the RAM when accesses were not necessary reduced dynamic IDD by 64 mA, bringing the total dynamic IDD down to 216 mA, still far above the required value. While searching for additional ways to reduce power consumption, we encountered the RTL Clock Gating feature supported by Power Compiler.

COMPONENT	CURRENT COMSUMPTION
Clock tree	36 mA
RAMs	64 mA
Flip-Flops	170 mA
Combinatorial Circuitry	10 mA
Total	**280 mA**

Table 1 Current consumption analysis of first pass design

3.0 RTL Clock Gating

In the traditional synchronous design style used for most HDL and synthesis-based designs, the system clock is connected to the clock pin on every flip-flop in the design. This results in three major components of power consumption: 1) power consumed by combinatorial logic whose values are changing on each clock edge; 2) power consumed by flip-flops (this has a non-zero value even if the inputs to the flip-flops, and therefore, the internal state of the flip-flops, is not changing); and 3) the power consumed by the clock buffer tree in the design. The design controls a mechanical system (an internal combustion engine) in which control and sense signals have data rates in the 1 KHz to 1 MHz range, far below the 32 MHz system clock speed. Therefore, the first component of the power consumption due to combinatorial logic was by far the smallest contributor to the total power consumption. Because of this, strategies such as gate resizing for power reduction (also offered by Synopsys Power Compiler) were not used. On the other hand, RTL clock gating had the potential of reducing both the power consumed by flip-flops and the power consumed by the clock distribution network.

RTL clock gating works by identifying groups of flip-flops which share a common enable term (a term which determines that a new value will be clocked into the flip-flops). Traditional methodologies use this enable term to control the select on a multiplexer connected to the D port of the flip-flop or to control the clock enable pin on a flip-flop with clock enable capabilities. RTL clock gating uses this enable term to control a clock gating circuit which is connected to the clock ports of all of the flip-flops with the common enable term. Therefore, if a bank of flip-flops which share a common enable term have RTL clock gating implemented, the flip-flops will consume zero dynamic power as long as this enable term is false. The Verilog code in Example 1 is an RTL description of a three-bit up counter.

```
module counter (CLK, RESET_, INC, COUNT);
  input CLK;
  input RESET_;
  input INC;
  output [2:0] COUNT;

  reg [2:0] COUNT;

  always @(posedge CLK or negedge RSTN)
    if (~RESET_)
      COUNT <= #1 3'b0;
    else if (INC)
      COUNT <= #1 COUNT + 1;
endmodule
```

Example 1 - RTL code for three-bit up counter

The counter has an asynchronous reset whenever *RESET_* is asserted low, increments on every clock cycle when *INC* is asserted high, and holds the old value when *INC* is deasserted low. We see the traditional implementation without clock gating in Figure 1. Note that the clock is routed directly to each of the flip-flops in the design, which means that they will be clocked continuously, with the old data recirculated into the flip-flops through the multiplexers on the flop inputs, when the *INC* input is low.

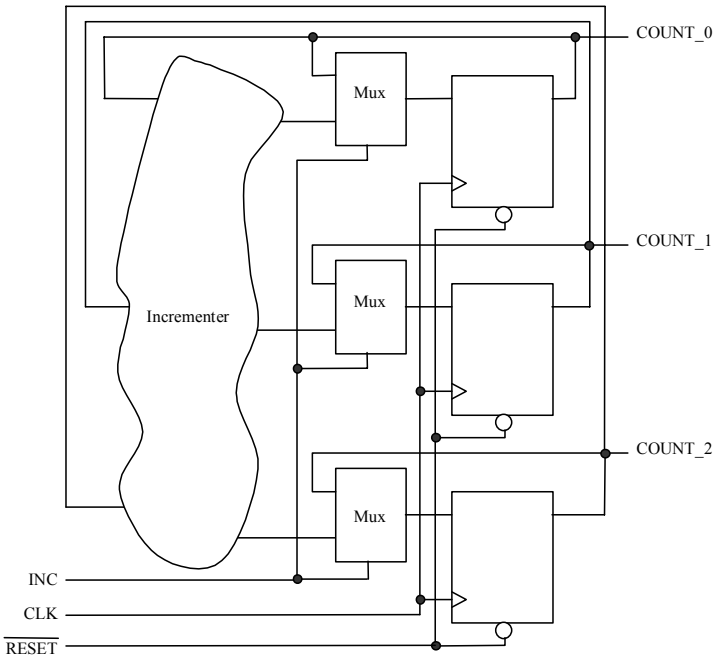

Figure 1 – Three Bit Counter Traditional Implementation

In Figure 2, the same circuit is implemented with clock gating. The circuit is similar to the traditional implementation except that a clock gating element has been inserted into the clock network, which causes the flip-flops to be clocked only when the *INC* input is high. When the *INC* input is low, the flip-flops are not clocked and therefore retain the old data just as in the original implementation. This allows the three multiplexers in front of the flip-flops to be removed, which can result in significant area savings when wide banks of registers are being implemented.

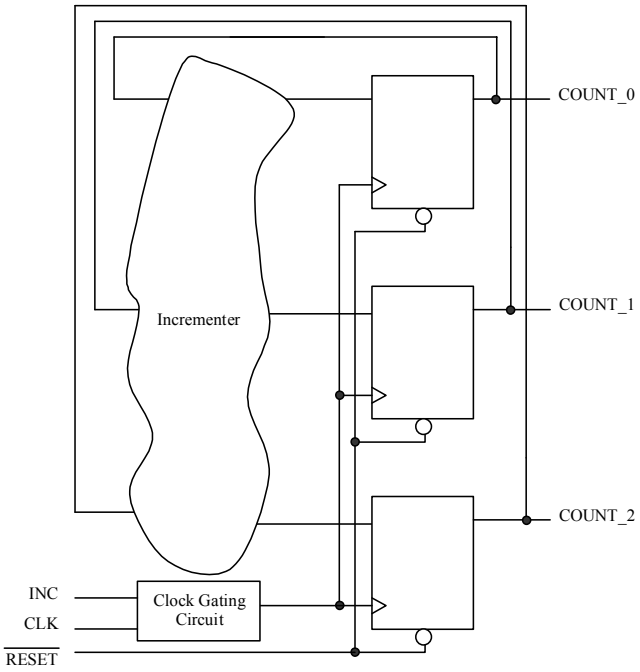

Figure 2 – Three Bit Counter with Clock Gating

Enabling RTL clock gating in a design requires only two modifications to the standard synthesis flow: using the `set_clock_gating_style` command to select parameters for determining when to use clock gating and the particular type of clock gating desired, and using the `–gate_clock` option to the elaborate command to instantiate the clock gating circuitry. Example 2 shows a basic synthesis script with these changes applied.

```
set_clock_gating_style –sequential_cell latch
  -positive_edge_logic {and} –negative_edge_logic {or}
analyze counter.v
elaborate counter –gate_clock
current_design counter
/* apply necessary timing constraints */
propagate_constraints –gate_clock
compile
```

Example 2 - Script for Basic Clock Gating

The set_clock_gating_style command has many options. Perhaps the most important option is the choice of latch-based or latch-free clock gating styles. The latch-free clock gating style (see Figure 3) uses a simple AND or OR gate (depending on the edge on which flip-flops are triggered), and imposes a requirement on the circuit that all enable signals be held constant from the active (rising) edge of the clock until the inactive (falling) edge of the clock (see Figure 4), in order to avoid truncating the generated clock pulse prematurely or generating multiple clock pulses where one is required.

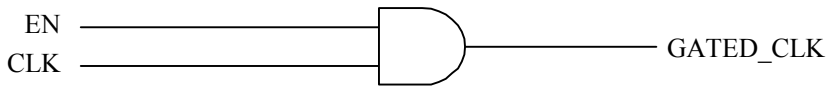

Figure 3 - Latch Free Clock Gating Circuit

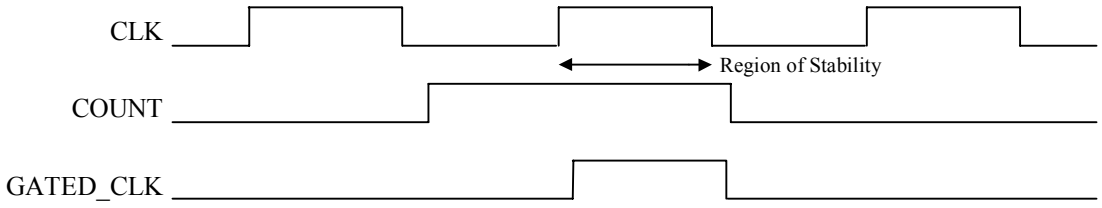

Figure 4 - Operation of Latch Free Clock Gating Circuit

This restriction makes the latch-free clock gating style inappropriate for our single-clock flip-flop based design. The latch-based clock gating style adds a level-sensitive latch to the design to hold the enable signal from the active edge of the clock until the inactive edge of the clock, making it unnecessary for the circuit to itself enforce that requirement (Figures 5). Since the latch captures the state of the enable signal and holds it until the complete clock pulse has been generated, the enable signal need only be stable around the rising edge of the clock, just as in the traditional ungated design style (see Figure 6). Apart from the latch-free vs. latch-based style issue, for our application, the defaults usually proved to be sufficient. By default, Power Compiler inserts clock gating circuits for banks of registers with common enable terms of 3 bits or more in width.

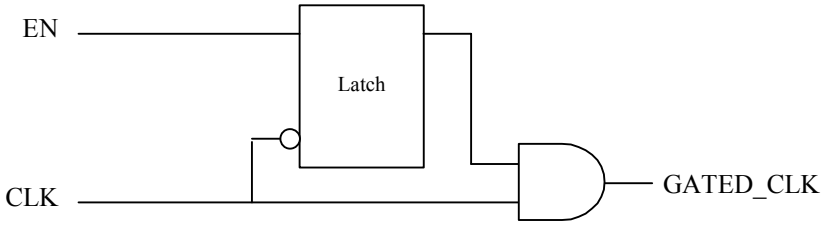

Figure 5 - Latch Based Clock Gating Circuit

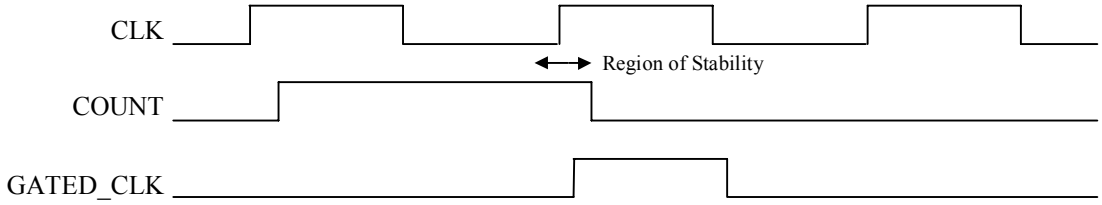

Figure 6 - Operation of Latch-Based Clock Gating Circuit

4.0 Interaction of RTL Clock Gating with DFT

Due to the complexity, high volume, and high quality requirements for this automotive application (every field return must have a full failure analysis performed on it), we elected to use a full scan multiplexed flip-flop DFT methodology to ensure high manufacturing fault coverage. We initially had concerns regarding the effect of RTL clock gating on the operation of scan insertion and ATPG, as the full scan technique traditionally assumes a synchronous design style in which all flip-flops are clocked directly from the system clock.

Power Compiler offers several options for controllability and observability enhancements to the basic clock gating circuit to enable its use within a scan methodology. A controllability signal which causes all flip-flops in the design to be clocked, regardless of the enable term value, can be added to allow the scan chain to shift information normally. This signal can be ORed in with the enable term either before (specified with the `-control_point before` option to the `set_clock_gating_style` command) or after (`-control_point after`) the latch, and can be connected to either a test mode enable signal (specified using `-control_signal test_enable`) which is asserted throughout scan testing or to a scan enable signal (`-control_signal scan_enable`) which is asserted only during scan shifting. The combination of parameters `-control_point before -control_signal scan_enable` provides the best testability of the clock gating circuits. A basic script using this configuration appears in Example 3.

```
set_clock_gating_style -sequential_cell latch
  -positive_edge_logic {and} -negative_edge_logic {or}
  -control_point before -control_signal scan_enable
analyze hdl.v /* hdl.v needs test_se hack for v1999.05 and
  previous */
elaborate top -gate_clock
current_design counter
hookup_testports /*valid only for v1999.10 and newer*/
/* apply necessary timing constraints */
propagate_constraints -gate_clock
compile -scan
/* perform check_test, scan chain routing, etc. */
```

Example 3 - Script for clock gating with scan insertion

The clock gating circuit constructed by using this configuration appears in Figure 7. The other options are provided as workarounds if the test generation tool cannot handle this optimal configuration. When using this configuration, testability is essentially the same as a design without clock gating.

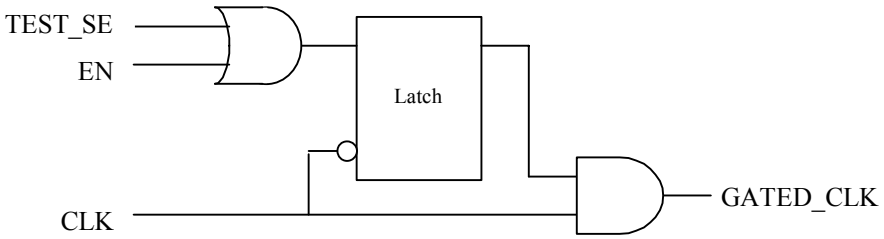

Figure 7 - Clock Gating Circuit with Optimal Scan Control Circuit

Unfortunately, for software versions 1999.05 and previous, RTL clock gating and DFT do not coexist peacefully. When using the set_clock_gating_style options –control_signal scan_enable, the software creates a *test_se* port on the design and routes it to the similarly named port on all of the clock gating circuits. Unfortunately, when the design is specified using implicit notation for port connections, these *test_se* ports are not always inserted when necessary in the instantiations of the subdesigns which have had the *test_se* ports added to them. One workaround is to use only explicit notation for port connections on instantiations. Another is to modify the input netlist to add a *test_se* port connection to the port list on instantiations of subdesigns which do not contain clock gating cells, but which contain subdesigns that do. The resulting netlist will not be accepted by a Verilog simulator, due to these connections to non-existent ports, but is accepted by the analyze command without complaint, and the test ports are hooked up correctly in the end. We added these dummy port connections using a Perl script just before reading the netlist for synthesis.

Fortunately, with software version 1999.10, the process of creating these new test signals in the design has been completely revamped, causing the designer to make some modifications to the synthesis scripts but allowing the use of implicit notation in HDL designs without the workarounds noted above. The designer must now run the new `hookup_testports` command to insert these test-related control signals into the design.

5.0 Implications of RTL Clock Gating for the Layout Flow

For layout, we used the Cadence Silicon Ensemble tools. The major effect of using RTL clock gating on the layout flow was on the clock tree generation step. The Cadence CT-Gen tool was able to product a clock tree including the clock gating elements introduced by RTL clock gating without any special action on our part; however, the results were less than optimal, and hand-tuning of the CT-Gen operation was required.

One problem is that CT-Gen, if not otherwise instructed, tends to build the clock tree using the largest available buffers. In our technology, these buffers had 32x the normal drive capability. A clock tree constructed of these large buffers generally had good insertion delay and skew characteristics, but consumed much more power then necessary. Since a primary design goal was low power consumption, we elected to manually direct CT-Gen to use smaller buffers, directing our selection using the maximum transition times for the technology library as a guide.

Another problem with the default CT-Gen methodology is that it tended to put as much of the buffering as possible upstream of the clock gating elements. For example, if one branch of the clock tree was much faster than the others, a CT-Gen inserted a buffer string in front of the clock gating element. This means that this string of buffers would consume power on every transition of the system clock. If these buffers were placed downstream of the clock gating element, then they would consume power only when the enable term for that clock gating element was active. By manually editing the buffer tree produced by CT-Gen, we caused these buffer chains to be placed after the clock gating elements whenever possible, reducing clock tree power consumption. A side effect of this practice is that the enable terms for the clock gating elements now have to be valid sooner, to allow the gated clock to propagate down the buffer chain after it is enabled. This did not cause much of a problem for us, for in practice, the enable terms for the clock gating elements had positive slack, and any timing violations introduced by this manual clock tree buffer structuring were fixed by placement-based optimization. The timing analyzer embedded in Synopsys allows this to be analyzed post-layout. If problems are anticipated, the `-setup` option to the `set_clock_gating_style` command allows the designer to anticipate the additional setup time requirement for clock gating elements during initial synthesis.

A general effect on the clock tree generation is that the introduction of clock gating elements into the clock tree introduces some structure onto the tree which must be maintained regardless of placement. This does make extremely low clock skew numbers difficult to attain (our worst-case clock skews were on the order of 1.1 ns); however, once again, placement-based optimization comes to the rescue by fixing setup and hold time violations introduced by the additional clock skew (we allowed for the increased clock skew in the initial synthesis run, and did not fix hold violations in Synopsys, instead allowing the placement-based optimization tool to take care of any hold violations).

With the default clock gating structures provided by Synopsys using standard library cells, we did have to construct two separate clock trees in order to get CT-Gen to produce good results: one clock tree which went to the latches in the clock gating structures, and a second clock tree which went to the AND gates in the clock gating structures. With version 1999.05, Power Compiler allowed the use of a specialized integrated clock gating cell, which is just the standard Synopsys clock gating circuitry contained in a single standard cell. The major advantage to us of

using the integrated clock gating cell is that since it has a single clock pin as opposed to the two pins which are connected to the clock signal in the discrete circuit, it allowed us to halve the number of endpoints on the clock tree for gated elements, helping us to reduce the size of the clock tree and making low skew easier to attain. Also, in the discrete version of the clock gating circuit, the latch element was sometimes separated from the AND gate by a significant distance. Engineers at Intel reported success in replacing the clock gating circuits with hard macro cells using a netlist post processing approach [KM99]. More recently, starting with software revision 1999.05, Power Compiler has added the capability to automatically use such a hard macro cell, termed by Synopsys an *integrated clock gating cell*. A SNUG paper written by engineers at STMicroelectronics [ZVB99] demonstrated successful use of the integrated clock gating cell using this new methodology. After discussions with the authors, we asked our ASIC vendor (coincidentally, also STMicroeletronics) to produce an integrated clock gating cell for our 0.5 µm technology (such a cell is standard for their 0.25 µm and smaller technologies). Example 4 is a script that uses an integrated clock gating cell.

```
set_clock_gating_style -sequential_cell latch
  -positive_edge_logic integrated -negative_edge_logic
  integrated
analyze counter.v
elaborate counter -gate_clock
current_design counter
compile
```

Example 4 - Script for Clock Gating Using Integrated Clock Gating Cell

Another advantage of using an integrated clock gating cell is that the clock signal integrity checks performed by the `-setup` and `-hold` options to the `set_clock_gating_style` command are specified in the cell library entry for the integrated clock gating cell, and the designer no longer needs to specify these parameters to the command (indeed, they are ignored when the integrated clock gating cell is used).

One issue we noted when using the integrated clock gating cell was due to our use of a bottom-up compile methodology (based on modified versions of the long-existing compile scripts written by Glenn Dukes [SYN98]). We discovered that if the design comprising the clock gating circuit (created during the elaborate command and given a name starting with the string "SNPS_CLK_GATE") was compiled directly, the software creates a discrete implementation of the clock gating circuit with no indication that the integrated clock gating cell specification was ignored. Only if this circuit is left uncompiled and the design containing these clock gating subdesigns is compiled (after issuing any necessary `uniquify` commands which are helpfully pointed out by the `elaborate` command) is the integrated clock gating cell used.

The use of the integrated clock gating cell resulted in the elimination of 567 end points from the clock tree (out of a total of 8824), since the AND gate pin and the latch D pin are combined in the integrated clock gating cell.

6.0 Results

The combination of DPR power management with RTL clock gating yielded massive improvements in current consumption, as detailed in Table 2 below.

COMPONENT	CURRENT COMSUMPTION	REDUCTION
Clock tree	36 mA	0%
RAMs	~0 mA	~100%
Flip-Flops	32 mA	81%
Combinatorial Circuitry	10 mA	0%
Total	**78 mA**	**72%**

Table 2 - Current Consumption Analysis of Clock Gated Design

The total current consumption, both for the ungated design and the clock gated design, was measured from prototype silicon. The calculation of the contributions of the different parts of the circuit was performed manually.

RTL clock gating helped us reduce current consumption significantly, mainly in the area consumed by the flip-flops which were being gated. Our analysis did not show a significant improvement in clock tree power consumption, as we expected. We allocated the power consumed in the last stage of the clock tree to the flip-flops, which

Statistics for the generated clock tree as reported by the report_clock_gating command are as follows:

Gating elements	567	
Gated registers	7310	(88.53%)
Ungated registers	947	(11.47%)

Considering that each gating element has an area of 6.33 equivalent gates in the implemented technology, and that each multiplexer eliminated because of the clock gating circuitry has an area of 2.33 equivalent gates, the use of clock gating resulted in a cell area savings of 13,465 equivalent gates, approximately 6.7% of the total design area.

Skew for the two clock trees remained at about 1.0ns. We made no attempt to improve beyond this figure on the pass of the design with clock gating.

7.0 Conclusion

For this design, RTL clock gating allowed us to reduce the dynamic current consumption of a 200K-gate ASIC from 280 mA to 78 mA. Although the chip is planned to enter production for the 2002 model year, prototypes are operational in several hundred modules installed in pre-production vehicles. Except for the reduced power consumption, the RTL clock gating flow is invisible to both the HDL designer and to the end user. It has a fairly minor impact on the synthesis flow and a larger impact on the layout flow. As an added bonus, significant cell area reduction was seen as well.

The main reason we found it necessary to use RTL clock gating for this chip was a combination of design size increase due to feature creep without a corresponding increase in dynamic power budget, along with the hard reality of thermal problems in the high-temperature the device was required to work in. In other cases, we have migrated schematic-based legacy designs, which made much use of asynchronously-clocked latches and other non-synchronous design techniques, to an HDL and logic synthesis-based methodology. This was done for ease of future technology migration, application of automated DFT techniques, and preparation for eventual inclusion in SoC designs. These legacy designs historically have had low power consumption due to the non-synchronous design techniques used in the original schematic implementation. Our customers would not tolerate a large increase in power consumption for these circuits due to our methodology change to a fully synchronous design style. RTL clock gating made it possible for us to enjoy the ease-of-use and portability benefits of using a synchronous design style while maintaining the relatively low power consumption of the non-synchronous versions of the designs. For these reasons, the benefits of RTL clock gating, along with the ease of using it, have made it a standard part of our design flow.

8.0 References

[KM99] Khan, Zia, and Mehta, Gaurav. *Automatic Clock Gating for Power Reduction*, SNUG San Jose, 1999.

[SYN98] Synopsys Inc. *Design Compiler Reference Manual: Fundamentals*, *Release 1998.08,* 1998.

[ZVB99] Zafalon, Roberto; Veggetti, Andrea; and Burger, Roberta. *New Clock Gating Feature in Power Compiler v1999.05,* SNUG San Jose, 1999.