



XAPP790 (v1.0) August 13, 2012

Analysis of Power Savings from Intelligent Clock Gating

Author: Smitha Sundaresan and Frederic Rivoallon

Summary

This application note introduces FPGA designers to intelligent clock gating by describing clock gating support in the Xilinx design tools while supplying a detailed analysis of the impact of clock gating on a design from a logic design and power perspective. Accessing and invoking clock gating support in the Xilinx design tools flow and how to analyze the results is also outlined.

Introduction

Intelligent clock gating is a set of algorithms that can detect unnecessary switching in the design and suppress it. This fully automated method adds a small amount of logic to suppress and minimize nonessential activity in the design, which reduces the power consumed by the design.

Power Terminology

There are two components of power—dynamic power and static power:

$$\text{Dynamic Power} = \alpha \times f_{\text{CLK}} \times C \times V^2 \quad \text{Equation 1}$$

where:

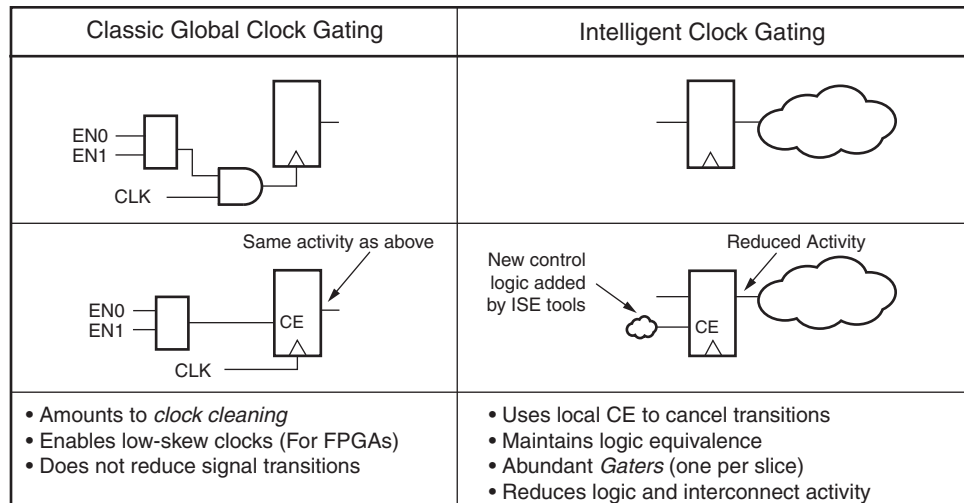
- α is the activity (percentage) of a circuit that switches in each cycle
- f_{CLK} is the frequency of the clock
- V is the supply voltage
- C is the capacitance

The device static power represents the transistor leakage power when the programmable device is powered and not configured.

Intelligent Clock Gating

Intelligent clock gating techniques are used to minimize activity on portions of the design that do not contribute to the design output for that clock cycle. Clock gating reduces dynamic power by preventing logic not used in a given clock cycle from toggling in that clock cycle. Additionally, it prevents the clocks to the flip-flops in cases where the clocking either does not produce new data or the flip-flop outputs are not used by subsequent logic in a given clock cycle. These gating techniques reduce α in [Equation 1](#), resulting in lower dynamic and total power consumed by the design.

The intelligent clock gating technique, shown in [Figure 1](#), uses the local clock enable (CE) signal in the slice to neutralize superfluous switching activity. CE is suited for power optimization because it connects to the basic cluster of logic (the slice). The slice-level clock gating algorithms analyze the logic equations, detect that the sourcing registers do not contribute to the result for each clock cycle, and disable the switching of these registers using a clock enable for these cycles. These changes do not impact the pre-existing logic or clock placement, nor do they create new clocks. These changes do not alter the behavior of the design after clock gating as the design is logically equivalent to the original design.

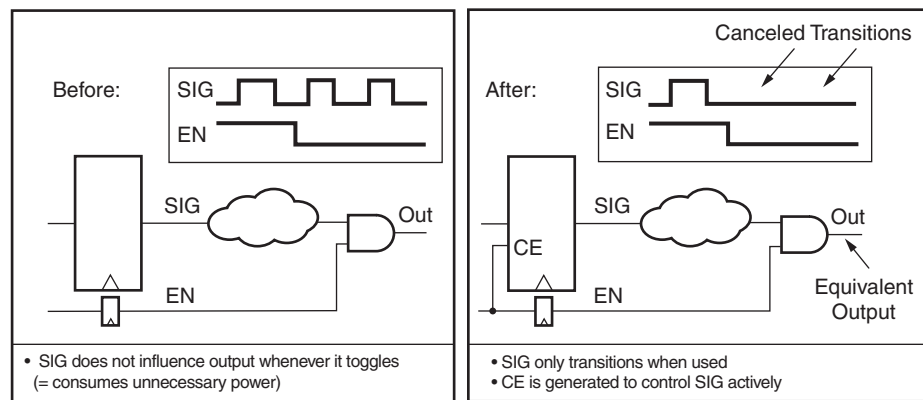


X790_01_061112

Figure 1: Classic Global Clock Gating vs. Intelligent Clock Gating

Clock Gating in Xilinx FPGAs

The Xilinx Design Suite supports clock gating from the pre-design stage to the post-route stage. The Xilinx Power Estimator (XPE) spreadsheet tool (download at <http://www.xilinx.com/power>) helps a new user understand the magnitude of power savings for a particular design when power optimization is enabled in the Xilinx tools. In 7 series FPGAs there are fundamentally eight flip-flops per slice that share a common clock enable. As shown in Figure 2, the clock enable locally gates the clock and also stops the flip-flop from toggling. The ISE® design tools can automatically suppress unnecessary switching by looking for cases where flip-flop outputs will not be used by a downstream target. This is done by a post-synthesis logic examination. The tool then generates local clock enables. Slice clock gating does not re-synthesize the original design, it creates additional LUTs or uses existing ones. However, these changes are incremental as the pre-existing logic is unchanged.



X790_02_071312

Figure 2: Clock Gating Using Clock Enable

Clock Gating Example

This example 8-to-1 multiplexer (8-bit buses) VHDL design illustrates the concept of clock gating. The RTL code creates a basic 8-to-1 multiplexer that is implemented with and without power optimizations switches:

```

library IEEE;
use IEEE.std_logic_1164.all;

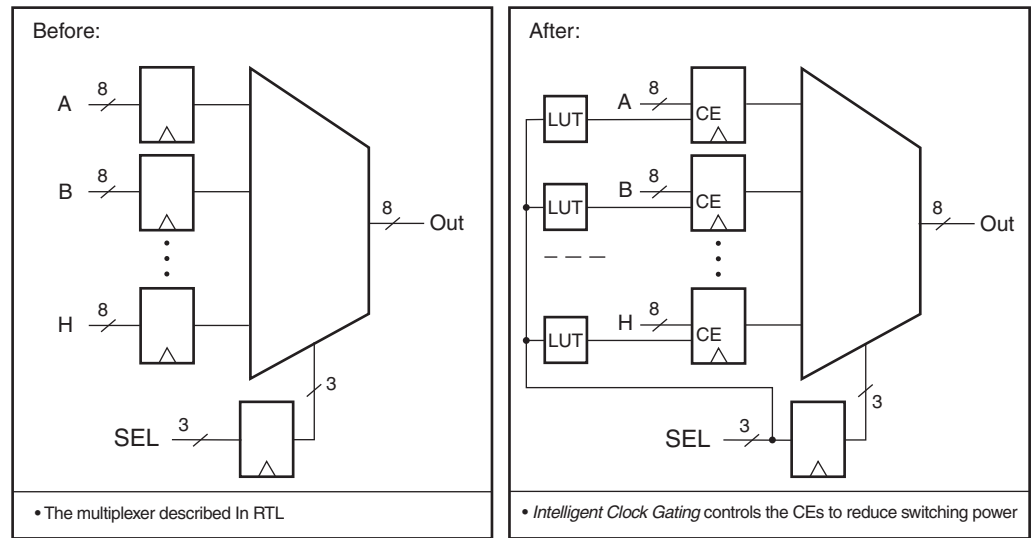
entity multiplexer is
    port (CLK
          : in std_logic;
          SEL_IN
          : in std_logic_vector (2 downto 0);
          A_IN, B_IN, C_IN, D_IN
          : in std_logic_vector (7 downto 0);
          E_IN, F_IN, G_IN, H_IN
          : in std_logic_vector (7 downto 0);
          SIG
          : out std_logic_vector (7 downto 0));
end multiplexer;
architecture RTL of multiplexer is
    signal SEL_INR, SEL
           : std_logic_vector(2 downto 0);
    signal A, B, C, D, E, F, G, H
           : std_logic_vector(7 downto 0);
begin
    -- purpose: register inputs
    -- type    : sequential
    -- inputs  : CLK
    -- inputs  : A_IN, B_IN, C_IN, D_IN, E_IN, F_IN, G_IN, SEL_IN
    -- outputs: SIG

    process (CLK)
    begin -- process
        if rising_edge(CLK) then
            SEL_INR <= SEL_IN;
            SEL      <= SEL_INR;
            A        <= A_IN;
            B        <= B_IN;
            C        <= C_IN;
            D        <= D_IN;
            E        <= E_IN;
            F        <= F_IN;
            G        <= G_IN;
            H        <= H_IN;
            A        <= A_IN;
        end if;
    end process;
    -- purpose: multiplexer 8-to-1
    -- type    : sequential
    -- inputs  : CLK, A, B, C, D, E, F, G, SEL
    -- outputs: SIG
    process (CLK)
    begin -- process
        if rising_edge(CLK) then
            case SEL is
                when "000" => SIG <= A;
                when "001" => SIG <= B;
                when "010" => SIG <= C;
                when "011" => SIG <= D;
                when "100" => SIG <= E;
                when "101" => SIG <= F;
                when "110" => SIG <= G;
                when "111" => SIG <= H;
                when others => SIG <= A;
            end case;
        end if;
    end process;
end RTL;

```

Logic Analysis of Design Before Clock Gating

As illustrated in [Figure 3](#), the [Clock Gating Example](#) has eight 8-bit registers (A, B, ... H). The output of these registers feeds into a multiplexer, which selects one of the registers depending on the contents of the select line. Since only one of the registers is selected by the multiplexer, the other registers can be shutdown, and power savings are achieved.



X790_03_061112

Figure 3: Example Design Using Clock Gating

Logic Addition Due to Clock Gating

[Figure 3](#) also shows the impact of clock gating on the [Clock Gating Example](#). Extra LUTs are added to the CE signal of the registers. The input that feeds these LUTs also feeds into the select line of the multiplexer. As a result, the input to the LUT enables one of the registers and also indicates to the select line the register that the MUX will select in the next clock cycle. Therefore, seven of the eight registers are de-selected during that clock cycle. Substantial power savings are realized at the expense of adding a few logic elements.

The output of the 8-bit registers in [Figure 3](#) (A, B, through H) is the input to a multiplexer, where one of the inputs is chosen depending on the SELECT signal.

Power Estimation using the XPE Tool

The XPE spreadsheet is a power estimation tool that is typically used in the pre-design and pre-implementation phases of a project. The XPE tool assists application specific with architecture evaluation, device selection, appropriate power supply components, and thermal management components. In addition, there are settings available in the tool to estimate the power in a power-optimized mode. These settings are used to determine the expected power savings when using power optimization while implementing a design.

Settings

Device

Family	Kintex-7
Device	XC7K325T
Package	FBG900
Speed Grade	-2L
Temp Grade	Extended
Process	Typical
Voltage ID Used	
Characterization	Advance, v0.7, 2012-04-23

Environment

Junction Temperature	<input type="checkbox"/> User Override	
Ambient Temp		25.0 °C
Effective Θ_{JA}	<input type="checkbox"/> User Override	
Airflow		250 LFM
Heat Sink		Medium Profile
Θ_{SA}		3.3 °C/W
Board Selection		Medium (10"x10")
# of Board Layers		12 to 15
Θ_{JB}		
Board Temperature		

Implementation

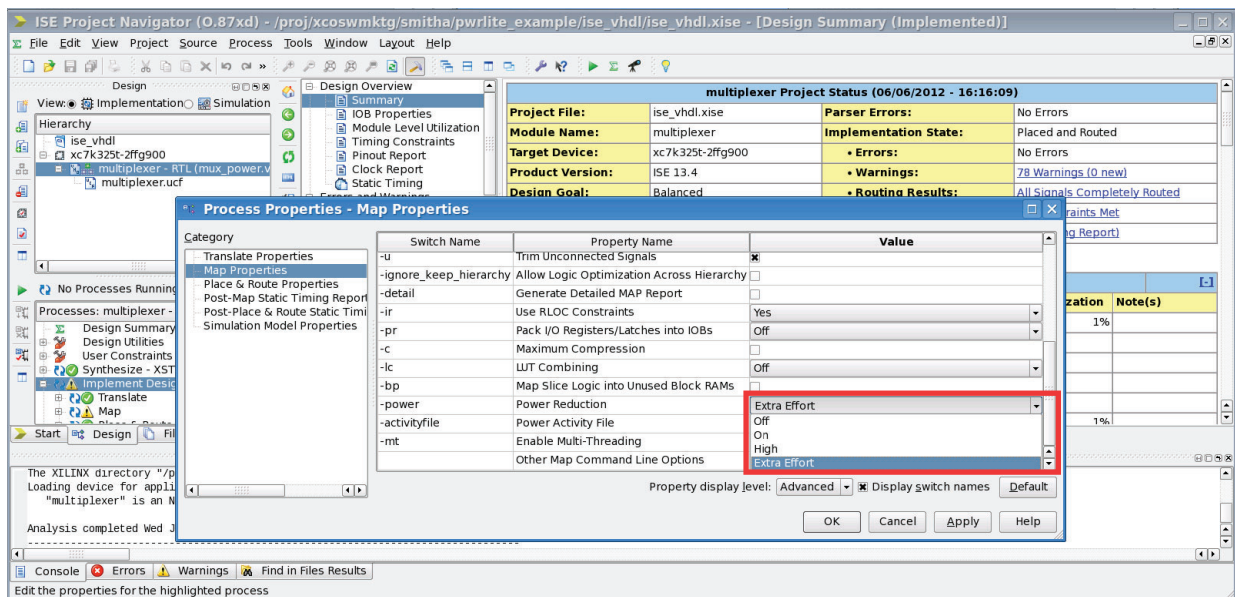
Optimization	Power Optimization
--------------	--------------------

X790_04_061112

Figure 4: Power Optimization Setting in the XPE Tool

Clock Gating Using ISE Design Tools

Using ISE Design Tools, the clock gating features are activated in the map phase by using the `-power high` or `-power XE` options.



X790_05_061112

Figure 5: Map Properties in ISE Design Tools

Analyzing Clock Gating

There are several ways to check that the clock gating algorithms have not changed the logic in the design and have added gating logic to improve power. The first step is to check that the gating algorithm did indeed find structures in the design that are conducive to clock gating.

Analysis of the map report before and after clock gating shows any changes to the design. The design summary section of the map report lists the various resources used by the design. The design summary section of map report for the design prior to power optimization shows:

```

Design Summary
-----
Number of errors:      0
Number of warnings:   78
Slice Logic Utilization:
Number of Slice Registers:      75 out of 407,600    1%
    Number used as flip-flops:   75
    Number used as Latches:      0
    Number used as Latch-thrus:  0
    Number used as AND/OR logics: 0
Number of Slice LUTs:          211 out of 203,800  1%
    Number used as logic:         208 out of 203,800  1%
        Number using O6 output only: 208
        Number using O5 output only:  0
        Number using O5 and O6:      0
        Number used as ROM:          0
Number used as Memory:         3 out of 64,000    1%
    Number used as Dual Port RAM:  0
    Number used as Single Port RAM: 0
    Number used as Shift Register:  3
        Number using O6 output only:  3
        Number using O5 output only:  0
  
```

```

Number using O5 and O6:                0
Number used exclusively as route-thrus: 0

```

Slice Logic Distribution:

```

Number of occupied Slices:                80 out of 50,950    1%
Number of LUT flip-flop pairs used:       211
Number with an unused flip-flop:         136 out of 211      64%
Number with an unused LUT:                0 out of 211        0%
Number of fully used LUT-FF pairs:        75 out of 211      35%
Number of unique control sets:            2
Number of slice register sites lost
to control set restrictions:              10 out of 407,600   1%

```

A LUT flip-flop pair for this architecture represents one LUT paired with one flip-flop within a slice. A control set is a unique combination of clock, reset, set, and enable signals for a registered element. The Slice Logic Distribution report is not meaningful if the design is over-mapped for a non-slice resource or if placement fails. OVERMAPPING of block RAM resources should be ignored if the design is over-mapped for a non-block RAM resource or if placement fails.

IO Utilization:

```

-----

```

```

Average Fanout of Non-Clock Nets:        1.11

```

```

Peak Memory Usage:  1654 MB
Total REAL time to MAP completion:  55 secs
Total CPU time to MAP completion:    54 secs

```

The summary section of the map report for the same design after power optimization shows:

Design Summary

```

-----

```

```

Number of errors:      0
Number of warnings:   157
Slice Logic Utilization:
Number of Slice Registers:  75 out of 407,600    1%
Number used as Flip Flops:  75
Number used as Latches:     0
Number used as Latch-thrus: 0
Number used as AND/OR logics: 0
Number of Slice LUTs:      219 out of 203,800   1%
Number used as logic:       216 out of 203,800   1%
Number using O6 output only: 216
Number using O5 output only: 0
Number using O5 and O6:     0
Number used as ROM:         0
Number used as Memory:      3 out of 64,000      1%
Number used as Dual Port RAM: 0
Number used as Single Port RAM: 0
Number used as Shift Register: 3
Number using O6 output only: 3
Number using O5 output only: 0
Number using O5 and O6:     0
Number used exclusively as route-thrus: 0

```

Slice Logic Distribution:

```

Number of occupied Slices:                99 out of 50,950    1%
Number of LUT Flip Flop pairs used:       219
Number with an unused Flip Flop:         144 out of 219     65%
Number with an unused LUT:                0 out of 219        0%

```

```

Number of fully used LUT-FF pairs:      75 out of 219      34%
Number of unique control sets:         10
Number of slice register sites lost
to control set restrictions:           10 out of 407,600  1%

```

A LUT flip-flop pair for this architecture represents one LUT paired with one flip-flop within a slice. A control set is a unique combination of clock, reset, set, and enable signals for a registered element. The Slice Logic Distribution report is not meaningful if the design is over-mapped for a non-slice resource or if Placement fails. OVERMAPPING of block RAM resources should be ignored if the design is over-mapped for a non-block RAM resource or if placement fails.

IO Utilization:

```
Average Fanout of Non-Clock Nets:      1.33
```

```
Peak Memory Usage: 1665 MB
```

```
Total REAL time to MAP completion: 1 mins 6 secs
```

```
Total CPU time to MAP completion: 1 mins 5 secs
```

A comparison of the two summary sections using power optimization shows a few more slice LUTs (211 in the first report before clock gating; 219 in the second report after clock gating). Analyzing the clock gating report includes further details on the changes made to the design by the clock gating algorithm.

Analyzing Clock Gating Reports

Invoking Power Optimization in the ISE design tools produces a report (PSR) file. Examining this file shows the optimizations the tool produced in the design. The report in this section illustrates the PSR file for the example design. The report lists the number of slice registers and block RAMs gated, the number of clock enable nets processed, and the number of flip-flops added for enable generation. The example design gated 64 registers and processed eight clock-enable nets. The 64 gated registers are listed separately and the CE signal introduced to enable this gating is also listed.

TABLE OF CONTENTS

- 1) Physical Synthesis Options Summary
- 2) Optimizations statistics and details

```

=====
*                               Physical Synthesis Options Summary                               *
=====
---- Options
Global Optimization              : OFF
  Retiming                      : OFF
  Equivalent Register Removal    : OFF
Timing-Driven Packing and Placement : ON
  Logic Optimization            : OFF
  Register Duplication          : OFF

---- Intelligent clock gating    : ON

---- Target Parameters
Target Device                    : 7k325tffg900-2

=====
=====

```



```

*                               Optimizations                               *
=====
---- Statistics

Number of Slice registers gated           :64
Number of block RAM ports gated          :0
Number of clock enable net processed      :8
Number of flip-flops added for Enable Generation:0

---- Details

Component Name | Type | CE Name | Objective
-----|-----|-----|-----
A_0            | FF   | A_0_CE_coolgate_en_sig_4 | Power
A_1            | FF   | A_0_CE_coolgate_en_sig_4 | Power
A_2            | FF   | A_0_CE_coolgate_en_sig_4 | Power
A_3            | FF   | A_0_CE_coolgate_en_sig_4 | Power
A_4            | FF   | A_0_CE_coolgate_en_sig_4 | Power
A_5            | FF   | A_0_CE_coolgate_en_sig_4 | Power
A_6            | FF   | A_0_CE_coolgate_en_sig_4 | Power
A_7            | FF   | A_0_CE_coolgate_en_sig_4 | Power
B_0            | FF   | B_0_CE_coolgate_en_sig_7 | Power
B_1            | FF   | B_0_CE_coolgate_en_sig_7 | Power
B_2            | FF   | B_0_CE_coolgate_en_sig_7 | Power
B_3            | FF   | B_0_CE_coolgate_en_sig_7 | Power
B_4            | FF   | B_0_CE_coolgate_en_sig_7 | Power
B_5            | FF   | B_0_CE_coolgate_en_sig_7 | Power
B_6            | FF   | B_0_CE_coolgate_en_sig_7 | Power
B_7            | FF   | B_0_CE_coolgate_en_sig_7 | Power
C_0            | FF   | C_0_CE_coolgate_en_sig_10 | Power
C_1            | FF   | C_0_CE_coolgate_en_sig_10 | Power
C_2            | FF   | C_0_CE_coolgate_en_sig_10 | Power
C_3            | FF   | C_0_CE_coolgate_en_sig_10 | Power
C_4            | FF   | C_0_CE_coolgate_en_sig_10 | Power
C_5            | FF   | C_0_CE_coolgate_en_sig_10 | Power
C_6            | FF   | C_0_CE_coolgate_en_sig_10 | Power
C_7            | FF   | C_0_CE_coolgate_en_sig_10 | Power
D_0            | FF   | D_0_CE_coolgate_en_sig_13 | Power
D_1            | FF   | D_0_CE_coolgate_en_sig_13 | Power
D_2            | FF   | D_0_CE_coolgate_en_sig_13 | Power
D_3            | FF   | D_0_CE_coolgate_en_sig_13 | Power
D_4            | FF   | D_0_CE_coolgate_en_sig_13 | Power
-----
-----

Flops added for Enable Generation
-----

```

Simulation of the Design Before and After Clock Gating

The same test bench can be used to probe registers when simulating the design before and after power optimization. The logic simulates exactly the same in both cases indicating functional equivalence. The gating of the registers and the reduced activity rate at the output of the registers after power optimization can be observed in the illustrations in this section.

The testbench is available in the reference design ZIP file. See the [Reference Design](#) section.

The design is simulated in ISE design tools by switching to the **Simulation** view in the GUI, choosing a **Post_Route Simulation**, and running the **Simulate Post-Place & Route Model** in the **Processes** window.

[Figure 6](#) and [Figure 7](#) are snapshots of the design's simulation waveform before and after power optimization. Prior to power optimization ([Figure 6](#)), the output of the registers in the design toggles in every clock cycle; however, after power optimization ([Figure 7](#)), the same output registers in the design toggle much less.

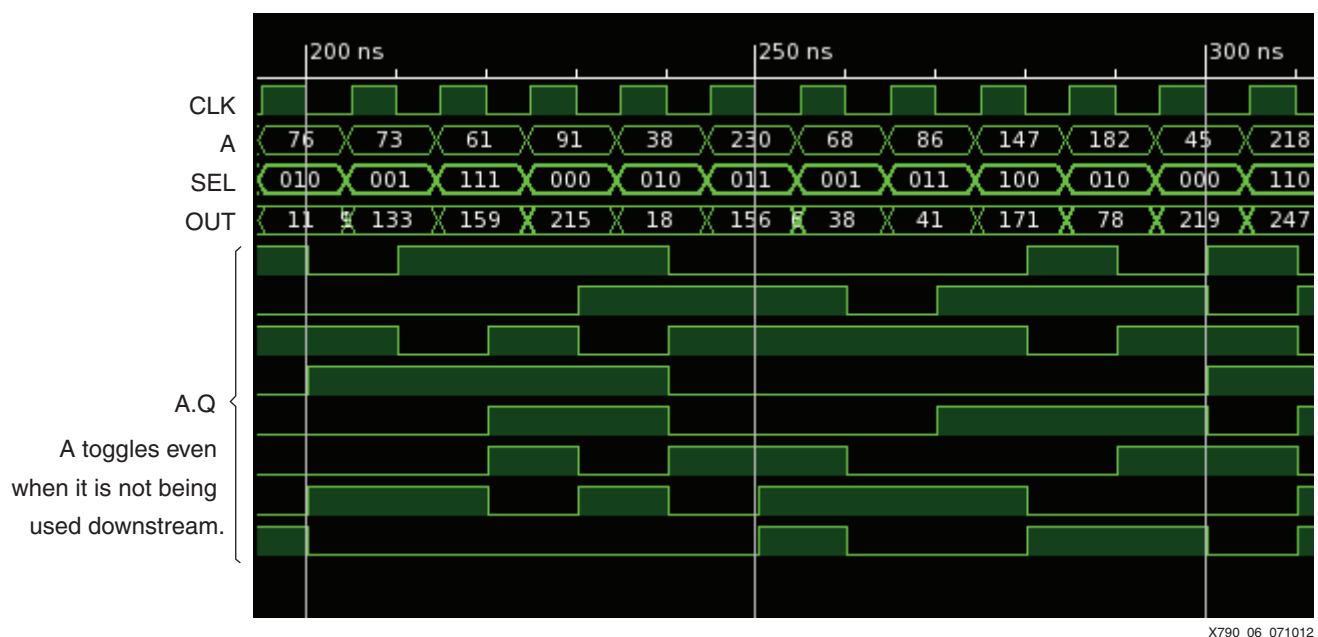
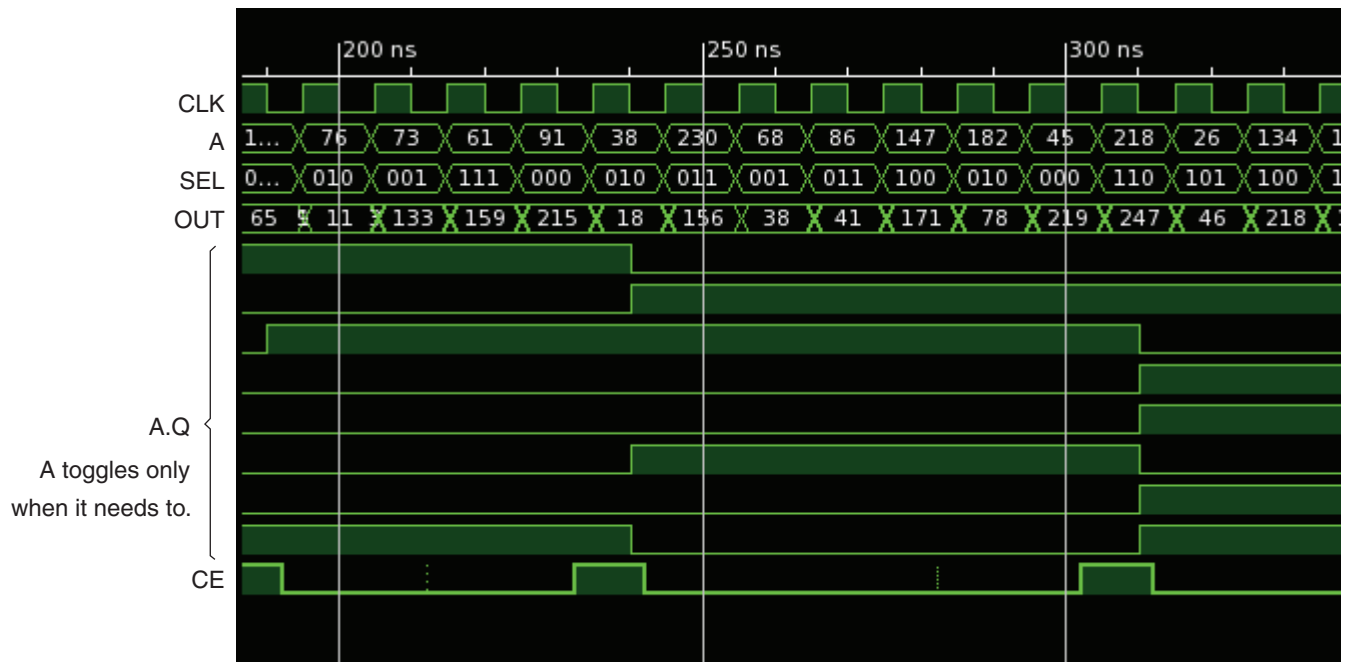


Figure 6: Before Power Optimization



X790_07_071012

Figure 7: After Power Optimization

Analyzing an SAIF File Before and After Clock Gating

When performing a post-route simulation on a design, specify an SAIF file as the output from the simulator to use for further analysis and to produce an accurate power estimation. To specify an SAIF file, select **Process Properties** when right clicking on the **Simulate Post-Place & Route Model** process. Under the **Category of ISIM** properties check **Generate SAIF file for Power Optimization/Estimation**. When the design is simulated it generates an SAIF file for the simulation run.

Analyzing an SAIF File Before Clock Gating

The activity rate for the individual registers prior to running clock gating is shown:

```
(INSTANCE A_1
  (NET
    (o (T0 530000) (T1 470000) (TX 0) (TZ 0) (TB 0) (TC 42))
    (ce (T0 0) (T1 1000000) (TX 0) (TZ 0) (TB 0) (TC 0))
    (clk (T0 500678) (T1 495000) (TX 4322) (TZ 0) (TB 0) (TC 199))
    (i (T0 527196) (T1 470000) (TX 2804) (TZ 0) (TB 0) (TC 43))
    (rst (T0 1000000) (T1 0) (TX 0) (TZ 0) (TB 0) (TC 0))
    (set (T0 1000000) (T1 0) (TX 0) (TZ 0) (TB 0) (TC 0))
  )
)
```

The activity on the output o of register A_1 is 42.

Analyzing an SAIF File After Clock Gating

The activity rate for the individual registers after running clock gating is shown:

```
(INSTANCE A_1
  (NET
    (o (T0 570000) (T1 430000) (TX 0) (TZ 0) (TB 0) (TC 12))
    (ce (T0 678913) (T1 320345) (TX 742) (TZ 0) (TB 0) (TC 42))
    (clk (T0 500696) (T1 495064) (TX 4240) (TZ 0) (TB 0) (TC 200))
    (i (T0 526558) (T1 470000) (TX 3442) (TZ 0) (TB 0) (TC 43))
    (rst (T0 1000000) (T1 0) (TX 0) (TZ 0) (TB 0) (TC 0))
    (set (T0 1000000) (T1 0) (TX 0) (TZ 0) (TB 0) (TC 0))
  )
)
```

The activity on the output o of register A_1 is 12. Before power optimization was run on this design, the activity rate was 42.

Power Estimation Before/After Power Optimization

Since this example design is small, it is difficult to demonstrate the power estimation differences before and after power optimization. However, the activity rate shown in the XPower Analyzer GUI at the output of the registers before and after optimization varies significantly. In the ISE tools GUI, on the Tools menu, select **XPower Analyzer**. The XPower Analyzer process estimates the power for the design. Traverse to the logic section (Figure 8 and Figure 9) and observe that due to clock gating, the signal rate on registers A_0 to A_7 decreases from the 55% – 42% range to the 12% – 5% range.

Name	Power (W)	Type	Clock (MHz)	Clock Name	Signal Rate
A_0	0.00000	FF	100.0	CLK_BUFGP	47.0
A_1	0.00000	FF	100.0	CLK_BUFGP	42.0
A_2	0.00000	FF	100.0	CLK_BUFGP	45.0
A_3	0.00000	FF	100.0	CLK_BUFGP	42.0
A_4	0.00000	FF	100.0	CLK_BUFGP	44.0
A_5	0.00000	FF	100.0	CLK_BUFGP	55.0
A_6	0.00000	FF	100.0	CLK_BUFGP	45.0
A_7	0.00000	FF	100.0	CLK_BUFGP	42.0
B_0	0.00000	FF	100.0	CLK_BUFGP	46.0
B_1	0.00000	FF	100.0	CLK_BUFGP	41.0
B_2	0.00000	FF	100.0	CLK_BUFGP	44.0
B_3	0.00000	FF	100.0	CLK_BUFGP	41.0
B_4	0.00000	FF	100.0	CLK_BUFGP	42.0
B_5	0.00000	FF	100.0	CLK_BUFGP	47.0
B_6	0.00000	FF	100.0	CLK_BUFGP	51.0
B_7	0.00000	FF	100.0	CLK_BUFGP	43.0
C_0	0.00000	FF	100.0	CLK_BUFGP	46.0
C_1	0.00000	FF	100.0	CLK_BUFGP	35.0

XAPP790_08_061212

Figure 8: Register Activity Prior-To Power Optimization

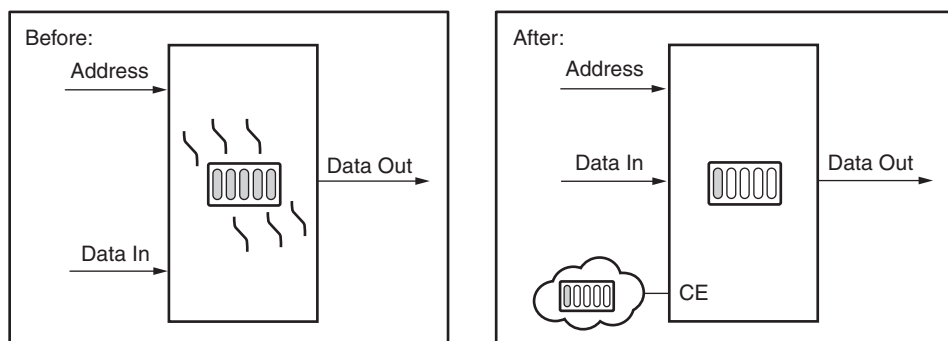
Name	Power (W)	Type	Clock (MHz)	Clock Name	Signal Rate
A_0	0.00000	FF	100.0	CLK_BUF GP	11.0
A_0 CE ...	0.00000	LUT6	Async	Async	42.0
A_1	0.00000	FF	100.0	CLK_BUF GP	12.0
A_2	0.00000	FF	100.0	CLK_BUF GP	11.0
A_3	0.00000	FF	100.0	CLK_BUF GP	10.0
A_4	0.00000	FF	100.0	CLK_BUF GP	10.0
A_5	0.00000	FF	100.0	CLK_BUF GP	9.0
A_6	0.00000	FF	100.0	CLK_BUF GP	5.0
A_7	0.00000	FF	100.0	CLK_BUF GP	6.0
B_0	0.00000	FF	100.0	CLK_BUF GP	7.0
B_0 CE ...	0.00000	LUT6	Async	Async	45.0
B_1	0.00000	FF	100.0	CLK_BUF GP	4.0
B_2	0.00000	FF	100.0	CLK_BUF GP	7.0
B_3	0.00000	FF	100.0	CLK_BUF GP	7.0
B_4	0.00000	FF	100.0	CLK_BUF GP	6.0
B_5	0.00000	FF	100.0	CLK_BUF GP	5.0
B_6	0.00000	FF	100.0	CLK_BUF GP	5.0
B_7	0.00000	FF	100.0	CLK_BUF GP	9.0

XAPP790_9_061212

Figure 9: Register Activity After Power Optimization

Additional Optimizations

Intelligent clock gating optimization also reduces power for dedicated block RAM in either simple or dual-port mode. These blocks provide several enables; an array enable, a write enable, and an output register clock enable. Most of the power savings comes from using the array enable, as shown in Figure 10.



X790_10_061112

Figure 10: Array Enable Implementation

For example, in a block RAM followed by a 2-to-1 multiplexer, the optimization implements an OR function in a LUT with the write enable (WER) and the select (PRESELECTR) and connects them to the ENARDEN of the block RAM. The OR function ensures that the block dissipates less power when no data is being written and when its output is not used (i.e., not selected in the multiplexer). Assuming a 50% toggle rate on the write enable of the block RAM, this optimization shows a 25% reduction in dynamic power. An example of the Verilog code is shown:

```

`timescale 1ns/1ps

module ram_mux #(
    parameter RAM_WIDTH = 8,
              RAM_ADDR_WIDTH = 12
) (
    input SELECT, CLK, WE,
    input [RAM_WIDTH-1:0] BYPASS,
    input [RAM_ADDR_WIDTH-1:0] ADDR,
    input [RAM_WIDTH-1:0] DATA_IN,
    output reg [RAM_WIDTH-1:0] RESULT_OUT = {RAM_WIDTH{1'b0}}
);

reg preselectR = 1'b0, selectR = 1'b0, weR = 1'b0;
reg [RAM_ADDR_WIDTH-1:0] addr_reg = {RAM_ADDR_WIDTH{1'b0}};
reg [RAM_WIDTH-1:0] mem [2**RAM_ADDR_WIDTH:0];
reg [RAM_WIDTH-1:0] data_in_reg = {RAM_WIDTH{1'b0}};
reg [RAM_WIDTH-1:0] data_out = {RAM_WIDTH{1'b0}};

always @(posedge CLK) begin
    // RAM block (inferred)
    if (weR)
        mem[addr_reg] = data_in_reg;
    data_out <= mem[addr_reg];
    // Registering inputs
    data_in_reg <= DATA_IN;
    addr_reg <= ADDR;
    weR <= WE;
    preselectR <= SELECT;
    selectR <= preselectR;
    // Mux: RAM output and input data
    RESULT_OUT <= selectR ? data_out : BYPASS;
end

endmodule // ram_mux

```

In that same version of design tools, these optimizations can detect a clock enable implemented as logic and replace it with a dedicated CE, as shown in [Figure 11](#).

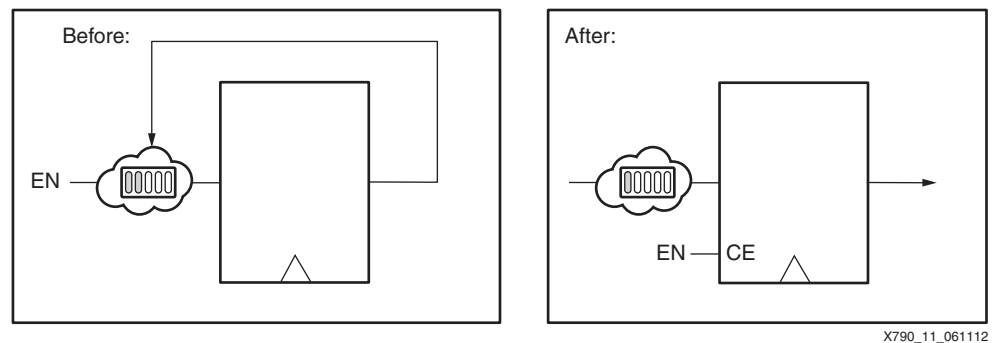


Figure 11: Clock Enable Implementation

Simulation of Design Before Clock Gating

By simulating the design before and after power optimization, the gating of the block RAM enables and the reduced activity rate at the output of the block RAM after power optimization can be observed.

The testbench is available in the reference design ZIP file. See the [Reference Design](#) section.

The design is simulated in ISE tools by switching to the **Simulation** view in the GUI, choosing a **Post_Route** Simulation, and running **Simulate Post-Place & Route Model** in the **Processes** window.

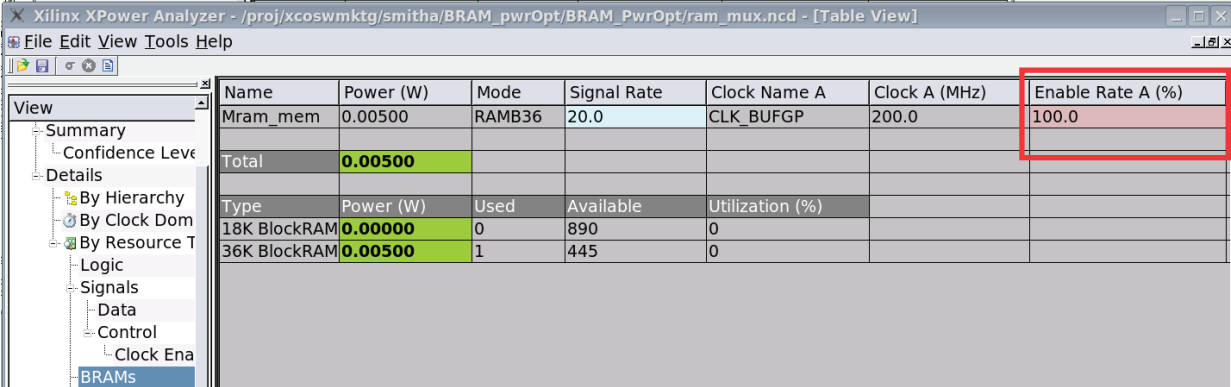
Generating an SAIF File Before and After Clock Gating

When performing a post-route simulation on the design the simulator can output an SAIF file to be used for further analysis and ensure an accurate power estimation.

- Select **Process Properties** after right clicking on the **Simulate Post-Place & Route Model** process pull-down.
- Under the category of **ISIM Properties** check **Generate SAIF file for Power Optimization/Estimation**.
- When the design is simulated, it generates an SAIF file for the simulation run.

Power Estimation Before and After Power Optimization

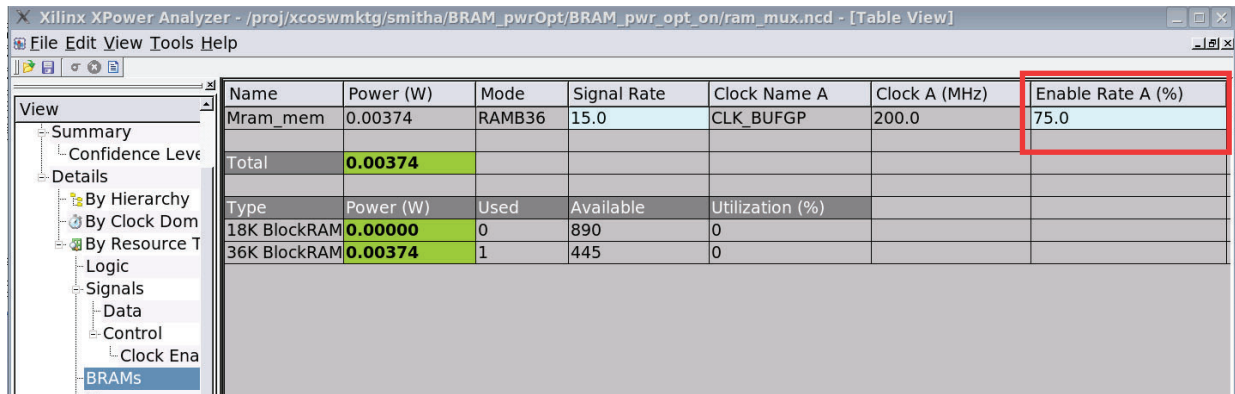
The design can be opened before and after power optimization in the XPower Analyzer GUI. The block RAM enable rate before and after optimization varies significantly. The SAIF file gets loaded automatically since simulation is run prior to running the XPower Analyzer. In the ISE tools GUI, on the **Tools** menu select **XPower Analyzer**. The XPower Analyzer process estimates the power for the design. Traverse to the block RAM section ([Figure 12](#) and [Figure 13](#)) and observe that the block RAM enable rate decreases from almost 100% to 75% and that the total block RAM power is reduced from 0.00500W to 0.00374W.



Name	Power (W)	Mode	Signal Rate	Clock Name A	Clock A (MHz)	Enable Rate A (%)
Mram_mem	0.00500	RAMB36	20.0	CLK_BUFGP	200.0	100.0
Total	0.00500					
Type	Power (W)	Used	Available	Utilization (%)		
18K BlockRAM	0.00000	0	890	0		
36K BlockRAM	0.00500	1	445	0		

X790_12_061112

Figure 12: BRAM Enable Rate Prior To Power Optimization



Name	Power (W)	Mode	Signal Rate	Clock Name A	Clock A (MHz)	Enable Rate A (%)
Mram_mem	0.00374	RAMB36	15.0	CLK_BUFGP	200.0	75.0
Total	0.00374					
Type	Power (W)	Used	Available	Utilization (%)		
18K BlockRAM	0.00000	0	890	0		
36K BlockRAM	0.00374	1	445	0		

X790_13_061112

Figure 13: Block RAM Enable Rate After Power Optimization

Power Optimization Best Practices

The recommended best practices for power savings using power optimization techniques are described in this section.

Gate Clock or Data Paths

This common technique stops paths when the results of these paths are not used. Gating a clock stops all driven synchronous loads; gating a data path keeps signal switching and glitches from continuing to propagate up to the next synchronous element. The software tools analyze the description and netlist to detect unwanted conditions. Still, there are things the designer knows about the application, data flow, and dependencies which are not available to the tool, and can only be specified by the designer.

General Guidelines for Gating

- Maximize the number of elements affected by the gating signal. For example, it is more power efficient to gate a clock domain at its driving source than to gate each load with a clock enable signal.
- When gating or multiplexing clocks to minimize activity or clock tree usage, use the clock enable ports of dedicated clock buffers. Inserting LUTs or other methods to gate-off clock signals is not efficient for power and timing considerations.
- Minimize the number of control sets. Since adding gating signals to stop the data or clock path could require additional logic and routing, it is important to minimize the number of additional structures to avoid defeating the original purpose. Placement and routing of these additional resources can complicate the implementation of the existing logic. This could result in spread-out placement, additional replications, or increased routing congestion, which would then increase the dynamic power.
- When a priority encoder is not needed, use a *case* block instead of an *if-then-else* block as illustrated in this example:

Incorrect:

```
assign val = reg1 ? reg1_in1 :
            ((reg2) ? reg1_in2 :
             ((reg3 ? reg1_in3 : reg1_in4))) ;
```

Correct:

```
case (1'b1)
reg1 : val = reg1_in1 ;
  reg2: val = reg1_in2 ;
  reg3: val = reg1_in3 ;
  default: val = reg1_in4 ;
endcase
```


- Xilinx recommends avoiding the use of asynchronous reset on data-path flip-flops, pipeline flip-flops, and block RAMs.
- Slice gating works on bytes. Buses of less than eight bits are not gated for designs targeted at 7-series devices. The power optimization code detects a bus based on the connectivity, not just by the name.
- Incomplete RTL reset templates lead to wasted CE pins. In [Figure 14](#), the example snippet of RTL code *bar* is assigned only in the *else* block and not in the *if* block. Synthesis assumes it needs to be held if reset is active and will end up connecting reset to CE, which will block power optimization on the register.

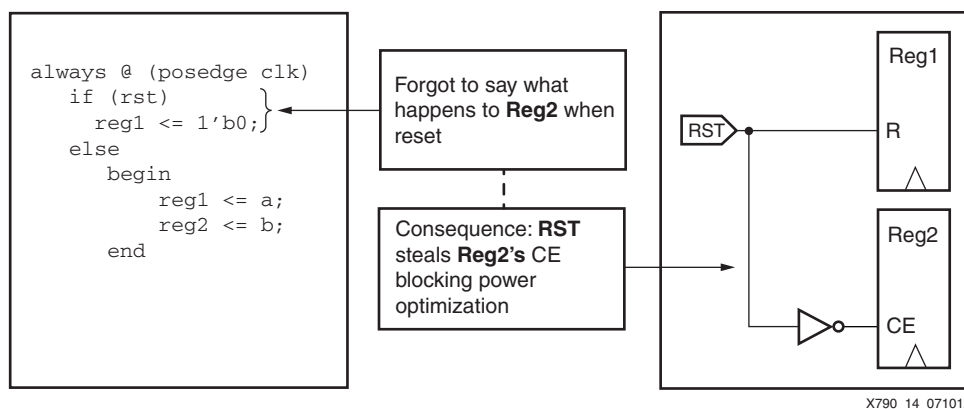


Figure 14: RTL Example

Block RAM

- The amount of power the block RAM consumes is directly proportional to the amount of time it is enabled. To save power, the block RAM enable can be driven Low on clock cycles when the block RAM is not used in the design. Both the block RAM enable rate and the clock rate are important parameters that must be considered for power optimization.
- The NO_CHANGE mode should be used in the TDP mode if the output latches remain unchanged during a write operation. This mode is the most power efficient. This mode is not available in the SDP mode because it is identical in behavior to the WRITE_FIRST mode.

Reference Design

The reference design files for this application note can be downloaded from:

<https://secure.xilinx.com/webreg/clickthrough.do?cid=190051>

[Table 1](#) shows the reference design checklist for this application note.

Table 1: Reference Design Matrix

General	
Developer Name	Smitha Sundaresan
Target devices	7 Series FPGAs
Source code provided	Yes
Source code format	Verilog
Design uses code/IP from existing Xilinx application note/reference designs, CORE Generator™ software, or 3rd party	No

Table 1: Reference Design Matrix (Cont'd)

Simulation	
Functional simulation performed	No
Timing simulation performed	Yes
Testbench used for functional and timing simulations	Yes
Testbench format	Verilog
Simulator software version	ISIM 13.4
SPICE/IBIS simulations	Y/N
Implementation	
Synthesis software tools/version	XST
Implementation software tools /versions	ISE 13.4
Static timing analysis performed	No
Hardware Verification	
Hardware verified	No
Hardware platform used for verification	N/A

Conclusion

As outlined in this application note and [WP370, Reducing Switching Power with Intelligent Clock Gating](#), FPGA designs can benefit from intelligent clock gating. Xilinx design tools readily support these techniques and supply detailed analysis of the impact of clock gating on a design from a logic design and power perspective.

Revision History

The following table shows the revision history for this document.

Date	Version	Description of Revisions
08/13/12	1.0	Initial Xilinx release.

Notice of Disclaimer

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of the Limited Warranties which can be viewed at <http://www.xilinx.com/warranty.htm>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: <http://www.xilinx.com/warranty.htm#critapps>.

Automotive Applications Disclaimer

XILINX PRODUCTS ARE NOT DESIGNED OR INTENDED TO BE FAIL-SAFE, OR FOR USE IN ANY APPLICATION REQUIRING FAIL-SAFE PERFORMANCE, SUCH AS APPLICATIONS RELATED TO: (I) THE DEPLOYMENT OF AIRBAGS, (II) CONTROL OF A VEHICLE, UNLESS THERE IS A FAIL-SAFE OR REDUNDANCY FEATURE (WHICH DOES NOT INCLUDE USE OF SOFTWARE IN THE XILINX DEVICE TO IMPLEMENT THE REDUNDANCY) AND A WARNING SIGNAL UPON FAILURE TO THE OPERATOR, OR (III) USES THAT COULD LEAD TO DEATH OR PERSONAL INJURY. CUSTOMER ASSUMES THE SOLE RISK AND LIABILITY OF ANY USE OF XILINX PRODUCTS IN SUCH APPLICATIONS.