02207 : Advanced Digital Design Techniques

Design for Low Power by Reducing Switching Activity

LAB 2

Group *dt07*

Markku Eerola (s053739)

Rajesh Bachani (s061332)

Josep Renard (s071158)

November 21, 2007

Contents

1 Introduction

The purpose of this exercise was to estimate the power dissipation in a digital circuit due to the switching activity in the cells. Power is dissipated in a digital circuit, dynamically, in two ways; one, the power that is spent in charging or discharging the capacitance load connected to the output of the cell, and two, the power dissipated inside the cell due to short circuit currents and the internal capacitance charging or discharging. This holds for combinational cells. For sequential cells, there is extra power spent at every clock cycle, even if the output of the cell does not change. This is because there is some reaction to every clock cycle in sequential cells, which would take some power.

Static power in digital circuits is due to the internal leakage currents in CMOS. Though, in this exercise, we are particularly interested in analyzing the dynamic power dissipation.

We estimate the dynamic power in a serial to parallel converter. The converter takes in 8 bits (one byte) in every clock cycle, and gives out 32 bits (4 bytes) after every 4 clock cycles. The input byte at the first clock cycle is the most significant byte in the output, whereas the input byte in the fourth clock cycle is the lowest significant byte. The converter, thus, waits for four clock cycles to produce an output. We refer to the register holding the most significant byte in the output as the most significant register, and that holding the least significant byte as the least significant register.

The report is organized as follows. In section 2, we discuss three designs for a serial to parallel converter. In section 3, we simulate the VHDL code for the designs using Modelsim, and verify that all the designs are working correctly. Section 4 contains the power results obtained from the synthesis of the VHDL using Design Vision, and Synopsys VSS for annotating the switching activity in a given time period. In this section, we discuss and justify the results obtained. Later, in section 5, the VHDL is provided, alongwith the power reports from Design Vision.

1.1 Authors by Section

- *Rajesh Bachani* VHDL code for Design A, Design B and Design C, and simulation of the designs in Modelsim.

- *Josep Renard* Synthesis of the designs for power reports, using Synopsys VSS and Design Vision.

- *Markku Eerola* Discussion and Analysis of the power reports.

2 Designs for Serial to Parallel Conversion

In this section, we give an overview of the three designs for serial to parallel conversion, which are evaluated for their power consumption in this exercise.
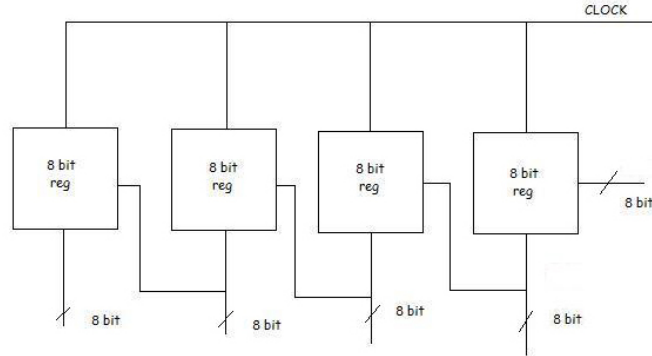
2.1 Design A: Shift Register

Figure 1: Converter using a 8-bit Shift Register

As we can see in figure 1, the input data flows continuously through the registers. On every rising clock edge each of the 8-bit registers takes on a new state. The least significant register takes its state from the input to the converter block and all the other registers take their states from the outputs of the adjacent less significant register. All components of this block are driven with the same clock signal, which ensures that the 8-bit registers change their state at the same time as the clock signal event.
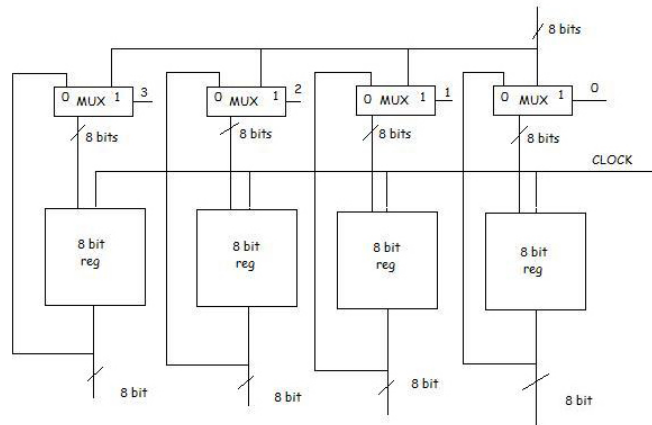
2.2 Design B: Register with Enable

Figure 2: Converter using 8-bit Registers with Enable

In figure 2 we see another design for a serial to parallel converter. In this design we use multiplexers to control when the 8-bit registers should take on a new state. The multiplexers take in two inputs, one from the output of the 8-bit register connected with that multiplexer, and one from the input to the converter block, which is Qk. When the enable signal to a multiplexer is SET, it lets through Qk, whereas, in the other case, it lets through the data from the 8-bit register's output, thus retaining the register's previous state. The enable signal to the four multiplexers is changed in a sequence, with the multiplexer 3 getting the enable first, and the multiplexer 0 getting the enable last. This way, the value of Qk in the first clock cycle is transferred to the most significant register, while Qk in the fourth clock cycle is transferred to the least significant register.
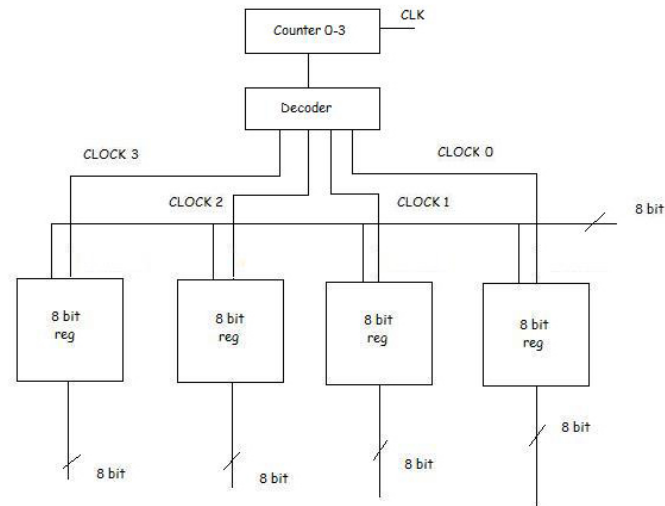
2.3   Design C: Register with Clock-Gating

Figure 3: Converter using 8-bit Registers with Clock Gating

In figure 3 we see the third design for a serial to parallel converter we used in the exercise. In this design we restirct the amount of register state changes by not driving the 8-bit registers with the clock directly but instead using a 2-bit counter and a 2:4 decoder to give a rising clock edge only to one of the four 8-bit register's at a time. In this design the 8-bit registers operate only when they change their state once every fourth clock cycle. This means that they consume much less power. Of course the logic for dividing the clock consumes power as well, but we expect that the power savings which are gained by reducing the operation of the 8-bit registers outweighs this, since we're not only restricting the number of state changes but we're also completely removing the power consumption for 'idle' operations.

3 Simulation of the designs with Modelsim

All the three designs are simulated with Modelsim, to verify the functionality.

The following two screenshots demonstrate the working of implementation for Design A. The first screenshot is taken at 33ns while the second is taken at 43ns. It can be seen that in the new clock cycle, the 8 bit registers have rippled their values to the more significant register, and the value of Qk for that clock cycle is fed into the least significant register. The most significant register looses its old value.
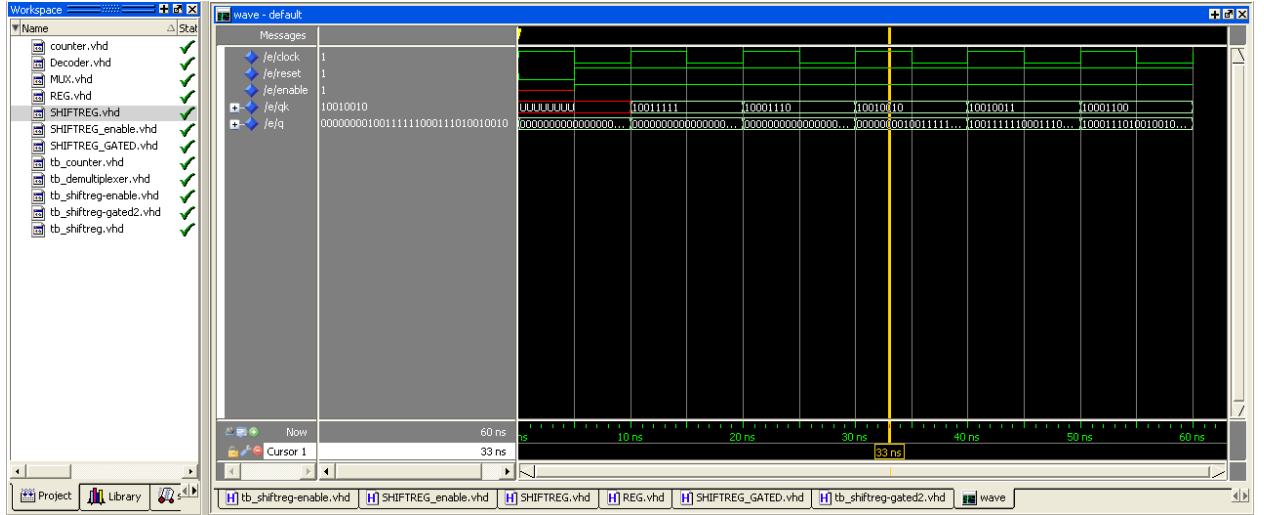

Figure 4: Simulation screenshot for Design A at 33ns
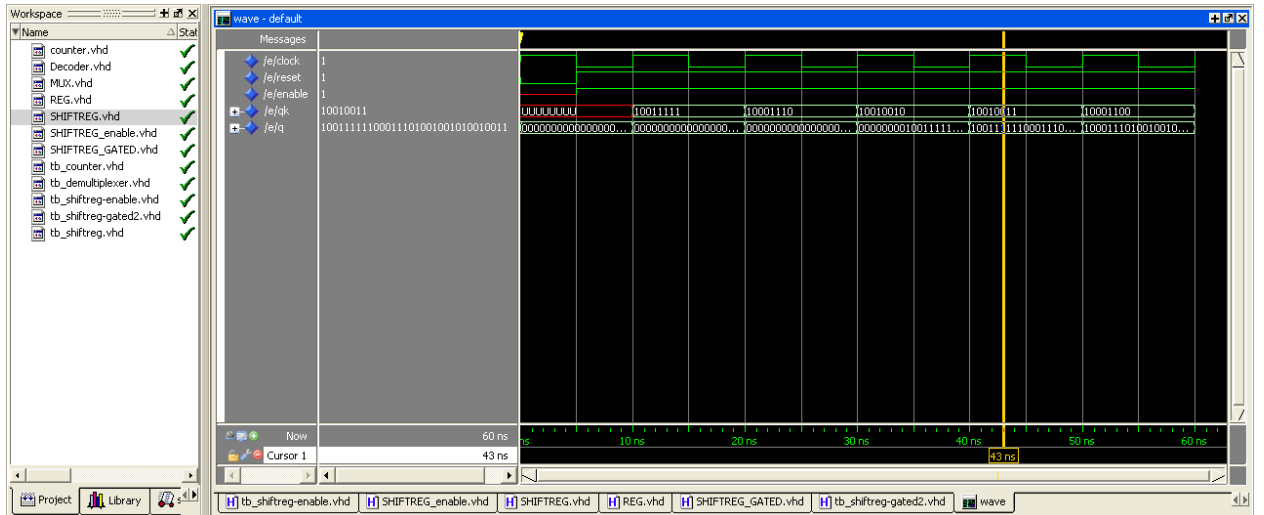
Figure 5: Simulation screenshot for Design A at 43ns

5

The following screenshots are from the simulation of Design B. In the first instance, at 14ns on the timeline, we have some value at Qk, but it has not been transferred in any way to the output Q. Then, at 24ns, the value of Qk in the previous clock cycle is loaded into the most significant register. Further on, at 33ns, the value of Qk in the previous clock cycle is loaded into the second most significant register. This repeats for four clock cycles, after which the most significant register is again loaded.
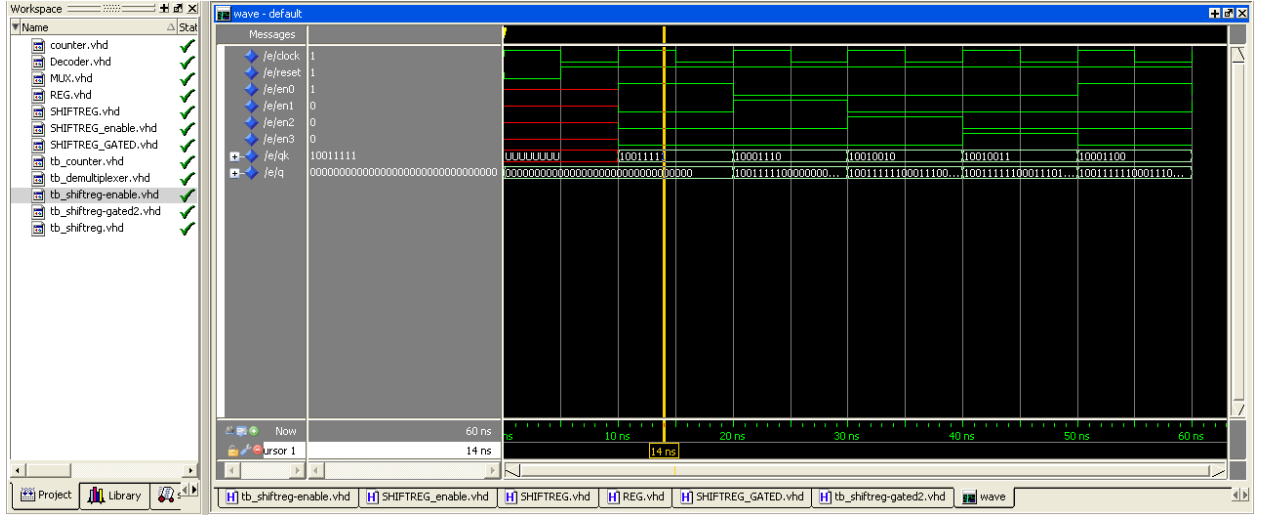
Figure 6: Simulation screenshot for Design B at 14ns
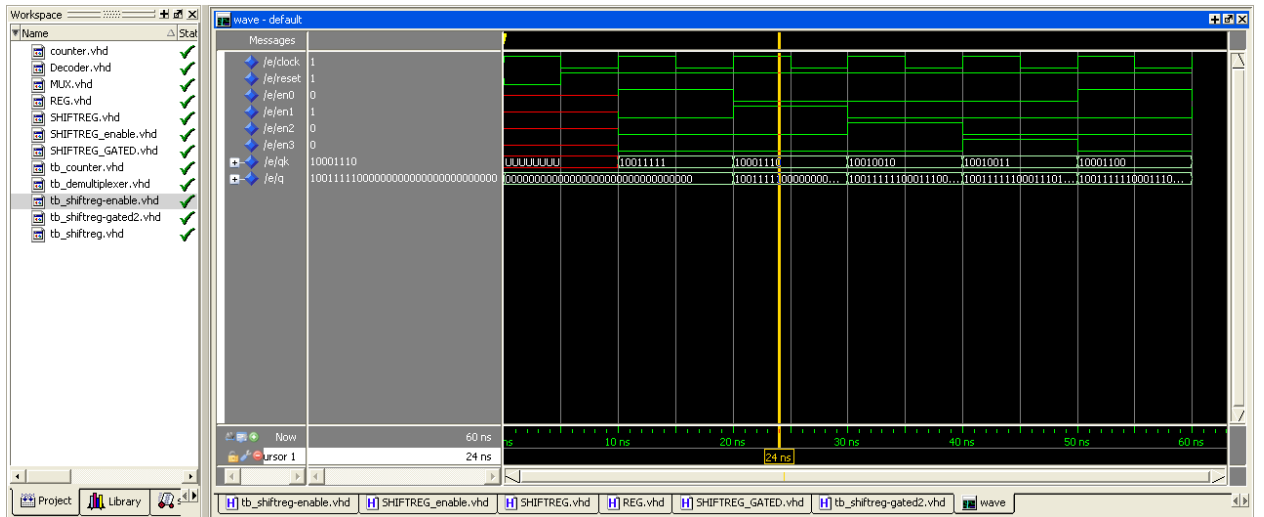
Figure 7: Simulation screenshot for Design B at 24ns

6
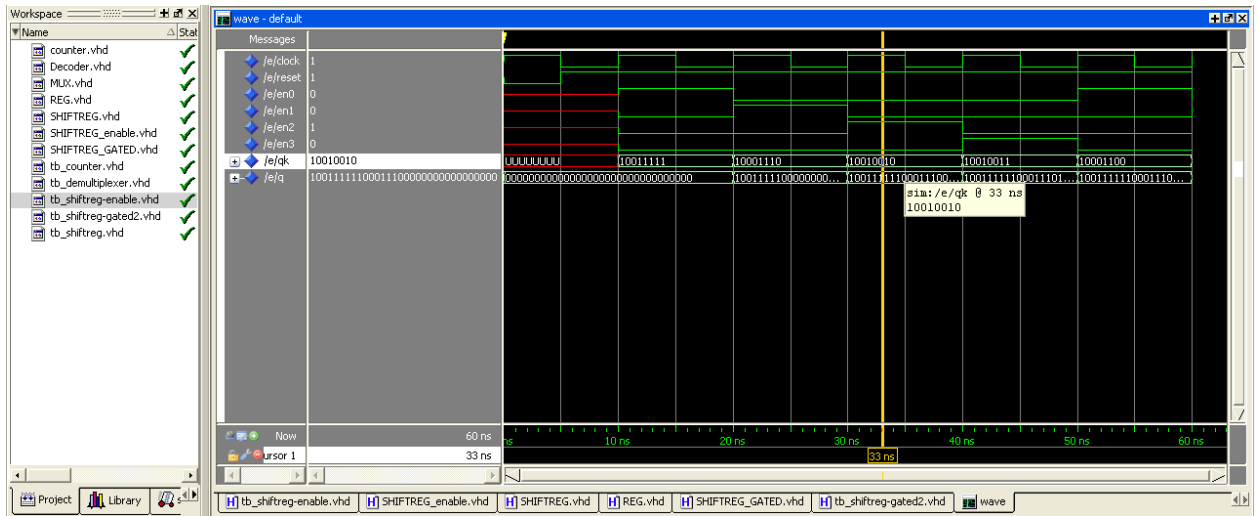
Figure 8: Simulation screenshot for Design B at 33ns

Then, for Design C, we have the following screenshots. As we can see, the values of Qk are transferred to different registers every clock cycle. This is practically the same functionality as Design B. The only difference though is that in Design C, Qk is transferred in the same clock cycle, while in Design B, it happens one clock cycle later. Ofcourse, the internal working of the two designs are completely different, which has already been discussed in section 2

Figure 9: Simulation screenshot for Design C at 14ns

Figure 10: Simulation screenshot for Design C at 24ns

4 Power Reports and Discussion
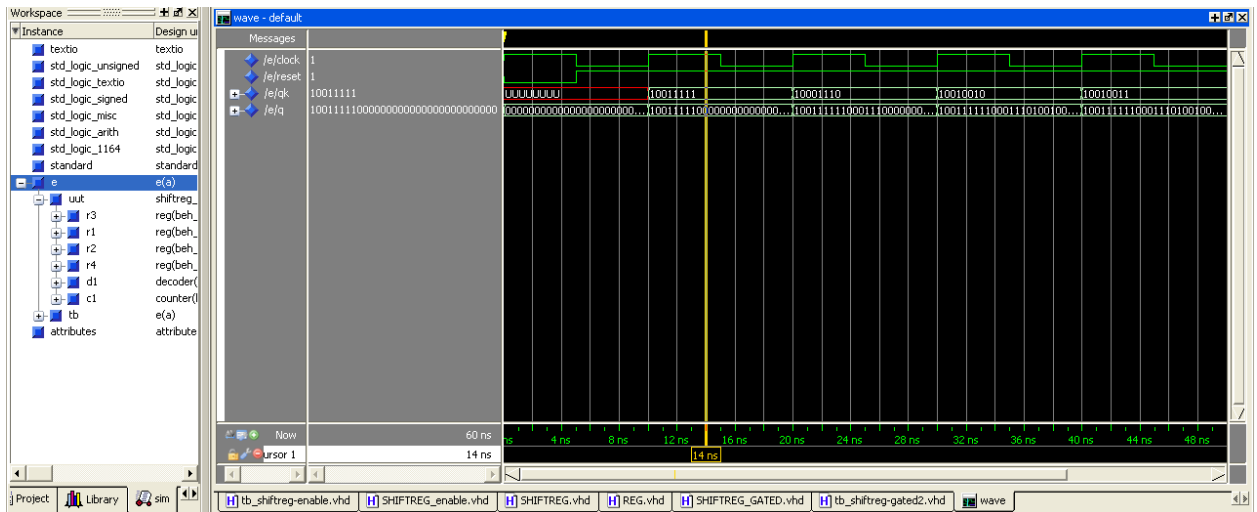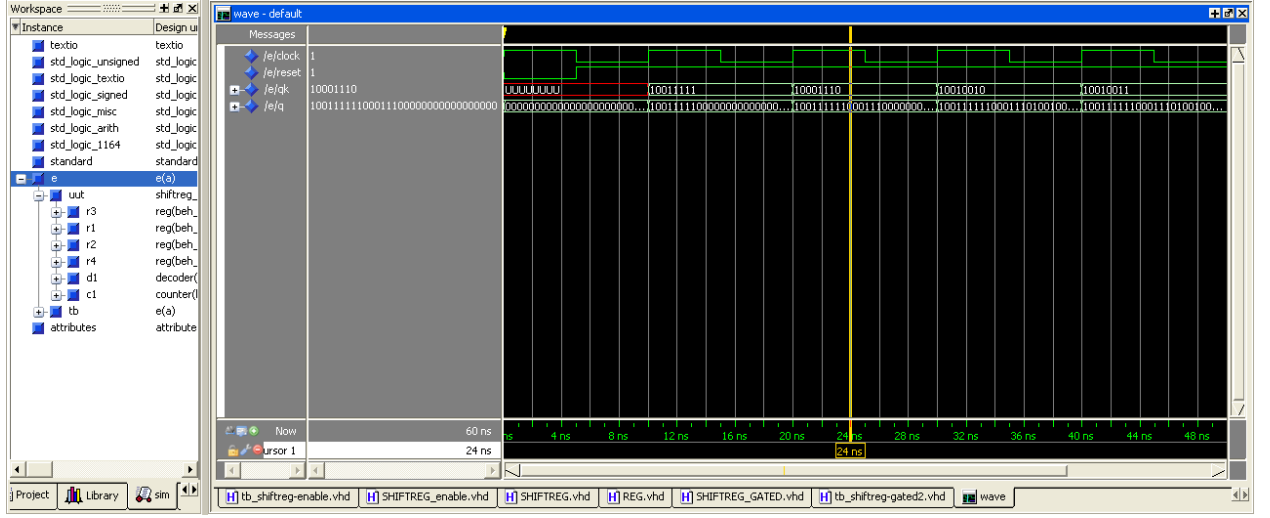
In this section we discuss the results obtained from the power-aware synthesis of the three designs. The Synopsys VSS Simulator is used to annotate the switching activity, based on a testbench for each design. This switching activity is used by Design Vision to estimate the total dynamic power consumption for the design. We have synthesized the designs for clock time periods of 2ns and 10ns and got the same results for all three designs for both clock periods. The power reports obtained from the synthesis are presented in section 5. For a short recap the results can be seen in table 1:

	Total Dynamic Power	Cell Leakage Power
Design A	55.2 uW	773.7 nW
Design B	47 uW	800.5 nW
Design C	32.6 uW	835.0 nW

Table 1: Overview of Results from Power Reports

According to the results design A has the highest dynamic power consumption while design C has the least. We expected as much from design C, but from what we had learned on the lectures we had expected that design B would have consumed more power than design A. The static power consumption which comes from the internal leakage currents, is considerably lower than the dynamic power consumption in all of the designs. This is because static power consumption just depends on the number of cells in the design and does not depend on the switching activity. In the following analysis, we will only consider the dynamic power consumption and from now on when we use the word power we refer to the dynamic power.

In Design A the 8-bit registers change their state on each clock cycle. The input Qk is transferred from one register to the adjacent more significant register, every clock cycle, until it reaches the most significant register. This is the reason why the design is much consuming in terms of power: in every clock cycle, there is a switching activity in all the output bits of

8

the four registers. Since switching accounts for a lot of power, for a given time line of the simulation, we have high levels of power consumption.

Design B is more efficient than Design A from what is seen in the dynamic power consumption. Switching in Design B is controlled by the enable signals, which indicate which register should be loaded with Qk in the next clock event. If the enable signal is SET, the register is loaded with Qk, otherwise the output of the register is reloaded back into the register. Where the former case consumes a lot of dynamic power since the output bits are changed, the latter consumes less dynamic power which is due to internal response to the clock signal. So, since the registers still consume internal power on each clock cycle and since the multiplexers and the logic generating their enabling signals, both add to the overall power consumption, Design B should be less efficient than Design A for the overall power consumption. This is backed up by the lecture notes: the ratio of power dissipated in Design A to that in Design B is 1:1.19, which indicates that Design B should consume more power. We believe that the reason why our results differed from this is that the logic that generated the enabling signals was within the testbench instead of the converter and thus its power dissipation was not considered in the power analysis.

Lets consider Design C now. It is quite clear that this design is most efficient among the three. This design is based on clock-gating, which means that the original clock signal is not sent directly to the registers, but sent only when the register should be loaded with a new value. So, if there is a clock signal, the register funtions normally, and power is dissipated both internally and for charging/discharging the load (if the output changes). So, 75% of the time the clock would be cut off completely, thus saving at least the internal power dissipation due to clock cycles. In Design B, even if the enable for a register was RESET, which meant that the register output would not change, there was still internal power dissipation in the cell for every clock cycle. This is avoided in Design C since the registers do not get clock signals except when they are supposed to change their state.

We see that the attempt to reduce power disspation has been successful, without affecting the functionality of the circuit. With the reduction in power consumption come some costs, in the form of extra area and time delay. The extra logic in designs B and C use up more area and some delay is also added to the critical path. It should also be noted, that in Design C the rising clock events for the 8-bit registers come a bit later than the event at the actual clock signal, which is referred to as skew. This means that the clock period must not be shorter than the delay or the circuit will not work properly.

5 Implementation and Power Reports

Listing 1: SHIFTREG.vhd

```vhdl
library IEEE;
   use IEEE.std_logic_1164.all;
   use IEEE.std_logic_misc.all;
   use IEEE.std_logic_signed.all;
   use IEEE.std_logic_arith.all;

entity SHIFTREG is
      Port (    CLOCK : In      std_logic;
                RESET : In      std_logic;
               ENABLE : In      std_logic;
                   QK : In      std_logic_vector (7 downto 0);
                    Q : InOut    std_logic_vector (31 downto 0) );
end SHIFTREG;

architecture BEHAVIORAL of SHIFTREG is

   begin
    process(RESET,CLOCK)
     variable i,j,k,l : integer;
     begin

        if ( RESET = '0' ) then
           for i in 0 to 31 loop
               q(i) <= '0';
           end loop;
        elsif ((CLOCK = '1') AND (CLOCK'EVENT)) then
               for i in 31 downto 8 loop
                   q(i) <= q(i-8);
               end loop;
               q(7 downto 0) <= qk;
        end if;

    end process;

end BEHAVIORAL;

configuration CFG_SHIFTREG_BEHAVIORAL of SHIFTREG is
   for BEHAVIORAL
   end for;

end CFG_SHIFTREG_BEHAVIORAL;
```

Listing 2: SHIFTREG_ENABLE.vhd

```vhdl
library IEEE;
    use IEEE.std_logic_1164.all;
    use IEEE.std_logic_misc.all;
    use IEEE.std_logic_signed.all;
    use IEEE.std_logic_arith.all;

entity SHIFTREG_ENABLE is
    Port (
                    CLOCK : In      std_logic;
                    RESET : In      std_logic;
                    QK : In         std_logic_vector (7 downto 0);
                    Q : InOut       std_logic_vector (31 downto 0);
                    en0: In std_logic;
                    en1: In std_logic;
                    en2: In std_logic;
                    en3: In std_logic
        );
end SHIFTREG_ENABLE;

architecture BEH_SHIFTREG_ENABLE of SHIFTREG_ENABLE is
        component REG is
        port(
                D : in std_logic_vector(7 downto 0);
                Clock, Reset : in std_logic;
                Q : out std_logic_vector(7 downto 0)
                );
        end component REG;

        component MUX is
        port (
                Q0 : in std_logic_vector(7 downto 0);
                Q1 : in std_logic_vector(7 downto 0);
                enable: in std_logic;
                Qmux : out std_logic_vector(7 downto 0)
                );
        end component MUX;

        signal Qout0, Qout1, Qout2, Qout3 : std_logic_vector(7 downto 0);
        begin
        m1: MUX port map (Q(31 downto 24), QK, en0, Qout0);
        m2: MUX port map (Q(23 downto 16), QK, en1, Qout1);
        m3: MUX port map (Q(15 downto 8), QK, en2, Qout2);
        m4: MUX port map (Q(7 downto 0), QK, en3, Qout3);

        r1: REG port map (Qout0, Clock, Reset, Q(31 downto 24));
        r2: REG port map (Qout1, Clock, Reset, Q(23 downto 16));
        r3: REG port map (Qout2, Clock, Reset, Q(15 downto 8));
        r4: REG port map (Qout3, Clock, Reset, Q(7 downto 0));
end BEH_SHIFTREG_ENABLE;

configuration CFG_SHIFTREG_enable_SCHEMATIC of SHIFTREG_ENABLE is
    for BEH_SHIFTREG_ENABLE
    end for;
end CFG_SHIFTREG_enable_SCHEMATIC;
```

Listing 3: SHIFTREG_GATED.vhd

```vhdl
library IEEE;
    use IEEE.std_logic_1164.all;
    use IEEE.std_logic_misc.all;
    use IEEE.std_logic_signed.all;
    use IEEE.std_logic_arith.all;

entity SHIFTREG_GATED is
      Port (    CLK : In      std_logic;
                RESET : In      std_logic;
                   QK : In      std_logic_vector (7 downto 0);
                    Q : Out     std_logic_vector (31 downto 0) );
end SHIFTREG_GATED;

architecture BEH_SHIFTREG_GATED of SHIFTREG_GATED is
    component Counter is
    port(
        clock:     in std_logic;
        clear: in std_logic;
            Qc:    out std_logic_vector(1 downto 0)
    );
    end component Counter;

        component REG is
        port(
                D : in std_logic_vector(7 downto 0);
                Clock, Reset : in std_logic;
                Q : out std_logic_vector(7 downto 0)
                );
        end component REG;

    component DECODER is
        port(     I:       in std_logic_vector(1 downto 0);
                  O:       out std_logic_vector(3 downto 0)
            );
    end component DECODER;

        signal out_counter : std_logic_vector(1 downto 0);
        signal out_decoder : std_logic_vector(3 downto 0);
        begin

        c1: Counter port map (CLK, Reset, out_counter);

    d1: DECODER port map (out_counter, out_decoder);

        r1: REG port map (QK, out_decoder(3), Reset, Q(31 downto 24));
        r2: REG port map (QK, out_decoder(2), Reset, Q(23 downto 16));
        r3: REG port map (QK, out_decoder(1), Reset, Q(15 downto 8));
        r4: REG port map (QK, out_decoder(0), Reset, Q(7 downto 0));

end BEH_SHIFTREG_GATED;

configuration CFG_SHIFTREG_GATED_SCHEMATIC of SHIFTREG_GATED is
    for BEH_SHIFTREG_GATED
    end for;

end CFG_SHIFTREG_GATED_SCHEMATIC;
```

Listing 4: REG.vhd

```vhdl
library IEEE;
   use IEEE.std_logic_1164.all;
   use IEEE.std_logic_misc.all;
   use IEEE.std_logic_signed.all;
   use IEEE.std_logic_arith.all;

   entity REG is
        port(
                D : in std_logic_vector(7 downto 0);
                Clock, Reset : in std_logic;
                Q : out std_logic_vector(7 downto 0));
end entity REG;

architecture BEH_REG of REG is
   begin
   p0: process (Clock, Reset) is
        begin
        if (Reset = '0') then
           Q <= (others => '0');
        elsif rising_edge(clock) then
           Q <= D;
        end if;
   end process p0;
end architecture BEH_REG;
```

Listing 5: MUX.vhd

```vhdl
library IEEE;
   use IEEE.std_logic_1164.all;
   use IEEE.std_logic_misc.all;
   use IEEE.std_logic_signed.all;
   use IEEE.std_logic_arith.all;

   entity MUX is
        port (
                Q0 : in std_logic_vector(7 downto 0);
                Q1 : in std_logic_vector(7 downto 0);
                enable: in std_logic;
                Qmux : out std_logic_vector(7 downto 0)
                );
end entity MUX;

architecture BEH_MUX of MUX is
   begin

   process (Q0,Q1,enable) is
        begin
        if (enable = '0') then
          Qmux <= Q0;
        elsif (enable = '1') then
          Qmux <= Q1;
             else
          Qmux <= (others => '0');
        end if;
   end process;
end architecture BEH_MUX;
```

Listing 6: COUNTER.vhd

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity counter is
port(
        clock:    in std_logic;
        clear: in std_logic;
            Qc:  out std_logic_vector(1 downto 0)
    );
end counter;

architecture beh_counter of counter is

    signal Pre_Q: std_logic_vector(1 downto 0);

begin

    process(clock, clear)
    begin
        if (clear = '0') then
            Pre_Q <= "11";
        elsif (clock='1' and clock'event) then
                Pre_Q <= Pre_Q + "01";
            end if;
    end process;
    Qc <= Pre_Q;
end beh_counter;
```

Listing 7: DECODER.vhd

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity DECODER is
port(   I:       in std_logic_vector(1 downto 0);
        O:       out std_logic_vector(3 downto 0)
);
end DECODER;

architecture BEH_DECODER of DECODER is
begin
    process (I)
    begin
    case I is
            when "00" => O <= "1000";
            when "01" => O <= "0100";
            when "10" => O <= "0010";
            when "11" => O <= "0001";
            when others => O <= "1000";
        end case;

    end process;

end BEH_DECODER;
```

Listing 8: Power Report Design A

```
******************************************
Report  : power
         −analysis_effort low
Design  : SHIFTREG
Version: X−2005.09−SP1
Date    : Fri Nov 16 20:21:52 2007
******************************************




Library(s) Used:

    CORE90GPSVT (File: /cell_libs/cmos090_50a/CORE90GPSVT_SNPS−AVT_2.1/
            SIGNOFF/bc_1.10V_m40C_wc_0.90V_105C/PT_LIB/CORE90GPSVT_NomLeak.db)




Operating Conditions: NomLeak    Library: CORE90GPSVT
Wire Load Model Mode: enclosed

Design              Wire Load Model              Library
––––––––––––––––––––––––––––––––––––––––––––––––––––––––––

SHIFTREG                 area_0to1K              CORE90GPSVT




Global Operating Voltage = 1
Power−specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000 pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1pW




  Cell Internal Power  =   52.5652 uW   (95%)
  Net Switching Power  =    2.6312 uW    (5%)
                          ––––––––––
Total Dynamic Power    =   55.1964 uW  (100%)


Cell Leakage Power     = 773.6685 nW
```

Listing 9: Power Report Design B

```
*****************************************
Report  : power
        −analysis_effort low
Design : SHIFTREG_ENABLE
Version: X−2005.09−SP1
Date    : Fri Nov 16 21:26:47 2007
*****************************************




Library(s) Used:

    CORE90GPHVT (File: /cell_libs/cmos090_50a/CORE90GPHVT_SNPS−AVT_2.1.a/
        SIGNOFF/bc_1.10V_m40C_wc_0.90V_105C/PT_LIB/CORE90GPHVT_NomLeak.db)
    CORE90GPSVT (File: /cell_libs/cmos090_50a/CORE90GPSVT_SNPS−AVT_2.1/
        SIGNOFF/bc_1.10V_m40C_wc_0.90V_105C/PT_LIB/CORE90GPSVT_NomLeak.db)




Operating Conditions: NomLeak    Library: CORE90GPSVT
Wire Load Model Mode: enclosed

Design              Wire Load Model              Library
————————————————————————————————————————————————————————
SHIFTREG_ENABLE            area_0to1K            CORE90GPSVT
MUX_3                      area_0to1K            CORE90GPSVT
MUX_2                      area_0to1K            CORE90GPSVT
MUX_1                      area_0to1K            CORE90GPSVT
MUX_0                      area_0to1K            CORE90GPSVT
REG_3                      area_0to1K            CORE90GPSVT
REG_2                      area_0to1K            CORE90GPSVT
REG_1                      area_0to1K            CORE90GPSVT
REG_0                      area_0to1K            CORE90GPSVT



Global Operating Voltage = 1
Power−specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1pW




  Cell Internal Power  =   42.5321 uW    (91%)
  Net Switching Power  =    4.4528 uW     (9%)
                         _____
Total Dynamic Power    =   46.9849 uW  (100%)


Cell Leakage Power     = 800.4604 nW
```

Listing 10: Power Report Design C

```
*****************************************
Report  :  power
         -analysis_effort  low
Design  :  SHIFTREG_GATED
Version:  X-2005.09-SP1
Date    :  Fri  Nov  16  23:59:48  2007
*****************************************




Library(s)  Used:

    CORE90GPSVT  (File:  /cell_libs/cmos090_50a/CORE90GPSVT_SNPS-AVT_2.1/
          SIGNOFF/bc_1.10V_m40C_wc_0.90V_105C/PT_LIB/CORE90GPSVT_NomLeak.db)
    CORE90GPHVT  (File:  /cell_libs/cmos090_50a/CORE90GPHVT_SNPS-AVT_2.1.a/
          SIGNOFF/bc_1.10V_m40C_wc_0.90V_105C/PT_LIB/CORE90GPHVT_NomLeak.db)




Operating  Conditions:  NomLeak    Library:  CORE90GPSVT
Wire  Load  Model  Mode:  enclosed

Design              Wire  Load  Model                Library
_____

SHIFTREG_GATED          area_0to1K              CORE90GPSVT
counter                 area_0to1K              CORE90GPSVT
DECODER                 area_0to1K              CORE90GPSVT
REG_3                   area_0to1K              CORE90GPSVT
REG_2                   area_0to1K              CORE90GPSVT
REG_1                   area_0to1K              CORE90GPSVT
REG_0                   area_0to1K              CORE90GPSVT




Global  Operating  Voltage  =  1
Power-specific  unit  information  :
    Voltage  Units  =  1V
    Capacitance  Units  =  1.000000 pf
    Time  Units  =  1ns
    Dynamic  Power  Units  =  1mW      (derived  from  V,C,T  units)
    Leakage  Power  Units  =  1pW




  Cell  Internal  Power    =    26.4885  uW    (81%)
  Net  Switching  Power    =     6.1442  uW    (19%)
                          _____
Total  Dynamic  Power     =    32.6327  uW    (100%)


Cell  Leakage  Power      =  834.9294  nW
```