

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 18, 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.

In the next section ??, we discuss three designs for such a converter. In section ??, we simulate the VHDL code for the designs using Modelsim, and verify that all the designs are working correctly. The VHDL code is then synthesized using Design Vision, and a switching activity is produced using VSS simulator. Based on this switching activity, a power report is presented for the synthesized design. These reports are presented in section ?. In section ??, we discuss the results obtained. The last section ?? contains the VHDL code for the designs.

1.1 Authors by Section

- *Markku Eerola*
- *Josep Renard*
- *Rajesh Bachani*

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

2.2 Design B: Register with Enable

2.3 Design C: Register with Clock-Gating

3 Simulation of the designs with Modelsim

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

4 Power Reports from Design Vision and VSS

```
*****
Report : power
        -analysis_effort low
Design : SHIFTRREG
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
-----
SHIFTRREG   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 = 52.5652 uW (95%)
Net Switching Power = 2.6312 uW (5%)
-----
Total Dynamic Power = 55.1964 uW (100%)

Cell Leakage Power = 773.6685 nW
```

Report : power
 -analysis_effort low
 Design : SHIFTRREG_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
SHIFTRREG_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** = mW (derived from V,C,T **units**)
 Leakage Power **Units** = pW

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

Report : power
 -analysis_effort low
 Design : SHIFTRREG_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
SHIFTRREG_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** = 1.000000pf
 Time **Units** = 1ns
 Dynamic Power **Units** = mW (derived from V,C,T **units**)
 Leakage Power **Units** = pW

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

5 Discussion on the Reports

6 Implementation

```
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;
```



```

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 SHIFTRREG_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 SHIFTRREG_ENABLE;

architecture BEHSHIFTRREG_ENABLE of SHIFTRREG_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 BEHSHIFTRREG_ENABLE;

configuration CFG_SHIFTRREG_enable_SCHEMATIC of SHIFTRREG_ENABLE is
    for BEHSHIFTRREG_ENABLE
        end for;
end CFG_SHIFTRREG_enable_SCHEMATIC;

```

```

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 SHIFTRREG_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 SHIFTRREG_GATED;

architecture BEHSHIFTRREG_GATED of SHIFTRREG_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 BEHSHIFTRREG_GATED;

configuration CFG.SHIFTRREG_GATED.SCHEMATIC of SHIFTRREG_GATED is
    for BEHSHIFTRREG_GATED
        end for;
end CFG.SHIFTRREG_GATED.SCHEMATIC;

```

```

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 BEHREG 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 BEHREG;

```

```

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 BEHMUX 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 BEHMUX;

```

```

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;

```

```

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 BEHDECODER 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 BEHDECODER;

```