Informatics and Mathematical Modelling
Technical University of Denmark

02207 : Advanced Digital Design Techniques

Lab 1: Exercise on Synthesis

Group *dt07*

Markku Eerola (s053739)

Rajesh Bachani (s061332)

Josep Renard (s071158)

October 27, 2007

Contents

1 Introduction

The goal of the exercise was to get familiar with the process of synthesis of digital circuits, using special tools for synthesis such as Design Vision. A register-level netlist containing a 24-bit adder is synthesized in the exercise, for different values of the clock time period. The reports concerning the timing constraints, area, power consumption etc. which are generated by the tool are documented in the report.

1.1 Authors by Section

- *Markku Eerola* Behavioral description for the 24-bit adder, a testbench for the adder and the top-level netlist.
- *Josep Renard* Simulation of testbench with Modelsim (for different values of signals in testbench), behavioral description for the 24-bit register.
- *Rajesh Bachani* Synthesis with Design Vision and analysis of the results from synthesis.

2 Behavioral Description for 24-bit Adder

2.1 VHDL code for the 24-bit Adder

The VHDL code for a simple 24-bit adder is provided below.

```vhdl
library IEEE;
use IEEE.std_logic_1164.all, IEEE.numeric_std.all;

entity NBitAdder is
    port (A, B: in std_logic_vector(23 downto 0);
        Cin: in std_logic;
        Sum: out std_logic_vector(23 downto 0);
        Cout: out std_logic);
end entity NBitAdder;

architecture unsgned of NBitAdder is
    signal result: unsigned(24 downto 0);
    signal carry: unsigned(24 downto 0);
    constant zeros: unsigned(23 downto 0) := (others => '0');

begin
    carry <= (zeros & Cin);
    result <= ('0' & unsigned(A)) + ('0' & unsigned(B)) + carry;
    Sum <= std_logic_vector(result(23 downto 0));
    Cout <= result(24);
end architecture unsgned;
```

2.2 Simulation with Modelsim

The behavior of this adder is verified in Modelsim. Following is a testbench for the adder, and also a screenshot for the waveform from the values provided by the testbench.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity TestNBitAdder is
end entity TestNBitAdder;

architecture TestBench_4 of TestNBitAdder is
signal A, B, Sumint : NATURAL;
signal Aslv, Bslv, Sum: std_logic_vector (23 downto 0);
signal Cin, Cout: std_logic;
signal error: BOOLEAN := FALSE;

begin
s0: entity WORK.NBitAdder(unsgned) port map(Aslv, Bslv, Cin, Sum, Cout);
Aslv <= std_logic_vector(to_unsigned(A, 24));
Bslv <= std_logic_vector(to_unsigned(B, 24));
Sumint <= to_integer(unsigned(Cout & Sum));
stim: process is
begin
Cin <= '0';
A <= 44;
B <= 8990;
wait for 5 NS;
A <= 34545;
wait for 5 NS;
Cin <= '1';
wait for 5 NS;
A <= 7840;
wait for 5 NS;
B <= 0;
wait for 5 NS;
Cin <= '0';
wait;
end process;

resp:   process (Cout, Sum) is
begin
error <= (A + B + BIT'POS(to_bit(Cin))) /= Sumint;
end process;
end architecture TestBench_4;
```

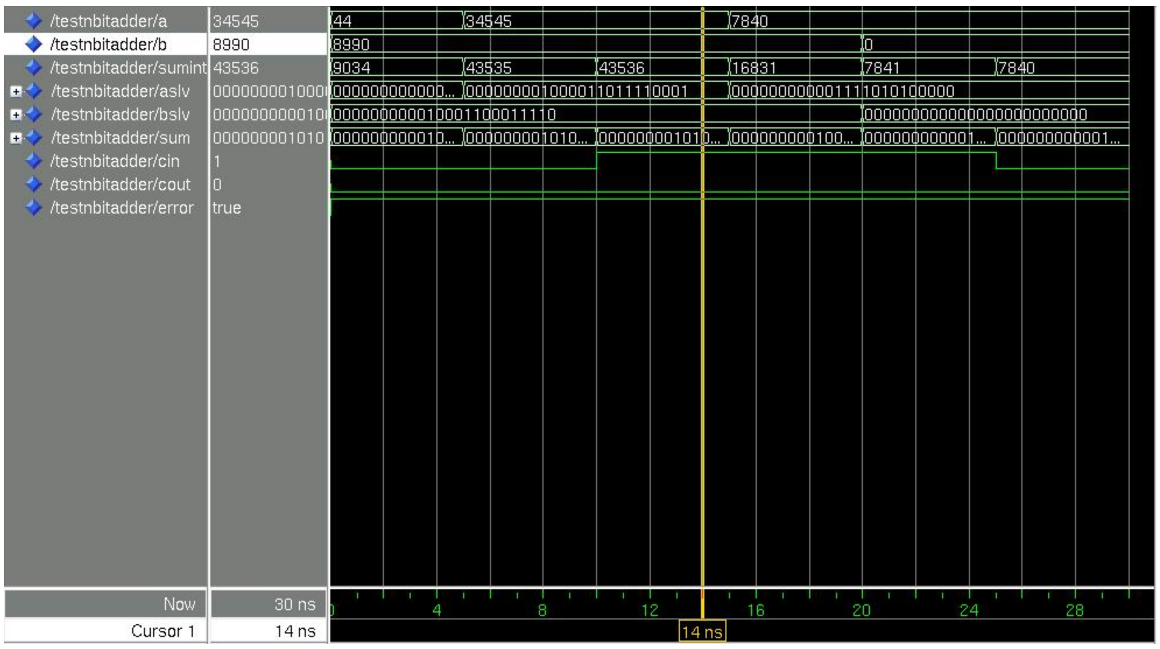

Figure 1: Simulation of the 24-bit adder with Modelsim for the testbench shown

As we can see, the values of a, b and cin at the point of the yellow line are 34545, 8990 and 1. The resulting sum is shown as 43536, which is correct. Even for the other values in the testbench, we can verify the adder is working fine.

3 Behavioral Description for 24-bit Register

The VHDL code for a 24-bit register is provided below.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity reg is
port (D : in std_logic_vector(23 downto 0);
Clock, Reset : in std_logic;
Q : out std_logic_vector(23 downto 0));
end entity reg;

architecture behavioural 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 behavioural;
```

4 Top-Level Netlist from 24-bit Adder and 24-bit Register

The VHDL code for the top-level netlist described in the exercise sheet, is provided below.

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity AdderNetlist is
port( A1 : in std_logic_vector(23 downto 0);
A2 : in std_logic_vector(23 downto 0);
Clock, Reset: in std_logic;
Z : out std_logic_vector( 23 downto 0));
end entity AdderNetlist;

architecture NetlistBehavior of AdderNetlist is
component NBitAdder is
    port (A, B: in std_logic_vector(23 downto 0);
        Cin: in std_logic;
        Sum: out std_logic_vector(23 downto 0);
        Cout: out std_logic);
end component NBitAdder;

component reg is
port (D : in std_logic_vector(23 downto 0);
Clock, Reset : in std_logic;
Q : out std_logic_vector(23 downto 0));
end component reg;

    signal Co, Ci : std_logic;
    signal A1out, A2out, Sum : std_logic_vector(23 downto 0);
    begin
        g1: reg port map (A1, Clock, Reset, A1out);
        g2: reg port map (A2, Clock, Reset, A2out);
        g3: nbitadder port map (A1out, A2out, Ci, Sum, Co);
        g4: reg port map (Sum, Clock, Reset, Z);
end architecture NetlistBehavior;
```

5 Synthesis Results

The synthesis of the top-level netlist described in section **??** is done using Design Vision by Synopsys. We have synthesized the netlist for three different sets of clock time periods, which are 0.5, 1.0 and 2.0 ns. For each of these time periods, we have synthesized the netlist with and without constraints for low power. In the end, we have commented on the results obtained from the data gathered.

The power constraints for low power are 1 uW and 1 pW for maximum dynamic power and maximum leakage power respectively. This is done by using the following two commands (as is also shown in the exercise sheet):

```
set_max_dynamic_power 1
set_max_leakage_power 1
```

For the synthesis in which no constraints are put on power, the values are kept at 10 mW and 30 uW for maximum dynamic power and maximum leakage power respectively.

```
set_max_dynamic_power 10 mW
set_max_leakage_power 30 uW
```

In the following subsections, we give the results and comments of the synthesis for both the above mentioned power constraints. In the next section, we give comments for the change in area and power with respect to the different clock periods.

5.1 Clock Time Period = 0.5 ns

Parameters	**Values**	**Comment**
Dynamic Power	5.91 mW	MET
Leakage Power	11.04 uW	MET
Library Setup Time	0.08 ns	-
Data Arrival Time	0.43 ns	-
SLACK	-0.01 ns	VIOLATED
Combinational Area	2700 um^2	-
Non-Combinational Area	1343 um^2	-
SVT cells	403	-
HVT cells	0	-

Table 1: Clock time-period 0.5 ns, normal power

Synthesis for low power:

Parameters	**Values**	**Comment**
Dynamic Power	5.29 mW	VIOLATED
Leakage Power	9.22 uW	VIOLATED
Library Setup Time	0.08 ns	-
Data Arrival Time	0.44 ns	-
SLACK	-0.02 ns	VIOLATED
Combinational Area	2345 um^2	-
Non-Combinational Area	1343 um^2	-
SVT cells	377	-
HVT cells	0	-

Table 2: Clock time-period 0.5 ns, low power

5.2 Clock Time Period = 1.0 ns

Parameters	**Values**	**Comment**
Dynamic Power	2.76 mW	MET
Leakage Power	11.25 uW	MET
Library Setup Time	0.08 ns	-
Data Arrival Time	0.89 ns	-
SLACK	0.02 ns	MET
Combinational Area	2335 um^2	-
Non-Combinational Area	1343 um^2	-
SVT cells	276	-
HVT cells	0	-

Table 3: Clock time-period 1.0 ns, normal power

Synthesis for low power:

Parameters	**Values**	**Comment**
Dynamic Power	1.71 mW	VIOLATED
Leakage Power	2.98 uW	VIOLATED
Library Setup Time	0.09 ns	-
Data Arrival Time	0.90 ns	-
SLACK	0.00 ns	MET
Combinational Area	1089 um^2	-
Non-Combinational Area	1343 um^2	-
SVT cells	205	-
HVT cells	0	-

Table 4: Clock time-period 1.0 ns, low power

5.3 Clock Time Period = 2.0 ns

Parameters	**Values**	**Comment**
Dynamic Power	0.896 mW	MET
Leakage Power	3.51 uW	MET
Library Setup Time	0.09 ns	-
Data Arrival Time	1.87 ns	-
SLACK	0.05 ns	MET
Combinational Area	631 um^2	-
Non-Combinational Area	1343 um^2	-
SVT cells	97	-
HVT cells	0	-

Table 5: Clock time-period 2.0 ns, normal power

Synthesis for low power:

Parameters	**Values**	**Comment**
Dynamic Power	0.88 mW	MET
Leakage Power	3.17 uW	VIOLATED
Library Setup Time	0.09 ns	-
Data Arrival Time	1.91 ns	-
SLACK	0.00 ns	MET
Combinational Area	614 um^2	-
Non-Combinational Area	1343 um^2	-
SVT cells	97	-
HVT cells	0	-

Table 6: Clock time-period 2.0 ns, low power

5.4 Comments

Comments on synthesis results for low-power:

- For low power, the tool has tried to reduce the power to some extent. In some cases the constraint for the maximum dynamic power is met, while the constraint for maximum leakage power is never met.
- The little reduction in the power with the low power constraints, is achieved by reducing the number of SVT cells in the design. This is because the SVT cells take more power as compared to the HVT cells. But since these cells are faster, reducing their number increases the delay in the circuit, as can be seen by an increase in the time of data arrival for all the three cases.
- The combinational area reduces when synthesizing for low power. This can be explained with the reduction in the number of SVT cells.
- We would have expected to see an increase in the number of HVT cells when synthesis is performed for low power, but it has not been the case. Of course, there is a reason for that. The number of SVT cells determined, are the ones on the critical path. Hence, no SVT cells from the critical path are replaced with HVT cells. Certain SVT cells are replaced with HVT cells, but outside the critical path.
- For the clock time period of 2.0 ns, we note that there is no reduction in the number of SVT cells, when synthesized for low power. The reason for this is follows: for 2.0 ns, the circuit synthesized consists of sequential gates to perform the addition. In this case, there is very little scope of reducing the number of cells on the critical path, since the path is quite linear in flow. But the area is reduced very slightly, since some cells which are not on the critical path, are changed for low power.

Comments on variations observed by changing the clock time-period:

- The design is not synthesized successfully for the clock time period of 0.5 ns. The critical path in this case for the adder is 0.43 ns and adding the register setup and propagation delay time, the total time delay becomes greater than 0.5 ns. However, we see that the circuit is successfully synthesized for timing constraints of 1.0 ns and 2.0 ns.
- The number of cells decreases with the increasing clock time period. So, as the constraint on the time period is reduced, the circuit is synthesized with lower number of cells. This is because the design is moving from parallel to sequential with an increase in clock time period constraint.
- The area for the non-combinational circuit remains the same in all the cases. This is expected since the non-combinational circuit consists of three registers, which would have the same internal design on synthesis. The synthesis is mainly done for the internal design of the 24-bit adder, depending on the constraints of clock time period.
- The area for the combinational circuit decreases with the increasing clock time period. This is because the number of cells decreases in the synthesized circuit.
- The total power, including the dynamic power and the leakage power, decreases as the clocl time period is increased. Again, the reason for this is the decrease in the number of cells.