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 22, 2007

Contents

1 Purpose of the Exercise

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.

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.

```
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.

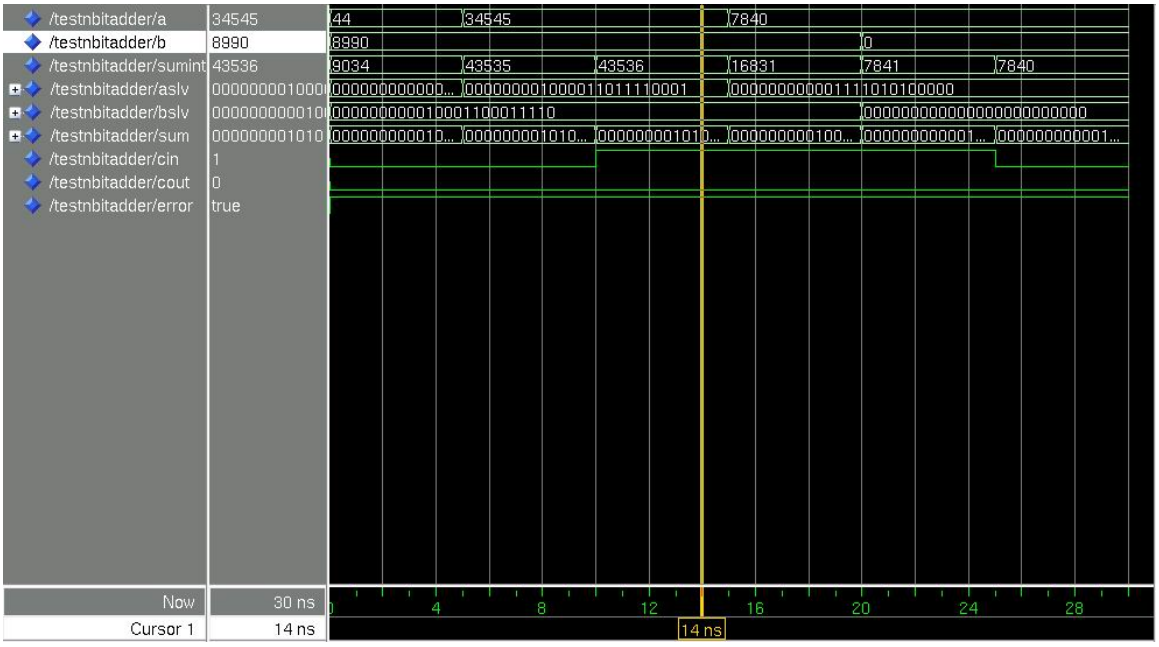
```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

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;
```



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.

```
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 4 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:

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:

Comments:

- For low power, the tool has tried to reduce the power to some extent. But still the constraints are not met.
- The little reduction in the power with the low power constraints, is achieved by reducing the number of SVT cells in the design, which take more power. But since these cells are faster, reducing them increases the delay in the circuit, as can be seen by an increase in the degree of violation of the time-period constraint.
- A time period of 0.5 ns is too less for the tool to synthesize any design.

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:

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:

Comments:

-
-

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:

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:

Comments:

-
-

6 Discussion

- The area for the combinational circuit remains the same in all the cases.
-
-