

Hardwarepraktikum WS 2007/2008

Versuch KOMB3

Gruppe 2

Tobias Doerffel
Heiner Reinhardt
Ken Schmidt

Chemnitz, 27. November 2007

1. Entwerfen Sie einen 8-Bit-Adder. Der Adder bestehe aus zwei Gruppen zu je 4 Bit. Jede Gruppe werde durch einen 4-Bit-Ripple-Carry-Adder realisiert. Zur Beschleunigung der Addition diene eine CLA für die niederwertige Gruppe.

Bauelementebasis: 1 x SN7400 (4 2er-NANDs), 1 x SN7408 (4 2er-ANDs), 2 x SN7410 (6 3er-NANDs), 1 x SN7432 (4 2er-ORs) und 2 x SN7483A (2 4-Bit-Adder).

Das Entwurfsergebnis soll in Form einer Zeichnung und in Form einer ternären VHDL-Strukturbeschreibung vorgelegt werden. Die VHDL-Strukturbeschreibung soll die zutreffenden VHDL-Gatterbeschreibungen aus der Gatterbibliothek verwenden und als UUT zusammen mit der Testbench *DBB2_16.VHD* für die binäre und die ternäre Simulation geeignet sein.

$$g_i = a_i b_i$$

$$p_i = a_i + b_i$$

$$c_4 = g_3 + p_3 c_3$$

$$c_3 = g_2 + p_2 c_2$$

$$c_2 = g_1 + p_1 c_1$$

$$c_1 = g_0 + p_0 c_0$$

$$\Rightarrow c_4 = g_3 + p_3(g_2 + p_2(g_1 + p_1(g_0 + p_0 c_0)))$$

$$\Rightarrow c_4 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_0$$

NAND-Realisierung von c_4 :

$$\begin{aligned} c_4 &= \overline{\overline{g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_0}} \\ &= \overline{\overline{g_3} \cdot \overline{\overline{p_3 g_2}} \cdot \overline{\overline{p_3 p_2 g_1}} \cdot \overline{\overline{p_3 p_2 p_1 g_0}} \cdot \overline{\overline{p_3 p_2 p_1 p_0 c_0}}} \\ &= \overline{\overline{g_3} \cdot \overline{\overline{p_3 g_2}} \cdot \overline{\overline{p_3 p_2 g_1}} \cdot (\overline{\overline{p_3 p_2 p_1}} + \overline{\overline{g_0}}) \cdot (\overline{\overline{p_3 p_2 p_1}} + \overline{\overline{p_0 c_0}})} \\ &= \overline{\overline{g_3} \cdot \overline{\overline{p_3 g_2}} \cdot \overline{\overline{p_3 p_2 g_1}} \cdot (\overline{\overline{\overline{\overline{p_3 p_2 p_1}}}} + \overline{\overline{\overline{\overline{g_0}}}}) \cdot (\overline{\overline{\overline{\overline{p_3 p_2 p_1}}}} + \overline{\overline{\overline{\overline{p_0 c_0}}}})} \\ &= \overline{\overline{g_3} \cdot \overline{\overline{p_3 g_2}} \cdot \overline{\overline{p_3 p_2 g_1}} \cdot (\overline{\overline{\overline{\overline{p_3 p_2 p_1}}}} \cdot \overline{\overline{\overline{\overline{g_0}}}}) \cdot (\overline{\overline{\overline{\overline{p_3 p_2 p_1}}}} \cdot \overline{\overline{\overline{\overline{p_0 c_0}}}})} \\ &= \overline{\overline{g_3} \cdot \overline{\overline{p_3 g_2}} \cdot \overline{\overline{p_3 p_2 g_1}} + (\overline{\overline{\overline{\overline{p_3 p_2 p_1}}}} \cdot \overline{\overline{\overline{\overline{g_0}}}}) \cdot (\overline{\overline{\overline{\overline{p_3 p_2 p_1}}}} \cdot \overline{\overline{\overline{\overline{p_0 c_0}}}})} \\ &= \overline{\overline{\overline{\overline{g_3} \cdot \overline{\overline{p_3 g_2}} \cdot \overline{\overline{p_3 p_2 g_1}} + (\overline{\overline{\overline{\overline{p_3 p_2 p_1}}}} \cdot \overline{\overline{\overline{\overline{g_0}}}}) \cdot (\overline{\overline{\overline{\overline{p_3 p_2 p_1}}}} \cdot \overline{\overline{\overline{\overline{p_0 c_0}}}})}}} \\ &= \overline{\overline{\overline{\overline{g_3} \cdot \overline{\overline{p_3 g_2}} \cdot \overline{\overline{p_3 p_2 g_1}} \cdot (\overline{\overline{\overline{\overline{p_3 p_2 p_1}}}} \cdot \overline{\overline{\overline{\overline{g_0}}}}) \cdot (\overline{\overline{\overline{\overline{p_3 p_2 p_1}}}} \cdot \overline{\overline{\overline{\overline{p_0 c_0}}}})}}} \end{aligned}$$

Ternäre VHDL-Strukturbeschreibung:

```
library ieee;
use ieee.std_logic_1164.all;
use work.pack_2.all;

entity uut is
  port( x_fghij : in  x01_vector(20 downto 5);
        z_abcde : out x01_vector(20 downto 5) );
end uut;

architecture structure of uut is

  alias a3 : x01 is x_fghij(13);
  alias a2 : x01 is x_fghij(12);
  alias a1 : x01 is x_fghij(11);
  alias a0 : x01 is x_fghij(10);

  alias b3 : x01 is x_fghij(18);
  alias b2 : x01 is x_fghij(17);
  alias b1 : x01 is x_fghij(16);
  alias b0 : x01 is x_fghij(15);

  alias c0 : x01 is x_fghij(20);

  component sn7400 is -- 4x 2er NAND
    port ( x : in X01_vector (1 to 2);
           y : out X01
         );
  end component;

  component sn7408 is -- 4x 2er AND
    port (x : in X01_vector (1 to 2); y : out X01);
  end component;

  component sn7410 is -- 3x 3er NAND
    port (x : in X01_vector (1 to 3); y : out X01);
  end component;

  component sn7432 is -- 4x 2er OR
    port (x : in X01_vector (1 to 2); y : out X01);
  end component;

  component sn7483A is -- 2x 4BIT ADDER
    port ( c0 : in  x01;
           b,a : in  x01_vector(3 downto 0);
           c4 : out x01;
           s : out x01_vector(3 downto 0)
         );
  end component;
```

```

        signal part_0, part_1, part_2, part_3, part_3_1, part_3_2,
               part_4, part_4_1, part_4_2, part_5 : X01;
        signal OUT_0, OUT_1, NOUT_0, NOUT_1: X01;

        signal p0, p1, p2, p3, g0, g1, g2, g3, c4: X01;

begin

-- propagate-Signale

PMP0: sn7432 port map( x(1)=>a0, x(2)=>b0, y=>p0 );    -- 2er OR
PMP1: sn7432 port map( x(1)=>a1, x(2)=>b1, y=>p1 );
PMP2: sn7432 port map( x(1)=>a2, x(2)=>b2, y=>p2 );
PMP3: sn7432 port map( x(1)=>a3, x(2)=>b3, y=>p3 );

-- generate-Signale

PMG0: sn7408 port map( x(1)=>a0, x(2)=>b0, y=>g0 );    -- 2er AND
PMG1: sn7408 port map( x(1)=>a1, x(2)=>b1, y=>g1 );
PMG2: sn7408 port map( x(1)=>a2, x(2)=>b2, y=>g2 );
PMG3: sn7408 port map( x(1)=>a3, x(2)=>b3, y=>g3 );

-- C4 Partial

CP0: sn7400 port map( x(1)=>g3, x(2)=>g3, y=>part_0 ); -- Negation
CP1: sn7400 port map( x(1)=>p3, x(2)=>g2, y=>part_1 );
CP2: sn7410 port map( x(1)=>p3, x(2)=>p2, x(3)=>g1, y=>part_2 );

CP3_1: sn7410 port map( x(1)=>p3, x(2)=>p2, x(3)=>p1, y=>part_3_1
);
CP3_2: sn7400 port map( x(1)=>part_3_1, x(2)=>part_3_1,
y=>part_3_2 ); -- Negation
CP3_0: sn7410 port map( x(1)=>part_3_2, x(2)=>g0, x(3)=>g0,
y=>part_3 );

CP4_0: sn7410 port map( x(1)=>part_3_2, x(2)=>p0, x(3)=>c0,
y=>part_4 );

-- C4 Complete

PC0: sn7410 port map( x(1)=>part_0, x(2)=>part_1, x(3)=>part_2,
y=>OUT_0 );
PC0_1: sn7400 port map( x(1)=>OUT_0, x(2)=>OUT_0, y=>NOUT_0 );
--Negation

PCC: sn7410 port map( x(1)=>NOUT_0, x(2)=>part_3, x(3)=>part_4,
y=>c4 );

```

```

-- ADDER
ADD_L: sn7483A port map (      c0 => c0,
                                a(0) => a0,
                                a(1) => a1,
                                a(2) => a2,
                                a(3) => a3,

                                b(0) => b0,
                                b(1) => b1,
                                b(2) => b2,
                                b(3) => b3,

                                s => z_abcde(14 downto 11)
);

ADD_H: sn7483A port map (      c0 => c4,
                                a(0) => '0',
                                a(1) => '0',
                                a(2) => '0',
                                a(3) => '0',

                                b(0) => '1',
                                b(1) => '1',
                                b(2) => '1',
                                b(3) => '1',
                                c4 => z_abcde(20),
                                s => z_abcde(18 downto 15)
);

end structure;

```

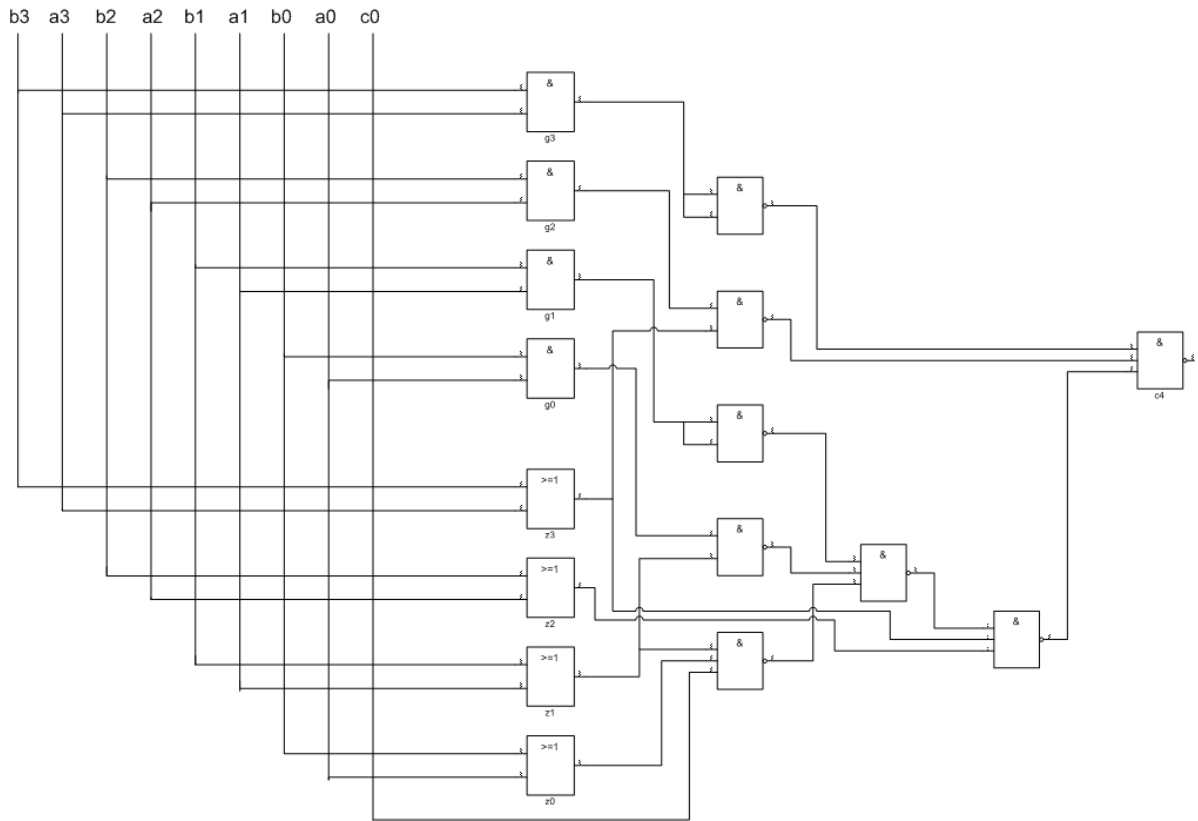


Abbildung 1: Schaltplan der NAND-Realisierung

2. *Geben Sie eine Stimulusfolge minimaler Länge an, die alle SA0- und alle SA1-Fehler an den Eingängen der CLA am Ausgang c_8 beobachtbar macht, und führen Sie die binäre Simulation mit dieser Stimulusfolge durch.*

Minimale Stimulusfolge zur Erkennung von SA0/1-Fehlern:

```

stimmapp dbb2_16 0-1111-0000-----|0-11111111-----
stimmapp dbb2_16 0-0000-1111-----|0-11111111-----
stimmapp dbb2_16 1-1111-0000-----|1-00000000-----
stimmapp dbb2_16 1-0000-1111-----|1-00000000-----

```