

Hardwarepraktikum WS 2007/2008

## **Versuch KOMB1**

Gruppe 2

Tobias Doerffel  
Heiner Reinhardt  
Ken Schmidt

Chemnitz, 30. Oktober 2007

1. Entwerfen Sie eine kombinatorische Schaltung, die die BOOLEsche Funktion  $y = f(a, b, c, d)$  auf zweierlei Weise realisiert. Die BOOLEsche Funktion sei durch eine Wahrheitstabelle beschrieben.

a	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
b	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
c	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
d	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
d	0	0	1	0	1	0	0	0	1	1	1	0	1	0	1	1

Gesucht sind eine minimale NOR-NOR-Realisierung und eine minimale NAND-NAND-Realisierung. Bauelementbasis: 1×SN74LS00 (4 2er-NAND-Gatter), 1×SN74LS27 (3 3er NOR-Gatter), 1×SN74LS260 (2 5er-NOR-Gatter), 1×SN74LS10 (3 3er-NAND-Gatter), 1×SN74LS20 (2 4er-NAND-Gatter).

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\_08.VHD für die binäre und ternäre Simulation geeignet sein.

### Karnaugh-Plan:

	$\bar{A}$	$A$	$A$	$A$	
$\bar{B}$	0	0	1	0	$D$
$B$	1	0	1	1	$\bar{D}$
$B$	0	1	1	1	$\bar{D}$
$B$	0	0	0	1	$D$
	$C$	$\bar{C}$	$\bar{C}$	$C$	

Zusammenfassung ergibt DNF:

$$\bar{b}\bar{c}\bar{d} + b\bar{c}\bar{d} + \bar{a}\bar{b}c + abc = \overline{\overline{\bar{b}\bar{c}\bar{d} + b\bar{c}\bar{d} + \bar{a}\bar{b}c + abc}} = \overline{\bar{b}\bar{c}\bar{d} \cdot b\bar{c}\bar{d} \cdot \bar{a}\bar{b}c \cdot abc}$$

KNF:

$$\bar{y} = (\bar{a} + b + c)(b + \bar{c} + d)(\bar{b} + c + d)(\bar{a} + \bar{b} + \bar{c}) = \overline{\overline{(\bar{a} + b + c)(b + \bar{c} + d)(\bar{b} + c + d)(\bar{a} + \bar{b} + \bar{c})}}$$

$$y = \overline{(a + \bar{b} + \bar{c}) + (\bar{b} + c + \bar{d}) + (b + \bar{c} + \bar{d}) + (a + b + c)}$$

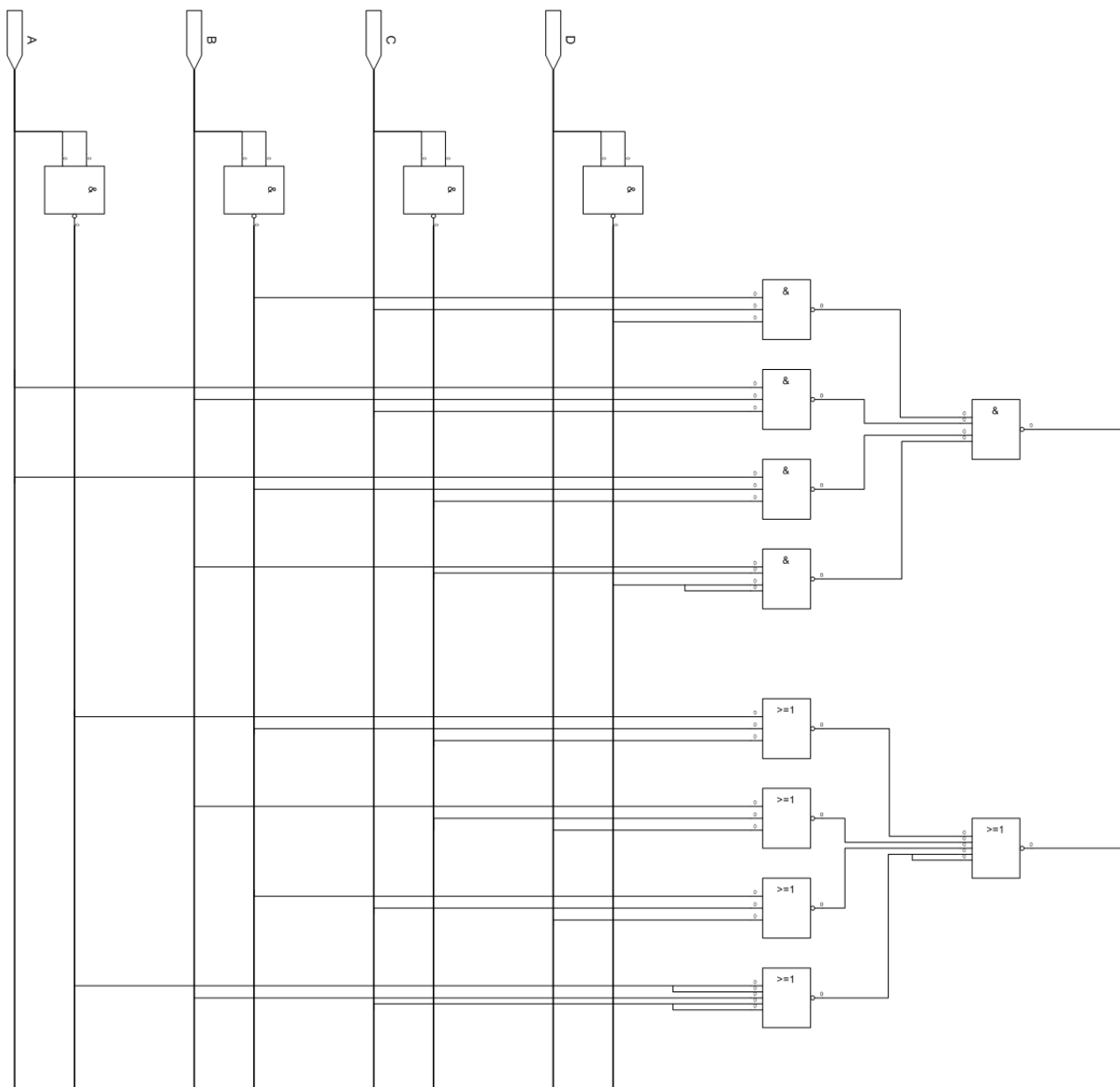


Abbildung 1: Schaltbild der NAND-NAND- und NOR-NOR-Realisierung

```

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

entity uut is
  port (x_fghij : in  X01_vector(20 downto 13);
        z_abcde : out X01_vector(20 downto 13));
end uut;

architecture structure of uut is

  alias a : x01 is x_fghij(20);
  alias b : x01 is x_fghij(19);
  alias c : x01 is x_fghij(18);
  alias d : x01 is x_fghij(17);

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

  component sn7427 -- 3NOR
    port (x : in  X01_vector (1 to 3);
          y : out X01);
  end component;

  component sn74260 -- 5NOR
    port (x : in  X01_vector (1 to 5);
          y : out X01);
  end component;

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

  component sn7420 -- 4NAND
    port (x : in  X01_vector (1 to 4);
          y : out X01);
  end component;

  signal not_a, not_b, not_c, not_d :X01;
  signal nand_out :X01_vector (1 to 4);
  signal nor_out  :X01_vector (1 to 4);

```

```

begin
  -- BLOCK 1 NEGATION

  NA:      sn7400  port map (x(1)=>a,x(2)=>a,y=>not_a);
  NB:      sn7400  port map (x(1)=>b,x(2)=>b,y=>not_b);
  NC:      sn7400  port map (x(1)=>c,x(2)=>c,y=>not_c);
  ND:      sn7400  port map (x(1)=>d,x(2)=>d,y=>not_d);

  -- BLOCK 2 NOR/NOR

  NOR_1:   sn7427  port map
    (x(1)=>a,x(2)=>not_b,x(3)=>not_c,y=>nor_out(1));
  NOR_2:   sn7427  port map
    (x(1)=>not_b,x(2)=>c,x(3)=>not_d,y=>nor_out(2));
  NOR_3:   sn7427  port map
    (x(1)=>b,x(2)=>not_c,x(3)=>not_d,y=>nor_out(3));
  NOR_4:   sn74260 port map
    (x(1)=>a,x(2)=>b,x(3)=>c,x(4)=>a,x(5)=>b,y=>nor_out(4));

  NOR_RESULT: sn74260 port map (x(1 to
    4)=>nor_out,x(5)=>nor_out(1),y=>z_abcde(20));

  -- BLOCK 3 NAND/NAND

  NAND_1:  sn7410  port map
    (x(1)=>c,x(2)=>not_b,x(3)=>not_d,y=>nand_out(1));
  NAND_2:  sn7410  port map
    (x(1)=>a,x(2)=>not_b,x(3)=>not_c,y=>nand_out(2));
  NAND_3:  sn7410  port map
    (x(1)=>b,x(2)=>not_c,x(3)=>not_d,y=>nand_out(3));
  NAND_4:  sn7420  port map
    (x(1)=>a,x(2)=>b,x(3)=>c,x(4)=>a,y=>nand_out(4));

  NAND_RESULT: sn7420  port map (x=>nand_out,y=>z_abcde(19));

end structure;

```

2. Entwerfen Sie eine vollständige binäre Stimulusfolge und leiten Sie daraus eine ternäre Stimulusfolge für die Simulation nach EICHELBERGER ab.

**binäre Stimulusfolge:**

```

stimmap dbb2_08 0000----|00-----
stimmap dbb2_08 0001----|00-----
stimmap dbb2_08 0010----|11-----
stimmap dbb2_08 0011----|00-----
stimmap dbb2_08 0100----|11-----
stimmap dbb2_08 0101----|00-----
stimmap dbb2_08 0110----|00-----
stimmap dbb2_08 0111----|00-----
stimmap dbb2_08 1000----|11-----
stimmap dbb2_08 1001----|11-----
stimmap dbb2_08 1010----|11-----
stimmap dbb2_08 1011----|00-----
stimmap dbb2_08 1100----|11-----
stimmap dbb2_08 1101----|00-----
stimmap dbb2_08 1110----|11-----
stimmap dbb2_08 1111----|11-----

```

**ternäre Stimulusfolge:**

```

stimmap dbb2_08 0000----|00-----
stimmap dbb2_08 000X----|-----
stimmap dbb2_08 0001----|00-----
stimmap dbb2_08 00XX----|-----
stimmap dbb2_08 0010----|11-----
stimmap dbb2_08 001X----|-----
stimmap dbb2_08 0011----|00-----
stimmap dbb2_08 0XXX----|-----
stimmap dbb2_08 0100----|11-----
stimmap dbb2_08 010X----|-----
stimmap dbb2_08 0101----|00-----
stimmap dbb2_08 01XX----|-----
stimmap dbb2_08 0110----|00-----
stimmap dbb2_08 011X----|-----
stimmap dbb2_08 0111----|00-----
stimmap dbb2_08 XXXX----|-----
stimmap dbb2_08 1000----|11-----
stimmap dbb2_08 100X----|-----
stimmap dbb2_08 1001----|11-----
stimmap dbb2_08 10XX----|-----
stimmap dbb2_08 1010----|11-----
stimmap dbb2_08 101X----|-----
stimmap dbb2_08 1011----|00-----
stimmap dbb2_08 1XXX----|-----

```

```
stimmap dbb2_08 1100----|11-----  
stimmap dbb2_08 110X----|-----  
stimmap dbb2_08 1101----|00-----  
stimmap dbb2_08 11XX----|-----  
stimmap dbb2_08 1110----|11-----  
stimmap dbb2_08 111X----|-----  
stimmap dbb2_08 1111----|11-----
```