Hardwarepraktikum WS 1997/98

Versuch 3

Kombinatorische Systeme II

Jan Horbach, 17518
Chris Hübsch, 17543
Lars Jordan, 17560

Aufgabe 1

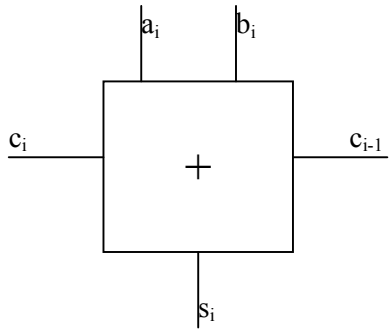
Gegenstand der Aufgabe 1 ist der Schaltplan der ALU SN 74 LS 181.
Es ist zu untersuchen, wie sich die Schaltung unter folgenden Bedingungen verhält:

- active high data
- arithmetische Addition (M = 0, S0 = 1, S1 = 0, S2 = 0, S3 = 1)

Vorbetrachtung:

In der Regel können Zahlen mit einem Ripple-Carry-Adder addiert werden Dabei wird ein einlaufendes Carry an dem Volladder für die Bitstelle 0 angelegt. In Abhängigkeit von den anderen dort anliegenden Werten wird in dieser Stufe neben der Summe ein Carry gebildet, welches dann an den Adder für die Bitstelle 1 der zu addierenden Zahlen angelegt wird. Auf diese Art und Weise der Zusammenschaltung können n-Bit-Ripple-Carry-Adder gebildet werden.

Aufbau eines Volladders

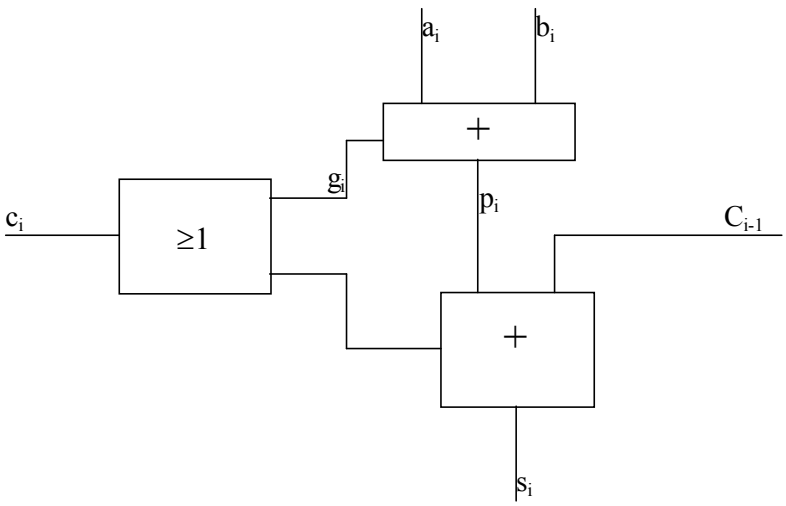


Wenn für die Erzeugung des Carrys in einer Stelle des n-Bit-Ripple-Carry-Adders eine Zeit t_{vadd} benötigt wird, dann liegt das letzte Carry C_n frühestens nach einer Zeit:

$$t_{add} = n * t_{vadd}$$

an. Um dies zu beschleunigen, werden aus dem Volladder an jeder Bitstelle jeweils zwei Signale p_i und g_i herausgeführt:

Struktur eines Volladders aus Halbaddern



Der Carry an der Stelle i kann nun durch folgende Gleichung beschrieben werden:

$$c_i = g_i \vee p_i c_{i-1}$$

Dabei entspricht das g_i der Bedingung, unter der ein Carry in Abhängigkeit von den Eingangswerten an der Stelle i generiert wird, und das p_i stellt die Bedingung dar, unter der ein anliegendes c_{i-1} auf das c_i durchgereicht wird. Für p_i und g_i gelten folgende Gleichungen:

$$g_i = a_i b_i \qquad p_i = a_i \oplus b_i$$

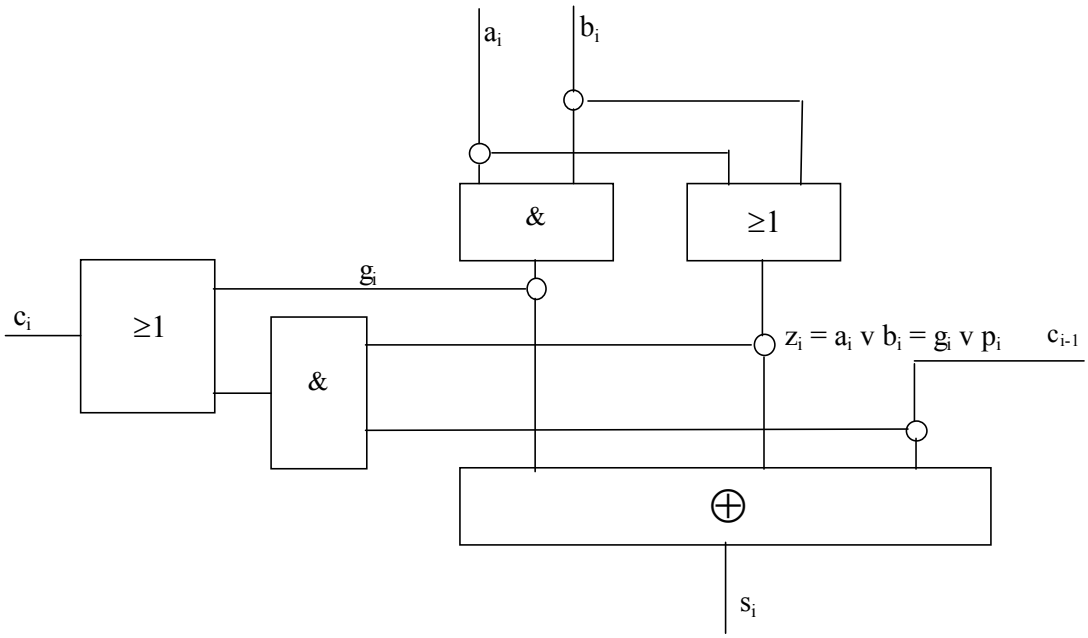
sodaß sich für das Carry ergibt:

$$c_i = a_i b_i \vee (a_i \oplus b_i) c_{i-1}$$

Mit Hilfe dieser Signale läßt sich nun das Carry an der Stelle n des Adders wie folgt berechnen:

$$c_0 = g_0 \vee p_0 \wedge c_{-1}$$
$$c_1 = g_1 \vee p_1 c_0 = g_1 \vee p_1 (g_0 \vee p_0 c_{-1}) = g_1 \vee p_1 g_0 \vee p_1 p_0 c_{-1}$$
$$\vdots$$
$$c_n = \bigcup_{i=-1}^{n} \left(g_i \bigcap_{j=i+1}^{n} p_j \right) \text{ mit } g_{-1} = p_{-1}$$

Eine andere Methode, das Carry an einer bestimmten Stelle zu ermitteln ist die, daß man sich des schon erwähnten Signales g_i und eines anderen Signales z_i bedient.



Dann lautet die Gleichung für das Carry wie folgt:

$$c_i = g_i \vee z_i c_{i-1}$$
$$c_i = a_i b_i \vee (a_i \vee b_i) c_{i-1}$$

Das Carry an einer Stelle n wird dann gebildet durch

$$c_0 = g_0 \vee z_0 c_{-1}$$
$$c_1 = g_1 \vee z_1 c_0 = g_1 \vee z_1 (g_0 \vee z_0 c_{-1}) = g_1 \vee z_1 g_0 \vee z_1 z_0 c_{-1}$$
$$\vdots$$
$$c_n = \bigcup_{i=-1}^{n} \left(g_i \bigcap_{j=i+1}^{n} z_j \right) \text{ mit } g_{-1} = c_{-1}$$

Für die Summe s_i einer Addition gilt stets:

$$s_i = a_i \oplus b_i \oplus c_{i-1}$$

Analyse

Aus der Analyse des Schaltplanes des SN 74 LS 181 ergeben sich für die internen z-Signale folgende Gleichungen

$$z_0 = \overline{(a_0 \vee b_0)} \qquad z_1 = \overline{(a_0 b_0)}$$
$$z_2 = \overline{(a_1 \vee b_1)} \qquad z_3 = \overline{(a_1 b_1)}$$
$$z_4 = \overline{(a_2 \vee b_2)} \qquad z_5 = \overline{(a_2 b_2)}$$
$$z_6 = \overline{(a_3 \vee b_3)} \qquad z_7 = \overline{(a_3 b_3)}$$

Die z_i mit den geraden Indizes sind dabei gleichzusetzen mit den oben genannten z-Signalen. Sie liefern die Bedingung, unter der ein anliegendes Carry c_{i-1} an einem Volladder weitergereicht wird zum ausgehenden Carry c_i. Die z_i mit den ungeraden Indizes stellen die Generate-Bedingung an den einzelnen Volladdern dar. Es ist zu beobachten, daß die internen Signale alle negiert vorliegen.
In der Schaltung sind desweiteren w-Signale zu finden. Für diese gilt die Gleichung:

$$w_i = z_{2i} \oplus z_{2i+1}$$
$$w_i = \overline{(a_i \vee b_i)} \oplus \overline{(a_i b_i)} = a_i \oplus b_i$$

Diese Signale entsprechen den Ergebnissen eines Halbadders, der a_i und b_i addiert.
Die letzten vorhandenen internen Signale, die es zu analysieren galt, waren die c_i . Die Gleichungen für diese Signale lauten wie folgt:

$$c_{-1} = c_n$$

$$c_0 = \overline{\left(z_0 \vee z_1 \overline{c_n}\right)}$$

$$c_1 = \overline{\left(z_3 z_1 \overline{c_n} \vee z_3 z_0 \vee z_2\right)} = \overline{\left(z_2 \vee z_3 \left(z_0 \vee z_1 \overline{c_n}\right)\right)} = \overline{\left(z_2 \vee z_3 \overline{c_0}\right)}$$

$$c_2 = \overline{\left(z_5 z_3 z_1 \overline{c_n} \vee z_5 z_3 z_0 \vee z_5 z_2 \vee z_4\right)} = \overline{\left(z_5 \left(z_3 \left(z_1 \overline{c_n} \vee z_0\right) \vee z_2\right) \vee z_4\right)} = \overline{\left(z_5 \overline{c_1} \vee z_4\right)}$$

$$\overline{c_{n+4}} = z_7 z_5 z_3 z_1 \overline{c_n} \vee z_7 z_5 z_3 z_0 \vee z_7 z_5 z_2 \vee z_7 z_4 \vee z_6 = z_7 \left(z_5 \left(z_3 \left(z_1 \overline{c_n} \vee z_0\right) \vee z_2\right) \vee z_4\right) \vee z_6$$

$$= z_7 \overline{c_2} \vee z_6$$

Das entspricht bis auf die zusätzliche Negation genau den theoretischen Betrachtungen zur parallelen Bildung eines Carrys. C_{n+4} ist aber kein internes Signal mehr, sondern das abschließende Carry.
Außerdem werden die folgenden Leitungen nach außen geführt:

$$F_i = w_i \oplus c_{i-1} = a_i \oplus b_i \oplus c_{i-1}$$

$$X = \overline{\left(z_7 z_5 z_3 z_1\right)}$$

$$Y = \overline{\left(z_7 z_5 z_3 z_0 \vee z_7 z_5 z_2 \vee z_7 z_4 \vee z_6\right)}$$

Dabei stellen die F_i das abschließende Ergebnis dar. Das X ist eine Konjunktion aus den internen z-Signalen mit ungeraden Indizes, d.h es wird nur high, wenn in allen Bitstellen ein Carry generiert wird. Man kann es also als das "globale" Generate-Signal des SN 74 LS 181 betrachten. Das Y entspricht dem "globalen" z (was etwa mit einem Propagate vergleichbar ist).

Damit ist die Analyse des SN 74 LS 181 abgeschlossen. Er realisiert unter den vorausgesetzten Bedingungen einen 4-Bit-Adder mit CLA-Logik, d.h. das ausgehende Carry wird nicht erst nach 4 1-Bit-Additionen geliefert, sondern es kann sofort nachdem die Eingangswerte anliegen (eigentlich nach einer bestimmten Verzögerungszeit, die von der Schalttiefe der ALU abhängt) berechnet werden.

Voraussetzung für die Aufagben 2 und 3

Um die in den Aufgaben 2 und 3 gefragten Schaltungen zu realisieren, wurde der SN 74 LS 181 vorgegeben, deren VHDL-Beschreibung nachfolgend zu sehen ist

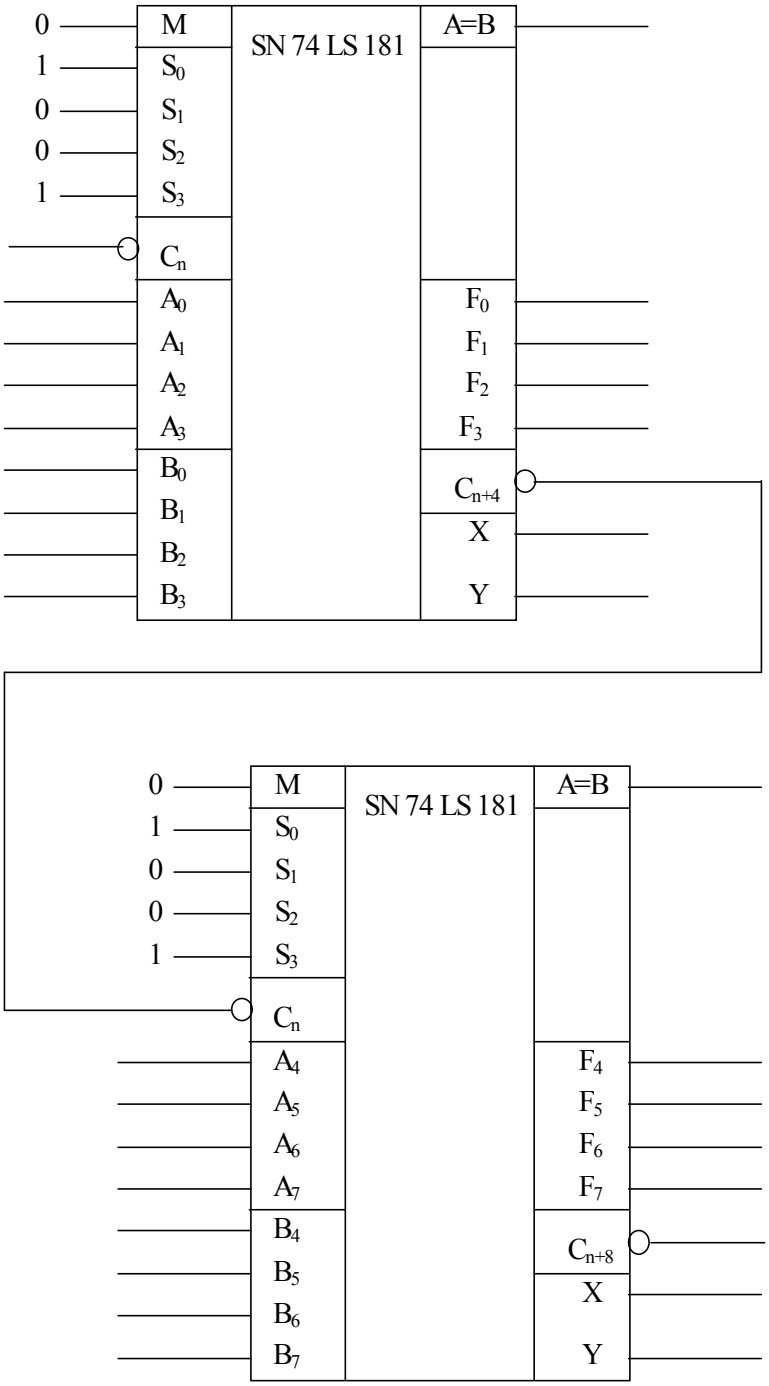
```
entity SN74LS181 is
 port (M: in bit;
       S: in bit_vector(0 to 3);
       Cn: in bit;
       A: in bit_vector(0 to 3);
       B: in bit_vector(0 to 3);

       AB: out bit;
       F: out bit_vector (0 to 3);
       Cn4: out bit;
       X,Y: out bit);
 end SN74LS181;

architecture DATENFLUSS of SN74LS181 is

  signal Z0, Z1, Z2, Z3, Z4, Z5, Z6, Z7: bit;
  signal W0, W1, W2, W3: bit;
  signal C_1, C0, C1, C2: bit;
  signal F0_i, F1_i, F2_i, F3_i: bit;
  signal Y_i: bit;

begin

  Z0 <= not (A(0) or (B(0) and S(0)) or (S(1) and not B(0)));
  Z1 <= not ((not B(0) and S(2) and A(0)) or (A(0) and S(3) and B(0)));
  Z2 <= not (A(1) or (B(1) and S(0)) or (S(1) and not B(1)));
  Z3 <= not ((not B(1) and S(2) and A(1)) or (A(1) and S(3) and B(1)));
  Z4 <= not (A(2) or (B(2) and S(0)) or (S(1) and not B(2)));
  Z5 <= not ((not B(2) and S(2) and A(2)) or (A(2) and S(3) and B(2)));
  Z6 <= not (A(3) or (B(3) and S(0)) or (S(1) and not B(3)));
  Z7 <= not ((not B(3) and S(2) and A(3)) or (A(3) and S(3) and B(3)));

  W0 <= Z0 xor Z1;
  W1 <= Z2 xor Z3;
  W2 <= Z4 xor Z5;
  W3 <= Z6 xor Z7;

  C_1 <= not (Cn and not M);
  C0 <= not ((not M and Z0) or (not M and Z1 and Cn));
  C1 <= not ((not M and Z2) or (not M and Z0 and Z3) or (not M and Z1 and
        Z3 and Cn));
  C2 <= not ((not M and Z4) or (not M and Z2 and Z5) or (not M and Z0 and
        Z3 and Z5) or (not M and Z1 and Z3 and Z5 and Cn));

  F0_i <= C_1 xor W0; F(0) <= F0_i;
  F1_i <= C0 xor W1; F(1) <= F1_i;
  F2_i <= C1 xor W2; F(2) <= F2_i;
  F3_i <= C2 xor W3; F(3) <= F3_i;

  AB <= F0_i and F1_i and F2_i and F3_i;

  X <= not (Z1 and Z3 and Z5 and Z7);
  Y_i <= not ((Z0 and Z3 and Z5 and Z7) or (Z2 and Z5 and Z7) or
         (Z4 and Z7) or Z6); Y <= Y_i;

  Cn4 <= (not Y_i) or (Cn and Z1 and Z3 and Z5 and Z7);

end DATENFLUSS;
```

Aufgabe 2

Durchführung

Die Aufgabenstellung lautet, einen 8-Bit-Ripple-Carry-Adder aus zwei 4er-Gruppen zu entwerfen. Dabei sollten der ein- sowie der auslaufende Carry und die a_i und b_i high-active sein.

```
entity ADD8 is
 port (Cn: in bit;
        A: in bit_vector(0 to 7);
        B: in bit_vector(0 to 7);
        F: out bit_vector(0 to 7);
        Cn8: out bit);
end ADD8;

architecture STRUCTWOCLA of ADD8 is
  component SN74LS181
  port (M: in bit;
        S: in bit_vector(0 to 3);
        Cn: in bit;
        A: in bit_vector(0 to 3);
        B: in bit_vector(0 to 3);

        AB: out bit;
        F: out bit_vector (0 to 3);
        Cn4: out bit;
        X,Y: out bit);
  end component;

  for all: SN74LS181 use entity work.SN74LS181(DATENFLUSS);

  signal C4, C8_i, Cn_i:bit;
  signal X0, Y0, X1, Y1, AB_i1, AB_i2: bit; --unbenutzt
  signal S:bit_vector(0 to 3):="1001";
  signal M_i: bit:='0';

begin
  Cn_i <= not Cn;
  u1: SN74LS181 port map (M_i, S, Cn_i, A(0 to 3), B(0 to 3), AB_i1,
      F(0 to 3), C4, X0, Y0);
  u2: SN74LS181 port map (M_i, S, C4, A(4 to 7), B(4 to 7), AB_i2,
      F(4 to 7), C8_i, X1, Y1);
  Cn8 <= not C8_i;

end;
```


0
1
0
0
1
M
S0
S1
S2
S3
Cn
A0
A1
A2
A3
B0
B1
B2
B3
SN 74 LS 181
A=B
F0
F1
F2
F3
Cn+4
X
Y
0
1
0
0
1
M
S0
S1
S2
S3
Cn
A4
A5
A6
A7
B4
B5
B6
B7
SN 74 LS 181
A=B
F4
F5
F6
F7
Cn+8
X
Y

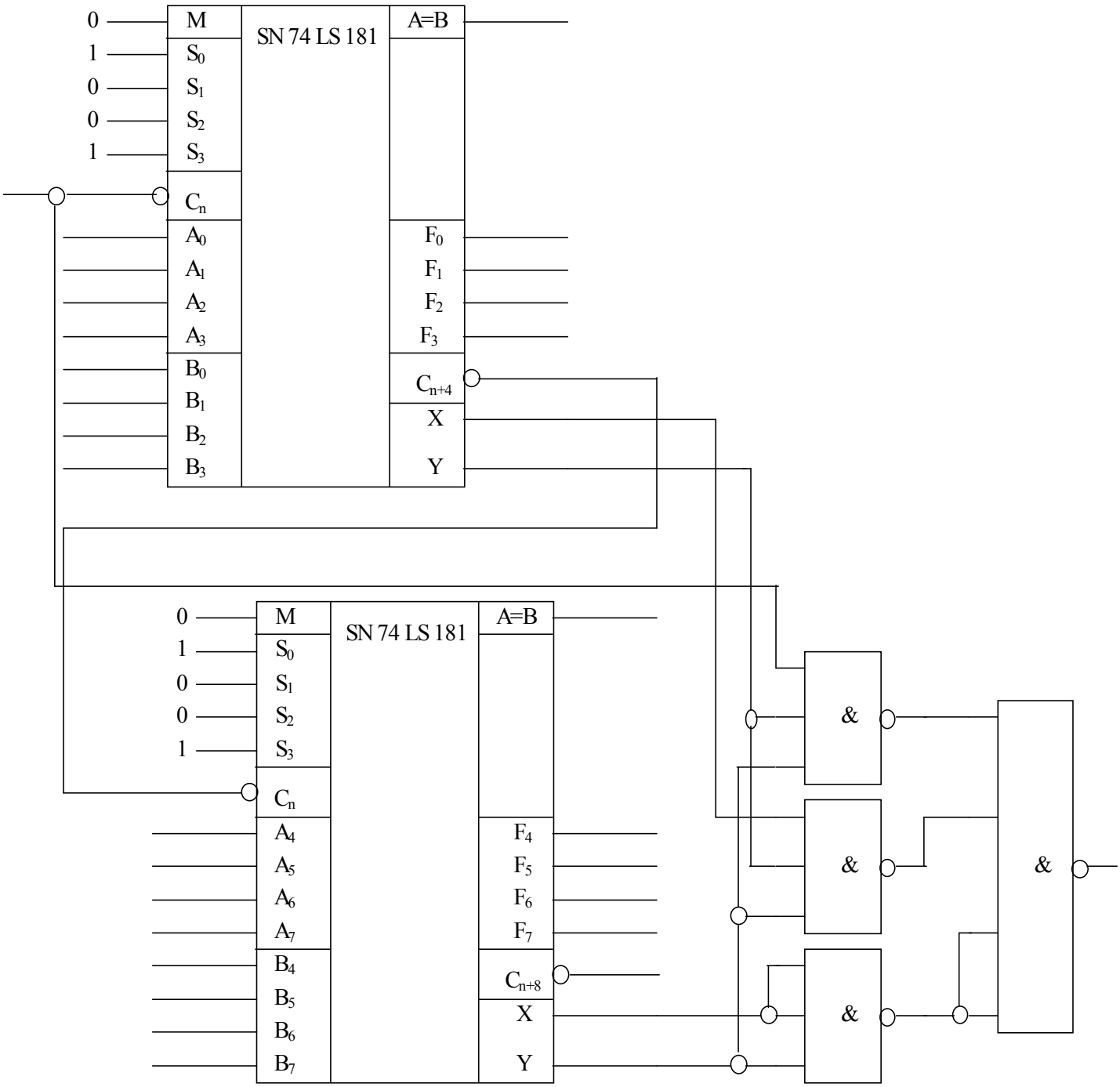
Aufgabe 3

Durchführung

Die Aufgabenstellung lautet, eine CLA-Logik für den 8-Bit-Ripple-Carry-Adder zu entwerfen.
Funktion der CLA:

$$c_{n+8} = \overline{x_1 y_1 \wedge y_0 y_1 x_0 \wedge y_0 y_1 c_{-1}}$$

```
entity CLA is
  port (Cn: in bit;
        X0, Y0, X1, Y1: in bit;
        Cn8: out bit);
end CLA;

architecture DATENFLUSS of CLA is
begin
  Cn8 <= (X1 and Y1) or (Y0 and Y1 and X0) or (Y0 and Y1 and Cn);
end DATENFLUSS;

entity ADD8 is
  port (Cn: in bit;
        A: in bit_vector(0 to 7);
        B: in bit_vector(0 to 7);
        F: out bit_vector(0 to 7);
        Cn8: out bit);
end ADD8;

architecture STRUCTCLA of ADD8 is
  component SN74LS181
  port (M: in bit;
        S: in bit_vector(0 to 3);
        Cn: in bit;
        A: in bit_vector(0 to 3);
        B: in bit_vector(0 to 3);
        AB: out bit;
        F: out bit_vector (0 to 3);
        Cn4: out bit;
        X,Y: out bit);
  end component;
  component CLA
  port (Cn: in bit;
        X0, Y0, X1, Y1: in bit;
        Cn8: out bit);
  end component;

  for all: SN74LS181 use entity work.SN74LS181(DATENFLUSS);
  for all: CLA use entity work.CLA(DATENFLUSS);
  signal C4, C8_i, Cn_i:bit;
  signal X0, Y0, X1, Y1, AB_i1, AB_i2: bit;
  signal S:bit_vector(0 to 3):="1001";
  signal M_i: bit:='0';

begin
  Cn_i <= not Cn;
  u1: SN74LS181 port map (M_i, S, Cn_i, A(0 to 3), B(0 to 3), AB_i1,
      F(0 to 3), C4, X0, Y0);
  u2: SN74LS181 port map (M_i, S, C4, A(4 to 7), B(4 to 7), AB_i2,
      F(4 to 7), C8_i, X1, Y1);
  u3: CLA port map (Cn_i, X0, Y0, X1, Y1, Cn8);
end;
```


0
M
SN 74 LS 181
A=B
1
S0
0
S1
0
S2
1
S3
Cn
A0
A1
A2
A3
B0
B1
B2
B3
F0
F1
F2
F3
Cn+4
X
Y
0
M
SN 74 LS 181
A=B
1
S0
0
S1
0
S2
1
S3
Cn
A4
A5
A6
A7
B4
B5
B6
B7
F4
F5
F6
F7
Cn+8
X
Y
&
&
&
&