

Berufsakademie Berlin
- Informatik -
2. Semester

Digitaltechnik

Belegarbeit:

4-Bit-ALU in FPGA Implementierung

Mitglieder der Arbeitsgruppe:

Michael Kuss
Thomas Jünger
Matthias Buchhorn
Andreas Spiller

Inhalt:

1. Aufgabenstellung
2. Tastaturdecoder (ABEL)
3. Register
 - Befehlsregister
 - Operandenregister
 - Hilfsregister
4. Bus
5. ALU-KI-Einheit (ABEL)
6. ALU-Control-Einheit (ABEL)
7. Display

Anhang:

- ABEL Dateien
- Verwendete Macros aus der XILINX-Bibliothek
- Schematik der Gesamtschaltung

1. Aufgabenstellung:

Entwurf einer 4-Bit-ALU in FPGA-Implementierung

Mindestanforderungen:

- 4-Bit-ALU
- 2 Operanden-Register (A=Akku, B)
- Befehls- und Direktwerteingabe über die Tastatur
- Akku-Inhaltsausgabe auf 7-Segmentanzeige permanent
- Datenformat: vorzeichenlose Integerwerte von 0 bis 9 (auch Ergebnis)
- Befehlssatz:

Befehl	Erklärung
mov A,c	lädt die Konstante c von Tastatur in Register A=Akku; A:=const
mov B,c	lädt die Konstante c von Tastatur in Register B; B:=const
add A,B	addiert die Inhalte der Register A und B, Ergebnis wird nach A=Akku zurückgeschrieben; A:=(A+B)
sub A,B	subtrahiert den Inhalt des Reg. B vom Inhalt des Reg. A, Ergebnis wird nach A zurückgeschrieben ; A:=(A-B)
inc A	inkrementiert den Inhalt von A um 1; A:= (A+1)
dec A	dekrementiert den Inhalt von A um 1; A:= (A-1)
shr A	verschieben des Inhalts von A nach rechts; $a_i:=a_{i+1}$
shl A	verschieben des Inhalts von B nach links; $a_{i+1}:=a_i$

Realisierung / Funktionsnachweis:

- Auf dem XILINX Schaltkreis XC 4003

Entwurfsanforderungen:

- Gesamtschaltung mit dem Schematic-Editor des XILINX Design Managers entwerfen
- Mindestens zwei Teilblöcke in HDL-Eingabe (ABEL)

Geforderte Tastaturbelegung :

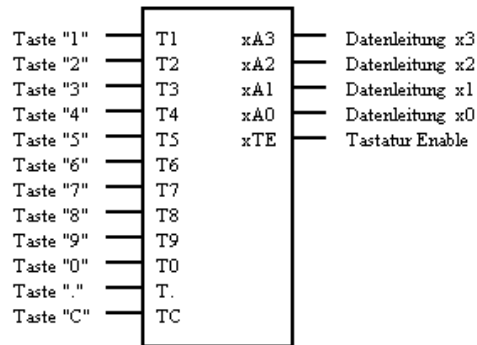
Befehl	Taste
add A,B	3
sub A,B	4
inc A	5
dec A	6
shr	7
shl	8
mov A,c	•
mov B,c	C

Tastatur-Decoder:

(ABEL-Datei siehe Anhang)

Der Tastatur-Decoder setzt die von der Tastatur kommenden Signale (12) in einen Binärcode (4) um.

Schematisches Bild für den Tastaturdecoder



Taste	dezimale Kodierung	binäre Kodierung
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
•	10	1010
C	11	1011

Der TE-Ausgang (Tastatur Enable) am Tastatur-Decoder wird immer dann HI, wenn eine beliebige Taste auf der Tastatur gedrückt wird. Mit diesem Signal wird der CE-Eingang des Befehlsregisters (Typ RD4, siehe Anhang) gesetzt. Der an den Ausgängen (xA3, xA2, xA1, xA0) des Tastatur-Decoders liegende Binärcode wird dann im Befehlsregister gespeichert.

3. Register:

In der Schaltung werden 5 Register verwendet. Basis ist immer das Register vom Typ RD4 aus der XILINX-Bibliothek (siehe Anhang). Es gibt ein Befehlsregister, zwei Operandenregister A (Akku) und B und zwei Hilfsregister für A und B. Die Signale zur Steuerung der Registerfunktionen kommen von der ALU-Control-Einheit.

Befehlsregister (Typ RD4):

Das Befehlsregister speichert die an der Tastatur getätigten Eingaben. Der CE-Eingang muß hierfür auf HI liegen (kommt vom TE-Ausgang des Tastaturdecoders) am C-Eingang muß der Systemtakt anliegen. Die Ausgänge des Befehlsregisters liegen über Tristate-Buffer (TBUF siehe Anhang) am BUS und außerdem direkt an der ALU-Control.

Operandenregister A (Akku) und B:

Die Operandenregister speichern die Operanden für die arithmetischen und logischen Funktionen der ALU. Im Register A werden auch die Ergebnisse der Berechnungen gespeichert. Die Eingänge beider Register sind direkt auf den BUS geschaltet. Die Ausgänge sind an BCD-Decoder angeschlossen, so daß der Inhalt des Registers auf einer 7-Segment-Anzeige dargestellt werden kann. Außerdem sind die Ausgänge über Tristate-Buffer (TBUF siehe Anhang) auf den BUS geschaltet.

Hilfsregister:

Die Hilfsregister für die Register A und B speichern den Inhalt der Operandenregister. Die Eingänge der Hilfsregister sind direkt auf den BUS geschaltet, die Ausgänge führen direkt zur ALU-KI (Rechenwerk). So wird gewährleistet, das innerhalb eines Zustandes beide Operanden verfügbar sind.

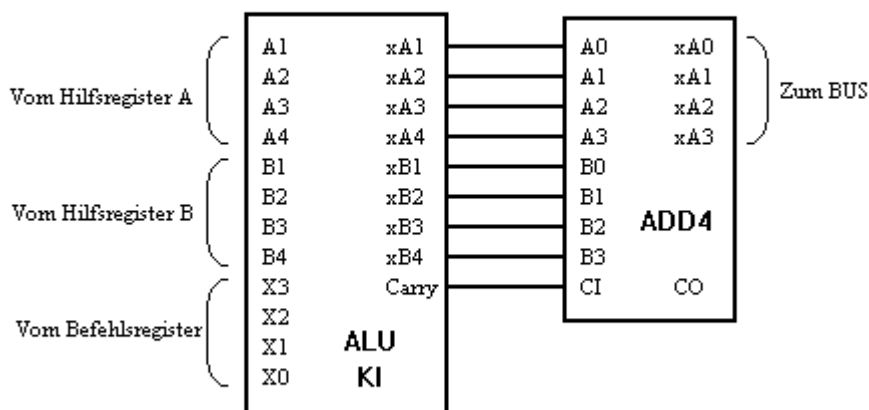
4. Bus:

Der BUS besteht aus vier parallelen Leitungen (4-Bit Busbreite), die alle Funktionsgruppen miteinander verbinden. Damit nicht mehrere Funktionsgruppen gleichzeitig ihre Signale auf den BUS senden (Buskonflikte), sind die Ausgänge der entsprechenden Gruppen mit Tristate-Buffern vom BUS entkoppelt. Die Freischaltung des Ausgangs einer Funktionsgruppe erfolgt durch ein Signal der ALU-Control-Einheit an die Tristate-Buffer. Soll eine Funktionsgruppe (Register) vom BUS lesen, so wird auch hier ein entsprechendes Signal von der ALU-Control benötigt.

5. ALU-KI (Rechenwerk):

Das Rechenwerk besteht aus einer kombinatorischen Schaltung und einem 4-Bit-Adder (ADD4 siehe Anhang) mit Carry-In/Out. Alle Berechnungen werden in eine Addition umgesetzt, die kombinatorische Schaltung wandelt die Operanden in die erforderliche Form um.

Schematisches Bild der ALU (Rechenwerk)



Die folgende Tabelle stellt die Ausgangsvariablen in Abhängigkeit von den Eingangsvariablen dar:

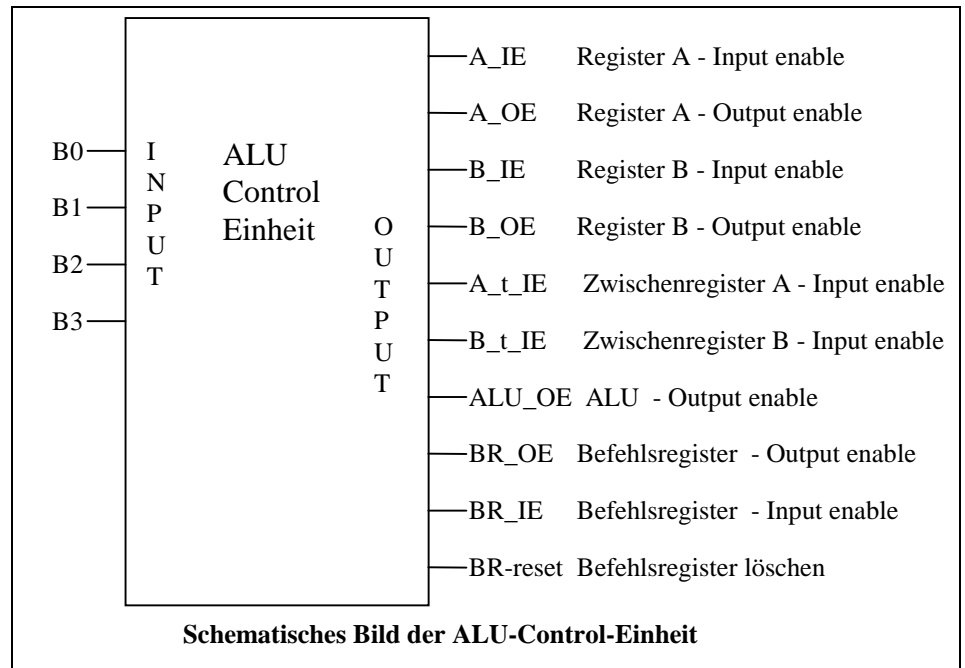
Befehl	Binär	xA4	xA3	xA2	xA1	xB4	xB3	xB2	xB1	CarryIn
add	0011	A4	A3	A2	A1	B4	B3	B2	B1	0
sub	0100	A4	A3	A2	A1	!B4	!B3	!B2	!B1	1
inc	0101	A4	A3	A2	A1	0	0	0	0	1
dec	0110	A4	A3	A2	A1	1	1	1	1	0
shr	0111	0	A4	A3	A2	0	0	0	0	0
shl	1000	A4	A3	A2	A1	A4	A3	A2	A1	0

Aus der Tabelle ergeben sich folgende Gleichungen für die Ausgangszustände:
(Die Kodierung der Befehle in den Gleichungen ist unterstrichen)

$$\begin{aligned}
 \text{xA1} &= \underbrace{(!X3 \& !X2 \& X1 \& X0 \& A1)}_{(!X3 \& X2 \& !X1 \& !X0 \& A1)} \# \underbrace{(!X3 \& X2 \& X1 \& X0 \& A1)}_{(!X3 \& X2 \& X1 \& X0 \& A2)} \# \underbrace{(!X3 \& X2 \& !X1 \& X0 \& A1)}_{(X3 \& !X2 \& !X1 \& !X0 \& A1)} \# \\
 \text{xB1} &= \underbrace{(!X3 \& !X2 \& X1 \& X0 \& B1)}_{(!X3 \& X2 \& X1 \& !X0)} \# \underbrace{(!X3 \& X2 \& !X1 \& !X0 \& !B1)}_{(X3 \& !X2 \& !X1 \& !X0 \& A1)} \# \\
 \text{xA2} &= \underbrace{(!X3 \& !X2 \& X1 \& X0 \& A2)}_{(!X3 \& X2 \& !X1 \& !X0 \& A2)} \# \underbrace{(!X3 \& X2 \& X1 \& X0 \& A2)}_{(!X3 \& X2 \& X1 \& X0 \& A3)} \# \underbrace{(!X3 \& X2 \& !X1 \& X0 \& A2)}_{(X3 \& !X2 \& !X1 \& !X0 \& A2)} \# \\
 \text{xB2} &= \underbrace{(!X3 \& !X2 \& X1 \& X0 \& B2)}_{(!X3 \& X2 \& X1 \& !X0)} \# \underbrace{(!X3 \& X2 \& !X1 \& !X0 \& !B2)}_{(X3 \& !X2 \& !X1 \& !X0 \& A2)} \# \\
 \text{xA3} &= \underbrace{(!X3 \& !X2 \& X1 \& X0 \& A3)}_{(!X3 \& X2 \& !X1 \& !X0 \& A3)} \# \underbrace{(!X3 \& X2 \& X1 \& X0 \& A3)}_{(!X3 \& X2 \& X1 \& X0 \& A4)} \# \underbrace{(!X3 \& X2 \& !X1 \& X0 \& A3)}_{(X3 \& !X2 \& !X1 \& !X0 \& A3)} \# \\
 \text{xB3} &= \underbrace{(!X3 \& !X2 \& X1 \& X0 \& B3)}_{(!X3 \& X2 \& X1 \& !X0)} \# \underbrace{(!X3 \& X2 \& !X1 \& !X0 \& !B3)}_{(X3 \& !X2 \& !X1 \& !X0 \& A3)} \# \\
 \text{xA4} &= \underbrace{(!X3 \& !X2 \& X1 \& X0 \& A4)}_{(!X3 \& X2 \& !X1 \& !X0 \& A4)} \# \underbrace{(!X3 \& X2 \& X1 \& X0 \& A4)}_{(X3 \& !X2 \& !X1 \& !X0 \& A4)} \# \underbrace{(!X3 \& X2 \& !X1 \& X0 \& A4)}_{(X3 \& !X2 \& !X1 \& !X0 \& A4)} \# \\
 \text{xB4} &= \underbrace{(!X3 \& !X2 \& X1 \& X0 \& B4)}_{(!X3 \& X2 \& X1 \& !X0)} \# \underbrace{(!X3 \& X2 \& !X1 \& !X0 \& !B4)}_{(X3 \& !X2 \& !X1 \& !X0 \& A4)} \# \\
 \text{Carry} &= (!X3 \& X2 \& !X1);
 \end{aligned}$$

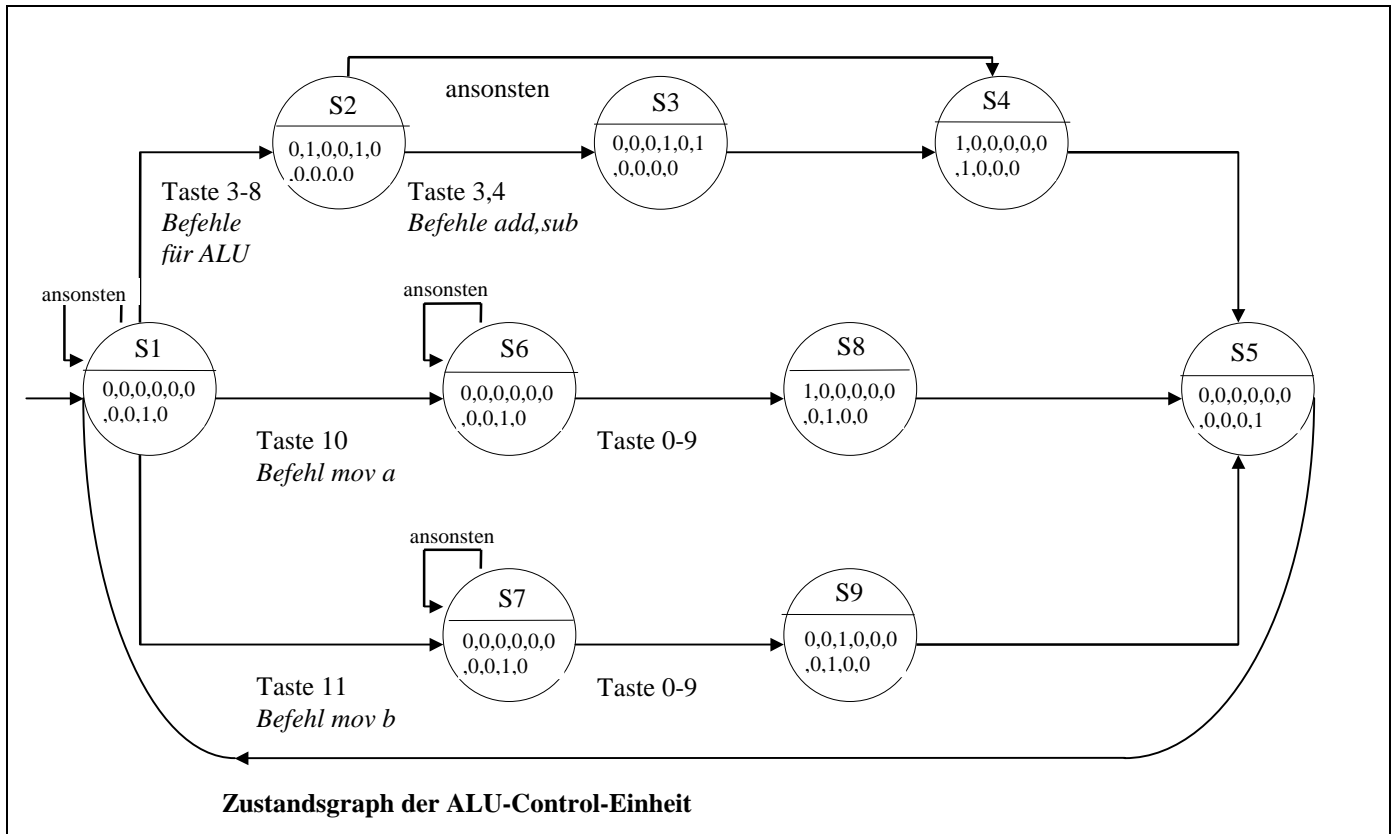
6. ALU-Control-Einheit:

Die Alu-Control-Einheit läßt sich als Zustandsgraph mit neun inneren Zuständen darstellen.
 Jeder dieser inneren Zustände steuert die Ausgänge anders an.
 Mit diesen Signalen (außer BR_IE) werden die jeweiligen Tristatebuffer an den entsprechenden Einheiten angesteuert.
 Diese Tristateeingänge sind low-active, deshalb werden die Ausgänge invertiert.



Ausgänge der ALU-Control-Einheit:

Ausänge	Zustand	nähere Beschreibung
A_IE	S4 OR S8	nehme das auf den Bus liegende Signal in Register A
A_OE	S2	lege Inhalt des Akku auf den Bus
B_IE	S9	nehme das auf den Bus liegende Signal in Register B
B_OE	S3	lege Registerinhalt B auf den Bus
A_t_IE	S2	nehme das auf den Bus ligende Signal ins Zwischenregister A_t
B_t_IE	S3	nehme das auf den Bus ligende Signal ins Zwischenregister B_t
ALU_OE	S4	lege Ergebnis der ALU-Einheit auf den Bus
BR_OE	S8 OR S9	lege Befehlsschlüssel auf den Bus
BR_IE	S1 OR S6 OR S7	Das clock-Signal wird an das Befehlsregister gelegt, sodaß das BR die Daten des Tastaturdecoders übernehmen kann.
BR_reset	S5	Lösche das Befehlsregister



(A_IE , A_OE , B_IE , B_OE , A_t_IE , B_t_IE,
 ALU_OE , BR_OE , BR_IE , BR_reset)

Innere Zustände der ALU-Control-Einheit:

innere Zustände	Bemerkungen
S1	Der clock steuert die Übernahme der Eingabe von dem Tastaturdecoder. die Eingabe wird vom Tastaturdecoder gelesen und in das Befehlsregister geschrieben. Ist die Eingabe 0,1,2,9, so wird wieder in Zustand S1 gesprungen (der Benutzer hat eine unbelegte Eingabe getätigt). Ist die Eingabe 3-8 wird in Zustand S2 gesprungen, da es diejenigen Befehle sind die in der Alu verarbeitet werden. Bei der Taste _/c also (10/11) erreicht das System den Zustand S6 bzw. S7, d.h. die Befehle mov A / mov B werden aktiv.
S2	In diesem Zustand wird der Ausgang des Akkus freigeschaltet, damit das gleichzeitig freigeschaltete Zwischenregister A_t das Signal vom Bus nehmen kann. Wenn ein Befehl eingegeben wurde, der den Akku B benötigt (add A,B; sub A,B), wird in den Zustand S3 gesprungen, ansonsten wird in den Zustand S4 gesprungen.
S3	In diesem Zustand wird der Ausgang des Registers B freigeschaltet, damit das gleichzeitig freigeschaltete Zwischenregister B_t das Signal vom Bus nehmen kann. Danach geht das System in den Zustand S4.
S4	Das Ergebnis der Alu wird auf den Bus gelegt und in den Akku geschrieben. Der nächste Zustand ist S5
S5	Das Befehlsregister wird gelöscht und das System gelangt wieder in den Zustand S1.

- S6 Diesen Zustand benötigt der Befehl mov A (Wert wird in den Akku geschrieben). Das clock-Signal wird wieder von der übrigen Schaltung abgeklemmt und wenn eine Taste 0-9 gedrückt wurde, geht das System in den Zustand S8, ansonsten wird wieder in den Zustand S6 gesprungen und eine neue Eingabe gefordert.
- S7 Diesen Zustand benötigt der Befehl mov B (Wert wird in das Reg. B geschrieben). Das clock-Signal wird wieder von der übrigen Schaltung abgeklemmt und wenn eine Taste 0-9 gedrückt wurde, geht das System in den Zustand S9, ansonsten wird wieder in den Zustand S7 gesprungen und eine neue Eingabe gefordert.
- S8 Das Befehlsregister legt seinen Inhalt auf den Bus und dieses Signal wird in den Akku hineingeschrieben. Das System wechselt in den Zustand S5.
- S9 Das Befehlsregister legt seinen Inhalt auf den Bus und dieses Signal wird in Register B hineingeschrieben. Das System wechselt in den Zustand S5.

7. Display:

Das Display besteht aus zwei 7-Segment-Anzeigen, die permanent die Inhalte der beide Operandenregister A und B anzeigen (dezimal). Sie werden über einen BCD-7-Segment-Umsetzer angesteuert. Zusätzlich wurden LEDs zur Überwachung verschiedener Funktionen (z.B. Takt, Reset-Befehlsregister, OE-Befehlsregister usw.) angesteuert. Auf die Benutzung des Rechners haben sie keinen Einfluß.

Anhang:

ABEL-Datei für den Tastaturdecoder:

```
MODULE Tast_Dec
TITLE 'Tastatur-Decoder'

"inputs
t1,t2,t3,t4,t5,t6,t7,t8,t9,t0,t_,tc PIN;
```

```
"outputs
  xA3, xA2, xA1, xA0, xte          PIN;
```

```
"equation table
```

```
"      I xA3 xA2 xA1 xA0
"      -----
"      t0 I 0 0 0 0
"      t1 I 0 0 0 1
"      t2 I 0 0 1 0
"      t3 I 0 0 1 1
"      t4 I 0 1 0 0
"      t5 I 0 1 0 1
"      t6 I 0 1 1 0
"      t7 I 0 1 1 1
"      t8 I 1 0 0 0
"      t9 I 1 0 0 1
"      t_ I 1 0 1 0
"      tc I 1 0 1 1
```

```
EQUATIONS
```

```
xA3=t8#t9#t_#tc;
xA2=t4#t5#t6#t7;
xA1=t2#t3#t6#t7#t_#tc;
xA0=t1#t3#t5#t7#t9#tc;
xte=t0#t1#t2#t3#t4#t5#t6#t7#t8#t9#t_#tc;
```

```
@DCSET
```

```
END
```

ABEL-Datei für die ALU-KI-Einheit (Rechenwerk):

```
MODULE ALU_KI
TITLE 'ALU Ki'
```

```
"inputs
  A1,A2,A3,A4,B1,B2,B3,B4,
```

```
X3,X2,X1,X0          PIN;

"outputs
  XA1,XA2,XA3,XA4,
  XB1,XB2,XB3,XB4,Carry          PIN;

EQUATIONS
XA1=(!X3&!X2&X1&X0&A1)#(!X3&X2&!X1&!X0&A1)#(!X3&X2&!X1&X0&A1)#
  (!X3&X2&X1&!X0&A1)#(!X3&X2&X1&X0&A2 )#(X3&!X2&!X1&!X0&A1);
XB1=(!X3&!X2&X1&X0&B1)#(!X3&X2&!X1&!X0&!B1)#
  (!X3&X2&X1&!X0) #(X3&!X2&!X1&!X0&A1);

XA2=(!X3&!X2&X1&X0&A2)#(!X3&X2&!X1&!X0&A2)#(!X3&X2&!X1&X0&A2)#
  (!X3&X2&X1&!X0&A2)#(!X3&X2&X1&X0&A3 )#(X3&!X2&!X1&!X0&A2);
XB2=(!X3&!X2&X1&X0&B2)#(!X3&X2&!X1&!X0&!B2)#
  (!X3&X2&X1&!X0) #(X3&!X2&!X1&!X0&A2);

XA3=(!X3&!X2&X1&X0&A3)#(!X3&X2&!X1&!X0&A3)#(!X3&X2&!X1&X0&A3)#
  (!X3&X2&X1&!X0&A3)#(!X3&X2&X1&X0&A4 )#(X3&!X2&!X1&!X0&A3);
XB3=(!X3&!X2&X1&X0&B3)#(!X3&X2&!X1&!X0&!B3)#
  (!X3&X2&X1&!X0) #(X3&!X2&!X1&!X0&A3);

XA4=(!X3&!X2&X1&X0&A4)#(!X3&X2&!X1&!X0&A4)#(!X3&X2&!X1&X0&A4)#
  (!X3&X2&X1&!X0&A4)#(X3&!X2&!X1&!X0&A4);
XB4=(!X3&!X2&X1&X0&B4)#(!X3&X2&!X1&!X0&!B4)#
  (!X3&X2&X1&!X0) #(X3&!X2&!X1&!X0&A4);

Carry=!X3&X2&!X1;

END
```

ABEL-Datei für die ALU-Control-Einheit:

```
MODULE ALU_CTRL
TITLE 'ALU Contrlol Einheit'

"clock
  clock          PIN;

"inputs
  B3,B2,B1,B0          PIN;
```

```
INPUT=[B3,B2,B1,B0];

"outputs
  A_IE, A_OE, B_IE, B_OE, A_t_IE,
  B_t_IE, ALU_OE, BR_OE, BR_IE,
  BR_reset          PIN;

"state diagram decleration
  sbit          STATE_REGISTER istype 'reg_D';
  s1,s2,s3,s4,s5,s6,s7,s8,s9    STATE;

XILINX PROPERTY 'INITIALSTATE s1';

EQUATIONS
  sbit.clk= clock;
  A_IE  = !(s4#s8);
  A_OE  = !(s2);
  B_IE  = !(s9);
  B_OE  = !(s3);
  A_t_IE = !(s2);
  B_t_IE = !(s3);
  ALU_OE = !(s4);
  BR_OE  = !(s8#s9);
  BR_IE  = s1#s6#s7;
  BR_reset= s5;

@DCSET

" C=11 -=10
STATE_DIAGRAM sbit
  STATE s1: if (((INPUT>=0)&(INPUT<=2))#(INPUT==9)) THEN s1
            else if (INPUT==10) THEN s6
            else if (INPUT==11) THEN s7
            else s2;
  STATE s2: if ((INPUT==3)#(INPUT==4)) THEN s3
            else s4;
  STATE s3: goto s4;
  STATE s4: goto s5;
  STATE s5: goto s1;
  STATE s6: if ((INPUT>=0)&(INPUT<=9)) THEN s8
            else s6;
  STATE s7: if ((INPUT>=0)&(INPUT<=9)) THEN s9
            else s7;
  STATE s8: goto s5;
  STATE s9: goto s5;
END
```