9 Versuch Nr. 7

9.1 Anmerkungen zum Versuch Nr. 7

In den letzten drei Versuchen haben Sie die wichtigsten Bestandteile eines Rechners kennen gelernt, in der Software MAX+PlusII eingegeben und in den Baustein EPM7128S programmiert. Sie verfügen nun über das Wissen, um einen kleinen Rechner selbst aufbauen zu können. So soll in diesem Versuch ein kleiner Rechner programmiert und aufgebaut werden. In Abbildung 1 ist der geplante Rechner zu sehen:

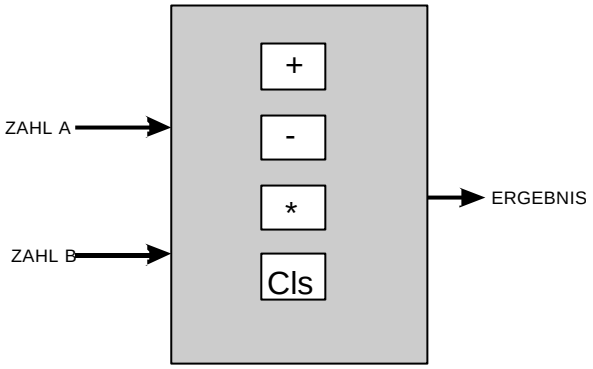

Abbildung 1: Recheneinheit

Dieser kleine "Rechner" soll zwei `Zahlen A und B` addieren (`+`), subtrahieren (`-`) und multiplizieren (`*`) können, außerdem soll das Ergebnis mit der Funktion `cls` auf *`0`* zurückgesetzt werden können. Der Einfachheit halber sollen die Zahlen A und B max. Werte bis *`9`* annehmen können, und das Ergebnis soll max. einen Wert von *`18`* annehmen können.

Der Aufbau der Schaltung geschieht mit dem HWPRAK-Altera-Board, das in diesem Versuch nun aus den folgenden Komponenten besteht:

- Der schon bekannten Leiterplatte mit dem Baustein EPM7128S und einem Programmieranschluss.
- Die Zahlen `A und B`, und die Auswahl der Funktionen werden über BCD-Schalter und ein Tastenfeld auf einer `Eingabeeinheit` eingegeben.
- Das Ergebnis des Rechners wird auf zwei 7-Segmentanzeigen ausgegeben (`Ausgabeeinheit`).
- Die entsprechenden Leitungen zum Rechner sind schon vorgegeben.

Der Aufbau und die Funktion der noch nicht bekannten Einheiten auf dem HWPRAK-Altera-Board sind im Anhang näher beschrieben

9.2 Literaturhinweis zu Versuch Nr. 7

- Tietze, Schenk: Halbleiterschaltungstechnik
- Keller / Paul: Hardwaredesign

9.3 Aufgabenstellung vor Versuchsdurchführung

In Abbildung 2 ist das Blockschalbild des geplanten Rechners zu sehen, der in diesem Versuch programmiert und aufgebaut werden soll.

Mit den `Zahlen A` und `B` soll der Rechner folgende Operationen durchführen können:

- Addieren: `A + B`
- Subtrahieren: `A - B`
- Multiplizieren: `A * B`
- CLS: Ergebnis auf `0` setzen

Die `Zahlen A` und `B` sind zwei 4-Bit-Zahlen, die im BCD-Kode anliegen. Einstellen lassen sich die Zahlen `0 - 9.`

Die Addition/Subtraktion wird mit ein und der selben Einheit durchgeführt (`4-Bit-Addierer-Subtrahierer`). Subtraktion wird ausgewählt, falls das Selektionsbit `sel_sub = 1` ist.

Die Anzahl der Bits des Ergebnisses aus der Multiplikation sind bekannter weise doppelt so groß, wie die `Zahlen A` und `B,` die miteinander multipliziert werden. Deshalb handelt es sich bei dem Multiplizierer um einen `2-Bit-Multiplizierer`, damit das Ergebnis nicht größer als 4 Bit ist. Damit kann man leider nur eine Multiplikation bis hin zu `3 * 3` durchführen!

Die Addition/Subtraktion und Multiplikation wird parallel durchgeführt.

Die Taster für die Auswahl der durchzuführenden Rechenoperation erzeugen nur einen `kurzen 1 - Impuls` beim Drücken der entsprechenden Taste. Deshalb muss eine Steuerlogik einen `stetigen Operationscode (opsel1, opsel0)` erzeugen, der nach dem Drücken der entsprechenden Taste erhalten bleibt, bis zum Drücken der nächsten Taste.

Der Operationscode sei wie folgt definiert:

+	-	*	cls	opsel1	opsel0	Funktion
0	0	0	0	X	X	---
0	0	0	1	0	0	Ergebnis auf Null setzen
1	0	0	0	0	1	Durchführung einer Addition
0	1	0	0	1	0	Durchführung einer Subtraktion
0	0	1	0	1	1	Durchführung einer Multiplikation

Das Ergebnis auf 0 setzen wird durchgeführt, wenn folgende Dinge erfüllt sind:

- Das Selektions-Bit des 4-Bit-Multiplexers Nr. 2 sel_mp2 muss = 1 sein. Damit wird erreicht, dass ein 4-Bit-breiter Bus mit dem Wert 0 auf den Ergebnisausgang (s0, s1, s2, s3) gesetzt wird.
- Das Carry-Ausgangs-Bit (carry_out) muss auch auf 0 gesetzt werden, was mit einem 1-Bit-Multiplexer geschieht, dessen Selektions-Bit sel_carry = 1 ist.

Eine Addition wird durchgeführt, wenn folgende Dinge erfüllt sind:

- Das Selektions-Bit am 4-Bit-Addierer-Subtrahierer sel_sub muss = 0 sein.
- Das Selektions-Bit am 4-Bit-Multiplexer Nr.1 sel_mp1 muss = 0 sein, damit das Ergebnis der Addition an den nächsten Multiplexer gelangt.
- Das Selektions-Bit am 4-Bit-Multiplexer Nr.2 sel_mp2 muss = 0, um das Ergebnis der Addition auf den Ergebnisausgang (s0, s1, s2, s3) zu legen.
- Das Carry-Ausgangs-Bit (carry_out) muss das Carry der Addition sein, welches mit dem 1-Bit-Multiplexer ausgewählt wird, in dem das Selektions-Bit sel_carry = 1 ist.

Eine Subtraktion wird durchgeführt, wenn folgende Dinge erfüllt sind:

- Das Selektions-Bit am 4-Bit-Addierer-Subtrahierer sel_sub muss = 1 sein
- Das Selektions-Bit am 4-Bit-Multiplexer Nr.1 sel_mp1 muss = 0 sein, damit das Ergebnis der Subtraktion an den nächsten Multiplexer gelangt.
- Das Selektions-Bit am 4-Bit-Multiplexer Nr.2 sel_mp2 muss = sein 0, um das Ergebnis der Subtraktion auf den Ergebnisausgang (s0, s1, s2, s3) zu legen.
- Das Carry-Ausgangs-Bit (carry_out) muss = 0 sein, welches mit dem 1-Bit-Multiplexer geschieht, in dem das Selektions-Bit sel_carry = 0 ist.

Eine Multiplikation wird durchgeführt, wenn folgende Dinge erfüllt sind:

- Das Selektions-Bit am 4-Bit-Multiplexer Nr.1 sel_mp1 muss = 1 sein, damit das Ergebnis der Multiplikation an den nächsten Multiplexer gelangt.
- Das Selektions-Bit am 4-Bit-Multiplexer Nr.2 sel_mp2 muss = 0 sein, um das Ergebnis der Multiplikation auf den Ergebnisausgang (s0, s1, s2, s3) zu legen.
- Das Carry-Ausgangs-Bit (carry_out) muss = 0 sein, welches mit dem 1-Bit-Multiplexer geschieht, in dem das Selektions-Bit sel_carry = 0 ist.

Die Selektionssignale sel_sub, sel_mp1, sel_mp2 und sel_carry sollen aus dem Operationscode (opsel1, opsel0) erzeugt werden (sel_sub-Bit-Erzeugung, sel_mp1-Bit-Erzeugung, sel_mp2-Bit-Erzeugung, sel_carry-Bit-Erzeugung)

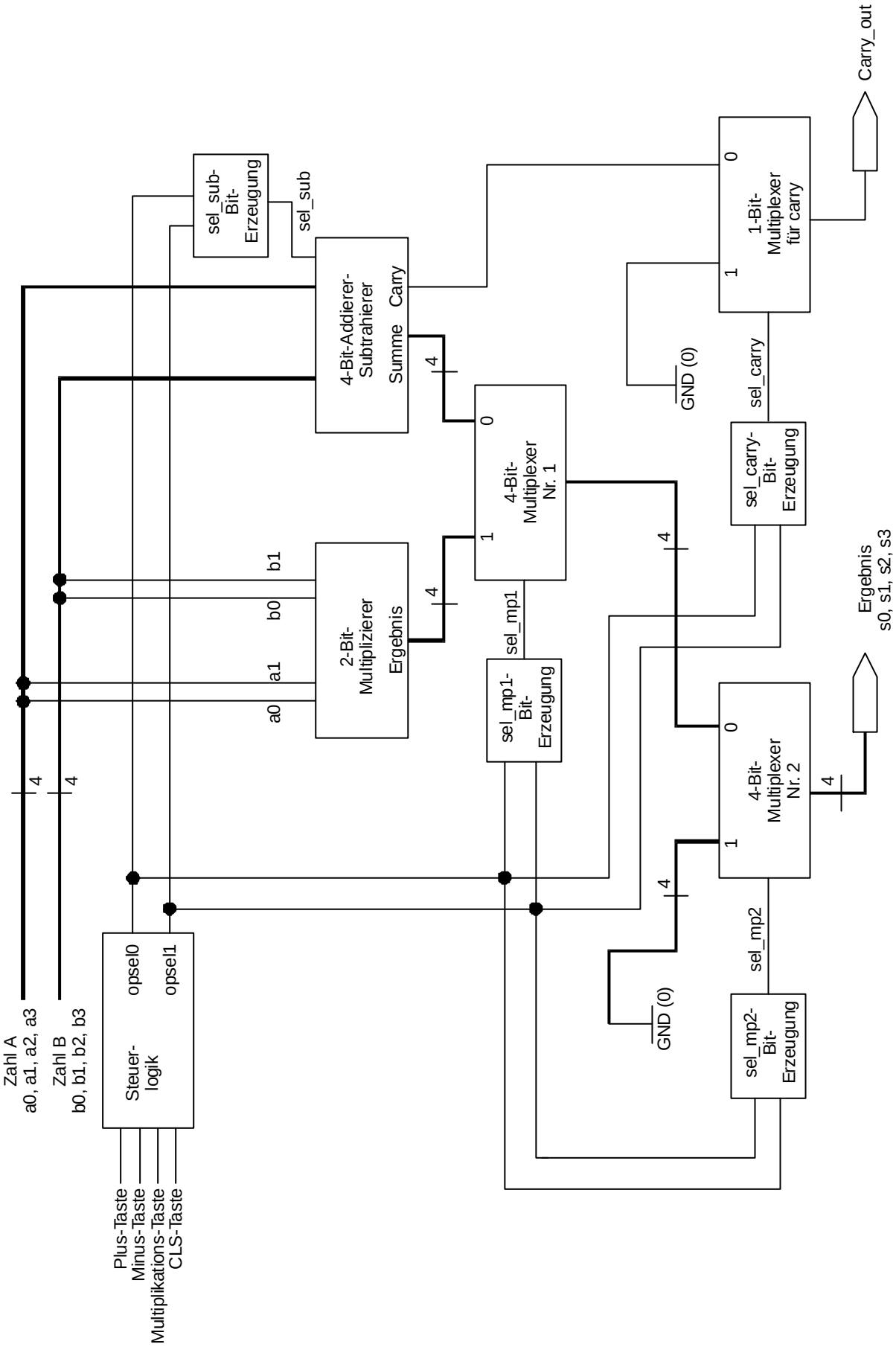

Abbildung 2: Blockschaltbild des zu programmierenden Rechners

9.3.1

- Entwerfen Sie den 4-Bit-Addierer/Subtrahier aus Abbildung 2, der einen 4-Bit-Carry-Chain-Addierer enthalten soll. Mit einem Bit `sel_sub = 1`, soll die Subtraktion ausgewählt werden können. Skizzieren Sie den entsprechenden Schaltkreis.
- Skizzieren Sie ein mögliches Zeitdiagramm (Vorlage für ein SCF-File), um den Addierer/Subtrahierer simulieren zu können.

9.3.2

Entwerfen Sie den 2-Bit-Multiplizierer, der nach Schulmethode rechnet:

Für alle $i \in \{0,...,n-1\}$ multipliziert man den `Multiplikanden a` mit jeder Stelle b_i `des Multiplikators`, schiebt das Zwischen-`Ergebnis` e_i um i Stellen nach links, und addiert schließlich die `Zwischen-Ergebnisse` e_i zum `Gesamtergebnis`.

Beispiel:

Multiplikation von zwei 4-Bitzahlen:

a (a3,a2,a1,a0)= 1110 (14)
b (b3,b2,b1,b0)= 0110 (6)
Ergebnis = 84

e0 = a * b0	1110 * 0	0	0	0	0	**0**	**0**	**0**	**0**
e1 = a * b1	1110 * 1	0	0	0	**1**	**1**	**1**	**0**	0
e2 = a * b2	1110 * 1	0	0	**1**	**1**	**1**	**0**	0	0
e3 = a * b3	1110 * 0	0	**0**	**0**	**0**	**0**	0	0	0
Ergebnis		0	1	0	1	0	1	0	0

Es ist leicht erkennbar, wie die `Zwischen-Ergebnisse` e_i aufgebaut sind:

$$e_i = (\underbrace{0,.....0,0}_{\text{Anzahl der 0en: n-i}},b_i \wedge a_{n-1},.....,b_i \wedge a_0,\underbrace{0,.....,0}_{\text{Anzahl der 0en: i}})$$

- Entwerfen Sie zunächst die beiden Schaltkreise zur Berechnung der Zwischenergebnisse `e0` und `e1`. Skizzieren Sie beide Schaltkreise.
- Entwerfen Sie nun den 2-Bit-Multiplizierer, unter Verwendung eines 4-Bit-Carry-Chain-Addierers. Skizzieren Sie den Schaltkreis.
- Skizzieren Sie ein mögliches Zeitdiagramm (Vorlage für ein SCF-File), um den Multiplizierer simulieren zu können.

9.3.3

- Entwerfen Sie einen Schaltkreis zur Erzeugung der Steuersignale `opsel0` und `opsel 1 (Steuerlogik)`, die auf Grund der Tasten wie weiter oben beschrieben aussehen sollen. Der Schaltkreis soll aus einem selbst konstruierten RS-Flip-Flop, bestehend aus NAND-Gattern, aufgebaut sein. Skizzieren Sie den Schaltkreis der Steuerlogik.
- Skizzieren Sie ein mögliches Zeitdiagramm (Vorlage für ein SCF-File), um die Steuerlogik simulieren zu können.

9.3.4

- Entwerfen Sie die Schaltkreise `sel_sub-Bit-Erzeugung, sel_mp1-Bit-Erzeugung, sel_mp2-Bit-Erzeugung und sel_carry-Bit-Erzeugung, welche die` Selektionssignale `sel_sub, sel_mp1, sel_mp2` und `sel_carry` aus dem Operationscode `(opsel1, opsel0)` erzeugen sollen, wie dies oben beschrieben wurde. Skizzieren Sie die Schaltkreise.

9.4 Versuchsdurchführung

Gehen Sie nun bei dem Aufbau Ihres Rechners wie folgt vor:

9.4.1

Aus den vorherigen Versuchen ist die Komponente Carry-Chain-Addierer für eine Breite von 8 Bit bekannt.

- Wandeln Sie das GDF-File des 8-Bit-Carry-Chain-Addierers in einen 4-Bit-Carry-Chain-Addierer ab. Erzeugen Sie sich von dem 4-Bit-Carry-Chain-Addierer ein Symbol, um diesen hinterher in weiteren Schaltkreisentwürfen einsetzen zu können.

- Kompilieren Sie Ihr Designs für das Device EPM7128SLC-10.

- Geforderte Dokumentation:

    - Eingegebener Schaltkreis des 4-Bit-Carry-Chain-Addierers (GDF-File)

9.4.2

Aus den vorherigen Versuchen ist die Komponente Multiplexer für eine Breite von 8 Bit bekannt.

- Wandeln Sie das GDF-File des 8-Bit-Multiplexers in einen 4-Bit-Multiplexer ab. Erzeugen Sie sich von dem 4-Bit-Multiplexer ein Symbol, um diesen hinterher in weiteren Schaltkreisentwürfen einsetzen zu können.

- Wandeln Sie das GDF-File des 8-Bit-Multiplexers in einen 1-Bit-Multiplexer ab. Erzeugen Sie sich von dem 1-Bit-Multiplexer ein Symbol, um diesen hinterher in weiteren Schaltkreisentwürfen einsetzen zu können.

- Kompilieren Sie Ihr Designs für das Device EPM7128SLC-10.

- Geforderte Dokumentation:

    - Eingegebener Schaltkreise der Multiplexer (GDF-File)

- Kompilieren Sie beide Schaltkreise für das Device EPM7128SLC-10.

- Geben Sie nun den von Ihnen entworfenen Multiplizierer als GDF-File ein. Verwenden Sie Ihren zuvor erzeugten 4-Bit-Carry-Chain-Addierer, und die Schaltkreise zur Berechnung des Zwischenergebnisses. Erzeugen Sie sich von dem Multiplizierer ein Symbol, um diesen hinterher in weiteren Schaltkreisentwürfen einsetzen zu können.

- Kompilieren Sie Ihr Design für das Device EPM7128SLC-10.

- Erstellen Sie ein SCF-File, um den Multiplizierer simulieren zu können. Simulieren Sie Multiplilzierer , und prüfen Sie Ihr Ergebnis.

- Geforderte Dokumentation:

    - Eingegebene Schaltkreise (GDF-File)
    - Das SCF-File der Simulation

9.4.5

- Geben Sie die von Ihnen entworfene Steuerlogik als GDF-File ein. Geben Sie hierzu zunächst ein RS-Flip-Flop bestehend aus NAND-Gattern als GDF-File ein, das Sie in Ihrer Steuerlogik verwenden soll. Erzeugen Sie sich von der Steuerlogik ein Symbol, um diese hinterher in weiteren Schaltkreisentwürfen einsetzen zu können.

- Kompilieren Sie Ihr Design für das Device EPM7128SLC-10.

- Erstellen Sie ein SCF-File, um die Steuerlogik simulieren zu können. Simulieren Sie schließlich die Steuerlogik , und prüfen Sie Ihr Ergebnis.

- Geforderte Dokumentation:

    - Eingegebener Schaltkreis (GDF-File)
    - Das SCF-File der Simulation

9.4.6

- Geben Sie nun den kompletten Rechner aus Abbildung 2 als GDF-File unter der Verwendung der zuvor erzeugten Symbole ein. Da es sich, wie Sie sicherlich herausgefunden haben, bei den Schalkreisen zur Erzeugung der Selektionssignale `sel_sub`, `sel_mp1`, `sel_mp2` und `sel_carry` nur um einzelne Gatter handelt, ergänzen Sie diese nun direkt in Ihrem GDF-File für den Rechner.

- Kompilieren Sie Ihr Design für das Device EPM7128SLC-10, und betrachten Sie sich das Ergebnis im Floorplan-Editor.

- Die willkürlich getroffenen I/O-Belegungen des Compilers sollen wie folgt im Floorplan-Editor abgeändert werden:

Pinbezeichnung	**I/O-Port**
Summenbits s0	70
s1	69
s2	68
s3	67
Carry-Bit	63
Zahl A a0	33
a1	34
a2	35
a3	36
Zahl B b0	48
b1	49
b2	50
b3	51
Funktion +	27
Funktion -	28
Funktion *	29
Funktion cls	30

  Vergessen Sie nicht, das Design nochmals zu kompilieren.

- Erstellen Sie ein SCF-File, das den Rechner simulieren soll. Simulieren Sie diesen, und überprüfen Sie Ihr Ergebnis.

- Programmieren Sie den Baustein auf dem HWPRAK-Altera-Board, und überprüfen Sie ihren Rechenr, in dem Sie die Zahlen A und B verstellen, und verschiedene Operationen durchführen.

Geforderte Dokumentation:

- Eingegebener Schaltkreis (GDF-File)
- Der Device View des gerouteten SRAMs
- Das SCF-File der Simulation

9.5 Anhang zu Versuch Nr. 7

9.5.1 Erläuterung des HWPRAK-Altera-Board für Versuch Nr. 7

Im Abbildung 3 ist das HWPRAK-Altera-Board zu sehen, wie es für diesen Versuch verwendet wird. Im folgenden sind kurz die Ein- und Ausgabeeinheit erläutert.

9.5.1.1 Die Eingabeeinheit

- Die Eingabeeinheit besteht aus 2 BCD-Drehschaltern, mit denen man die gewünschte Zahl A oder B einstellt, und einem Tastenfeld, mit denen man die auszuführende

- Der BCD-Schalter liefert an 4 Ausgängen die eingestellte Zahl in binärer Darstellung.

- Die Ausgänge der Schalter und des Tastenfeldes sind auf Buchsenleisten geführt, von denen man mit kleinen Leitungen Verbindungen zu den Eingängen des EPM7128S-Bausteines herstellen kann..

- Jeweils der gemeinsame Anschluss der Schalter und der Taster ist auf +5V (1) gelegt. Die Ausgänge sind mit einem Pullupwiderstand von 1kΩ gegen Masse (0) geführt. Dadurch liegt z.B. am Ausgang der Taster bei gedrückter Taste eine 1 an. Beim BCD-Schalter bewirkt es, dass die Zahl X auch tatsächlich so am Ausgang anliegt, wie man dies gewohnt ist (also z.B. 0110 für die Zahl 6).

- Die Versorgungsspannung wird über ein Kabel von der `DIGILAB picoMAX – Platine` geholt.

9.5.1.2 Die Ausgabeeinheit

- Die Ausgabeeinheit besteht aus zwei 7-Segmentanzeigen, und einem programmierten Baustein `ispLSI2032`.

- Auf den 7-Segmentanzeigen wird das Ergebnis der Rechnoperationen `+, -, *` und `cls` dargestellt.

- Das Ergebnis wird mit Verbindungsleitungen von den Anschlüssen des Altera-Bausteins EPM7128S als Binärzahl auf eine 5-polige Buchsenleiste auf die Ausgabeeinheit verbunden. Die Binärzahl wird dann in dem programmierten Baustein ispLSI2032 so umgewandelt, dass die entsprechenden Segmente der beiden Anzeigen richtig aufleuchten.

- Die Segmente der beiden Anzeigen besitzen alle einen Vorwiderstand von 220 Ω.

- Die Versorgungsspannung wird über ein Kabel von der `DIGILAB picoMAX – Platine` geholt.

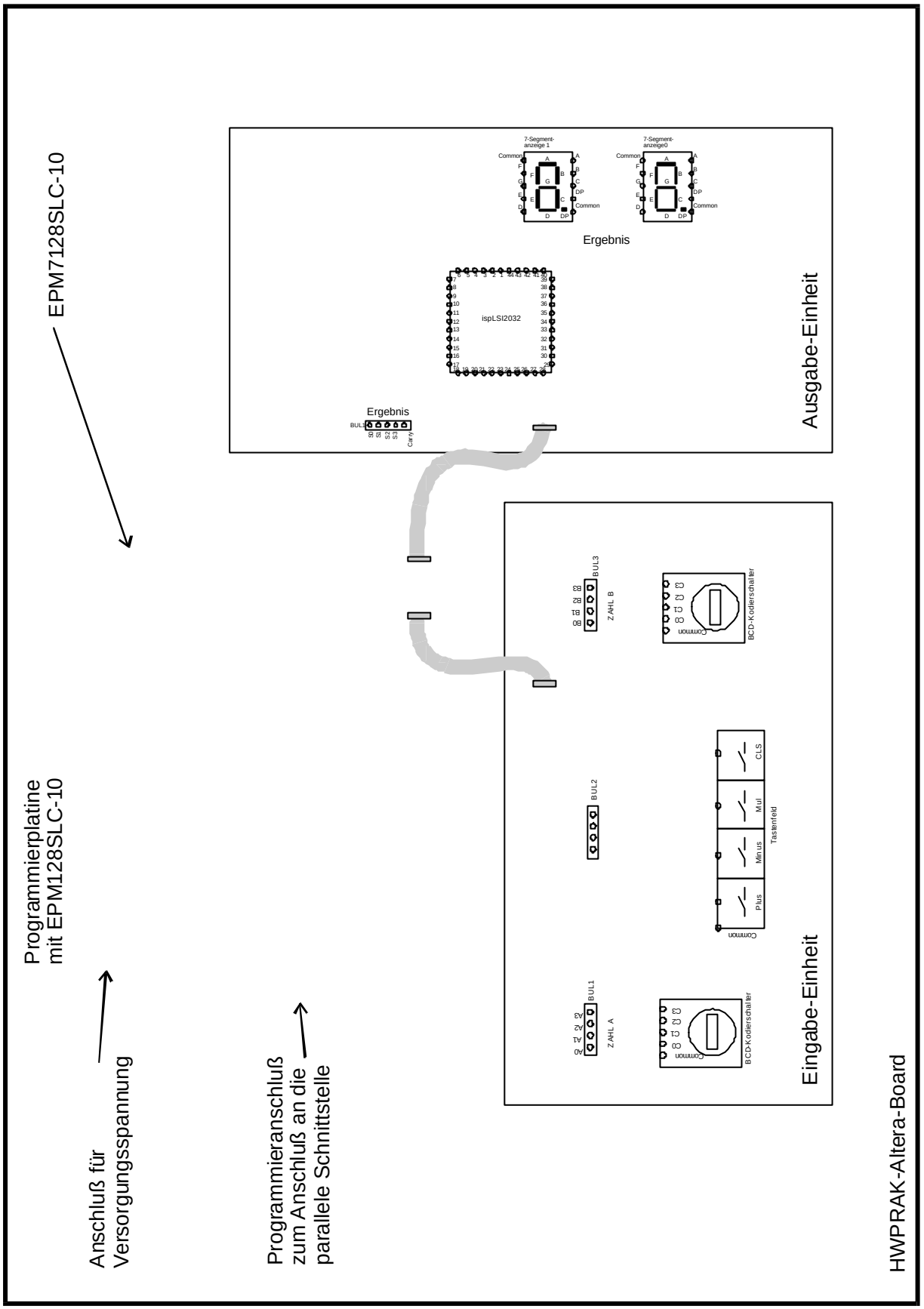

Abbildung 3: HWPRAK- Altera-Board für Versuch 7