

# Mikroprozessor-interne Datenbusse (MIB)

Labor Digital Design

## Inhalt

1 Ziel .....	1
2 Datenbusse der ALU .....	2
2.1 Schaltung .....	2
2.2 Anschluss der Register zu den Datenbussen .....	2
2.3 Anschluss zu den Ein- / Ausgangsbus .....	3
2.4 Daten aus dem Mikroprozessor-Befehl stammend .....	3
2.5 Erstellung .....	3
3 Software-erstellung eines seriellen Ports .....	4
3.1 Serielle Übermittlung .....	4
3.2 Algorithmus .....	4
3.3 Erstellung .....	4

## 1 | Ziel

Dieses Labor dient, den Entwurf von gemeinsam benutzten Datenbussen zu üben. Es zeigt die Anwendung von Tri-State Schaltungen.

Es stellt den Betrieb einer arithmetischen und logischen Einheit (Arithmetic and Logical Unit (ALU)) eines Mikroprozessors mit Datenregistern vor.



## 2 | Datenbusse der ALU

### 2.1 Schaltung

Die Abbildung [Abbildung 1](#) zeigt einen Teil eines  $\mu$ prozessors, zusammengesetzt aus einer ALU, 4 Registern, einem Interface zu einem Ein-/Ausgangs- (Input/Output (I/O)) Bus und einem Anschluss zum  $\mu$ prozessor Befehl (instruction).

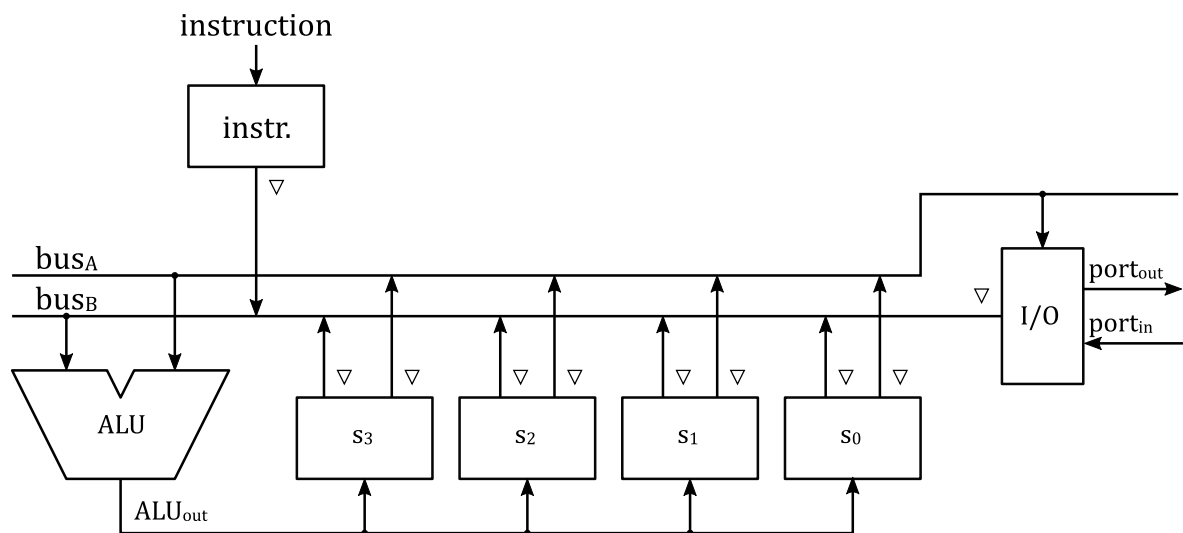


Abbildung 1: ALU und Register mit Datenbussen zusammengeknüpft.

### 2.2 Anschluss der Register zu den Datenbussen

Die Abbildung [Abbildung 2](#) zeigt den Bestandteil der Registern. Auf die steigende Flanke des Takt-signals, wenn das Kontrollsignal  $en = '1'$ , werden die Eingangsdaten ins Register geladen:  $Q^+ = D$ . Wenn  $en = '0'$ , wird der Inhalt des Registers gespeichert:  $Q^+ = Q$

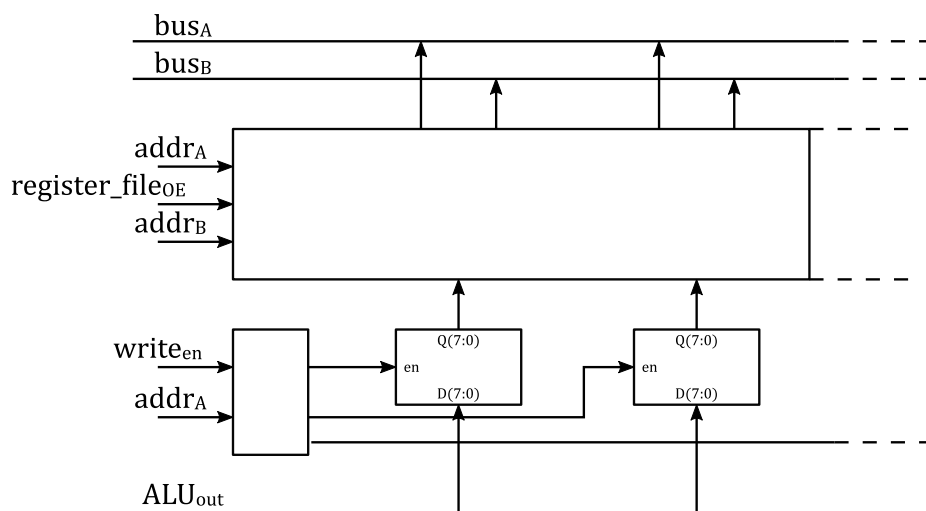


Abbildung 2: Datenregister



Vervollständigen Sie die Schaltung der Abbildung [Abbildung 2](#), um die Registerausgänge an den beiden Bussen verknüpfen zu können. Das erstellte System soll es erlauben, dass gleichzeitig ein Register seine Daten auf dem  $\text{bus}_A$  und ein anderer die seinen auf dem  $\text{bus}_B$  legt.

Die Zahlen  $\text{addr}_A$  und  $\text{addr}_B$  geben an, welches Register seine Information auf dem  $\text{bus}_A$  bzw. auf dem  $\text{bus}_B$  schreibt. Der Signal  $\text{register\_file\_OE}$  gibt an, ob Register-Daten auf dem  $\text{bus}_B$  zu bringen sind.

Auf ähnlicher Weise geben  $\text{write}_{\text{en}}$  und  $\text{addr}_A$  an, ob und in welchem Register geschrieben wird. Somit wird das Resultat einer Operation, dessen erster Operand das  $\text{addr}_A$ -Register ist, in dasselbe Register zurück geschrieben.

## 2.3 Anschluss zu den Ein- / Ausgangsbuss

Zeichnen Sie die interne Schaltung des I/O-Blocks der Abbildung [Abbildung 3](#).

Wenn ein Wort im  $\mu$ prozessor gelesen wird, muss das Steuersignal  $\text{port}_{\text{in\_OE}} = '1'$  aktiviert werden, und die Daten des  $\text{port}_{\text{in}}$ -Busses werden dann auf den  $\text{bus}_B$  geschrieben. Soll ein Wort nach aussen geschrieben werden, dann werden die Daten des  $\text{bus}_A$  auf den  $\text{port}_{\text{out}}$ -Bus geschrieben, und ein zur ALU externer Signal,  $\text{write}_{\text{strobe}}$ , wird aktiviert.

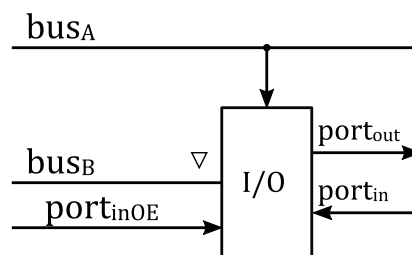


Abbildung 3: Ein- / Ausgangsblock

## 2.4 Daten aus dem Mikroprozessor-Befehl stammend

Die zweite Operand einer Operation kann im  $\mu$ prozessor-Befehl gegeben sein. Er wird dann mit Hilfe des Blocks von Abbildung [Abbildung 4](#) auf dem B-Bus gebracht.

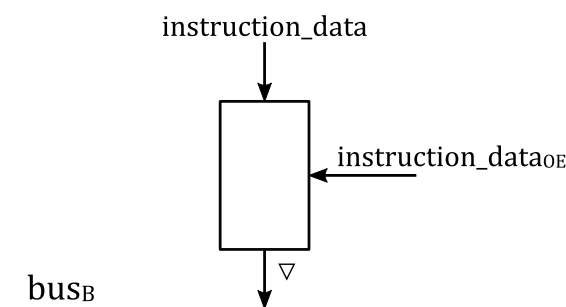


Abbildung 4: Daten aus dem  $\mu$ prozessor-Befehl stammend

Zeichnen Sie das interne Schema des Blocks in Abbildung [Abbildung 4](#).

## 2.5 Erstellung

Abhand von den vorigen Punkten, ergänzen Sie die Schaltung mit den internen  $\mu$ prozessorbussen, welche Ihnen für dieses Labor vorbereitet wurde



## 3 | Software-erstellung eines seriellen Ports

### 3.1 Serielle Übermittlung

Die Abbildung [Abbildung 5](#) gibt das zeitliche Verhalten der seriellen Übermittlung eines Datenwortes.

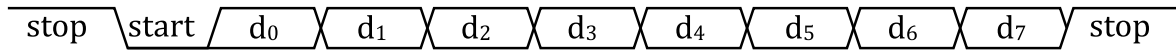


Abbildung 5: Serielle Übermittlung

In unserer Anwendung werden die seriellen Daten auf dem niederwertigsten Bit des  $\text{port}_{\text{out}}$ -Datenbusses übermittelt. In der Testbank ist dieser Bus einem externen Register angeschlossen und die Testbank steuert den Labebefehl  $\text{write}_{\text{strobe}}$  dieses Registers.

### 3.2 Algorithmus

Der zu programmierende Algorithmus ist der folgende:

```

LOAD      s3, FF          ; load stop bit
OUTPUT    s3              ; output stop bit
LOAD      s3, s3          ; no operation
LOAD      s3, s3          ; no operation
LOAD      s3, s3          ; no operation
LOAD      s3, s3          ; no operation
LOAD      s0, 00          ; load start bit
OUTPUT    s0              ; output start bit
INPUT     s1              ; load word to send
OUTPUT    s1              ; output word, LSB is considered
SR0       s1              ; shift word, bit 1 -> LSB
OUTPUT    s1              ; output bit 1
SR0       s1              ; bit 2 -> LSB
OUTPUT    s1              ; output bit 2
SR0       s1              ; bit 3 -> LSB
OUTPUT    s1              ; output bit 3
SR0       s1              ; bit 4 -> LSB
OUTPUT    s1              ; output bit 4
SR0       s1              ; bit 5 -> LSB
OUTPUT    s1              ; output bit 5
SR0       s1              ; bit 6 -> LSB
OUTPUT    s1              ; output bit 6
SR0       s1              ; bit 7 -> LSB
OUTPUT    s1              ; output bit 7
LOAD      s3, s3          ; no operation
OUTPUT    s3              ; output stop bit
  
```

### 3.3 Erstellung

Ergänzen Sie die Testbank der ALU mit den Registern, um die Befehlssequenz zur seriellen Übermittlung zu erstellen.



Es ist wichtig, nie einen Bus im hochohmigen Zustand zu lassen. Steuern Sie Ihr System so, dass es immer ein Signal auf die A- und B-Busse gibt, auch wenn keine Information daraus benötigt wird.