

## Aufgabe 1.1: Technologien und Grundlagen (12,5 Punkte)

a) Gegeben sei folgende Wahrheitstabelle:

$a$	$b$	$c$	$f(a, b, c)$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Geben Sie  $f(a, b, c)$  in konjunktiver kanonischer Normalform (KKN) an.

Realisieren Sie  $f(a, b, c)$  mithilfe eines 4:1 MUX-Bausteins, der Eingangsvariablen und beliebig vielen Invertern.



b) Vereinfachen Sie  $g(a, b, c)$  algebraisch soweit wie möglich.

$$g(a, b, c) = \overline{a}\overline{b}\overline{c} + \overline{a}\overline{b}c + \overline{a}b\overline{c} + \overline{a}bc + a\overline{b}\overline{c} + ab\overline{c}$$

- c) Formen Sie  $h(a, b, c, d)$  so um, dass sich die Schaltfunktion ausschließlich aus NOR-Ausdrücken über jeweils zwei Termen zusammensetzt. Das Resultat darf keine Negationen in Form eines Negations-Strichs enthalten. Verwenden Sie für die finale Darstellung die Notation:  $(a \downarrow b)$ .

$$h(a, b, c, d) = (a + b) * (c \downarrow d)$$

Zeichnen Sie den Schaltplan der Schaltungsfunktion  $h(a, b, c, d)$  unter Verwendung von Schaltsymbolen neuer DIN-Norm bestehend aus NOR-Gattern mit zwei Eingängen.

- d) Realisieren Sie  $l(a, b, c) = a \downarrow (b * c)$  unter Verwendung der CMOS-Technologie (Grundsaltung und Komplementärschaltung). Verwenden Sie folgende Symbole für n-Kanal-  und p-Kanal-CMOS-Transistoren .

# Aufgabe 1.2: Steuerwerksentwurf

(12,5 Punkte)

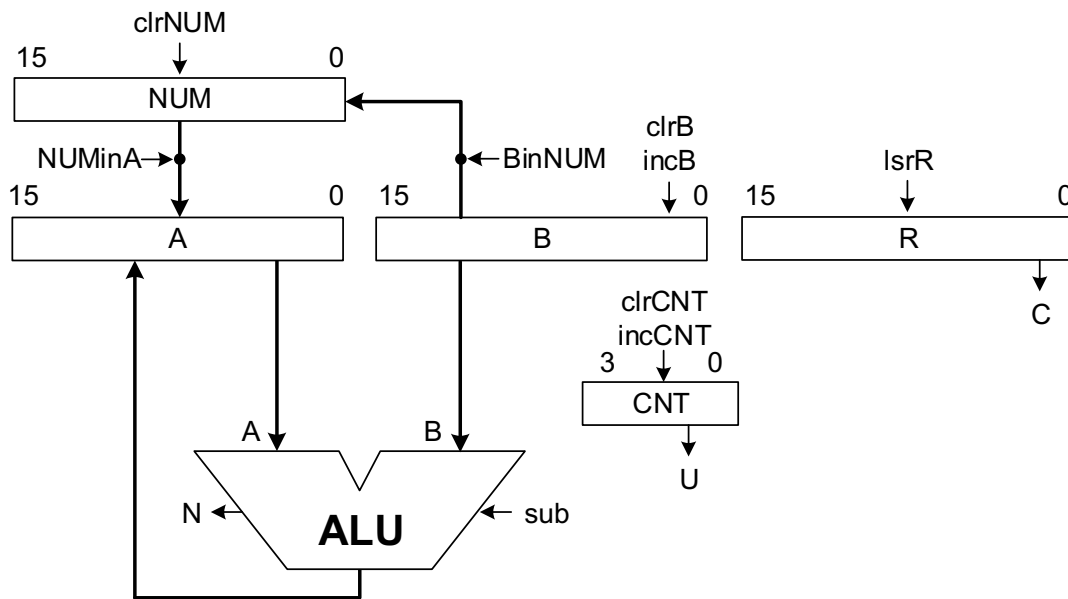


Abbildung 1: Operationswerk

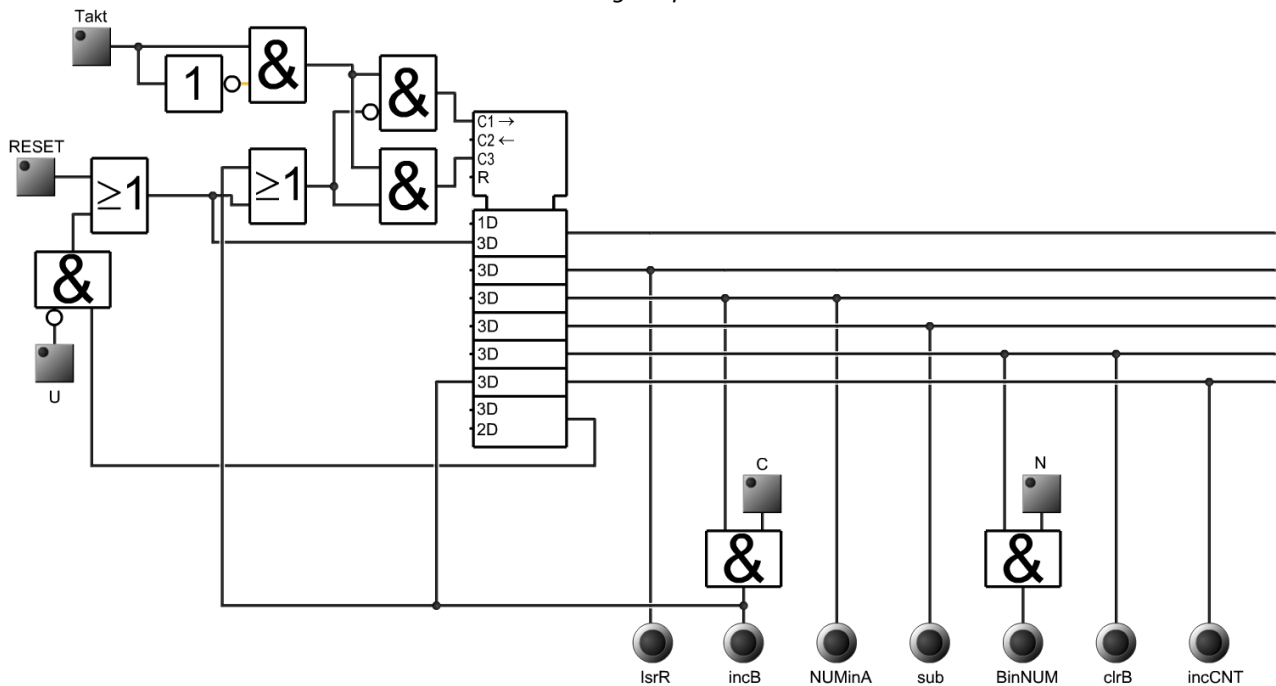


Abbildung 2: Steuerwerk

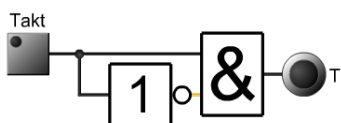


Abbildung 3: Teilschaltung 1

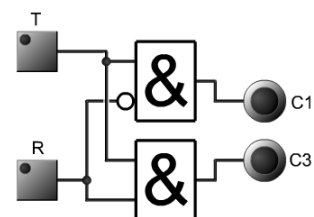


Abbildung 4: Teilschaltung 2

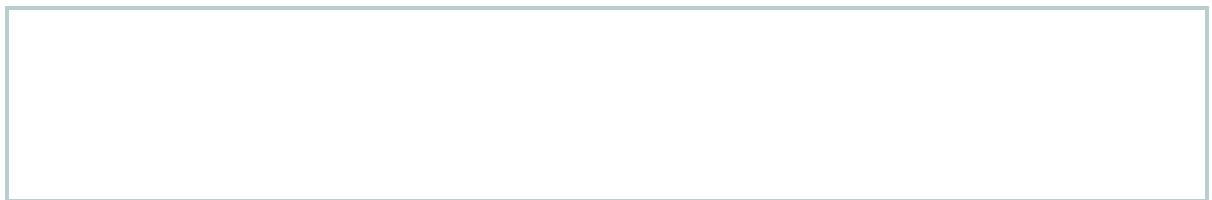
Matrikelnummer: \_\_\_\_\_

Studiengang: \_\_\_\_\_

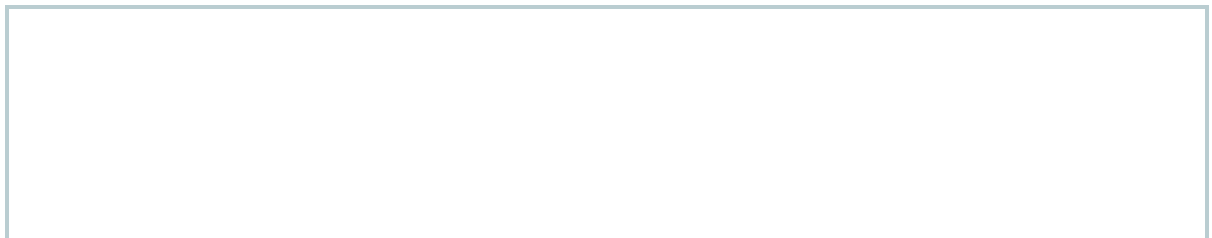
Das gegebene Operationswerk aus Abbildung 1 basiert auf den vier 16-Bit Registern  $A$ ,  $B$ ,  $R$  und  $NUM$ , dem 4-Bit Register  $CNT$  und einer 16-Bit ALU. Sowohl die Eingänge als auch der Ausgang der ALU und Register sind 16 Bit breit. Mithilfe des Steuersignals  $sub$  kann die Subtraktion  $A = A - B$  auf der ALU ausgeführt werden. Ist das Ergebnis negativ, so setzt die ALU das  $N$ -Flag. Als weitere Kriterien stehen das  $U$ -Flag, welches bei einem Überlauf des Registers  $CNT$  gesetzt wird, und das  $C$ -Flag, welches das aus dem Register  $R$  im Rahmen eines durch  $IsrR$  hervorgerufenen Rechtsshifts herausgeschiftete Bit enthält, zur Verfügung. Die Steuersignale  $clrX$  ( $X \in \{CNT, B, NUM\}$ ) setzen den Wert des entsprechenden Registers auf 0 und die Steuersignale  $incB$  und  $incCNT$  inkrementieren die korrespondierenden Registerwerte. Das Steuersignal  $NUMinA$  erlaubt das Kopieren des Wertes von Register  $NUM$  in Register  $A$  wohingegen  $BinNUM$  den Wert aus dem Register  $B$  in das Register  $NUM$  kopiert.

Abbildung 2 zeigt ein für das Operationswerk konzipiertes Steuerwerk, deren Analyse Kern dieser Aufgabe ist.

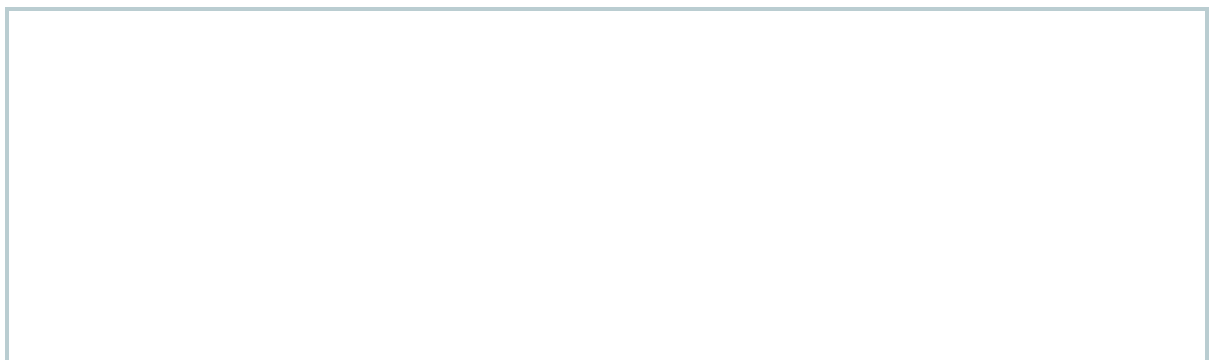
- a) Um welche Art von Steuerwerk handelt es sich?



- b) Welche Funktionalität realisiert die Teilschaltung aus Abbildung 3?



- c) Geben Sie die durch die in Abbildung 4 gegebene Teilschaltung realisierten Schaltfunktionen für  $C1$  und  $C3$  an.



- d) Analysieren Sie nun das in Abbildung 2 gegebene Steuerwerk für das Operationswerk aus Abbildung 1, indem Sie die nachfolgend die Steuersignale ergänzen.

RT-Quellcode	Steuersignale
declare register CNT(3:0),A(15:0),B(15:0), R(15:0),NUM(15:0),C,N,U	
R <- 252, CNT <- 0, A <- 0, U <- 0, B <- 0;	
LOOP:	
_____;	_____
if _____ then	
_____	_____
fi,	
_____;	_____
_____;	_____
if _____ then	
_____	_____
fi,	
_____;	_____
CHECK:	
_____;	_____
if _____ then	
_____	
fi;	
END:	
goto END;	

- e) Geben Sie nun die zu den Steuersignalen korrespondierenden Registertransferoperationen an, indem Sie den RT-Code ergänzen. Das Setzen der einzelnen Flags müssen Sie nicht realisieren.

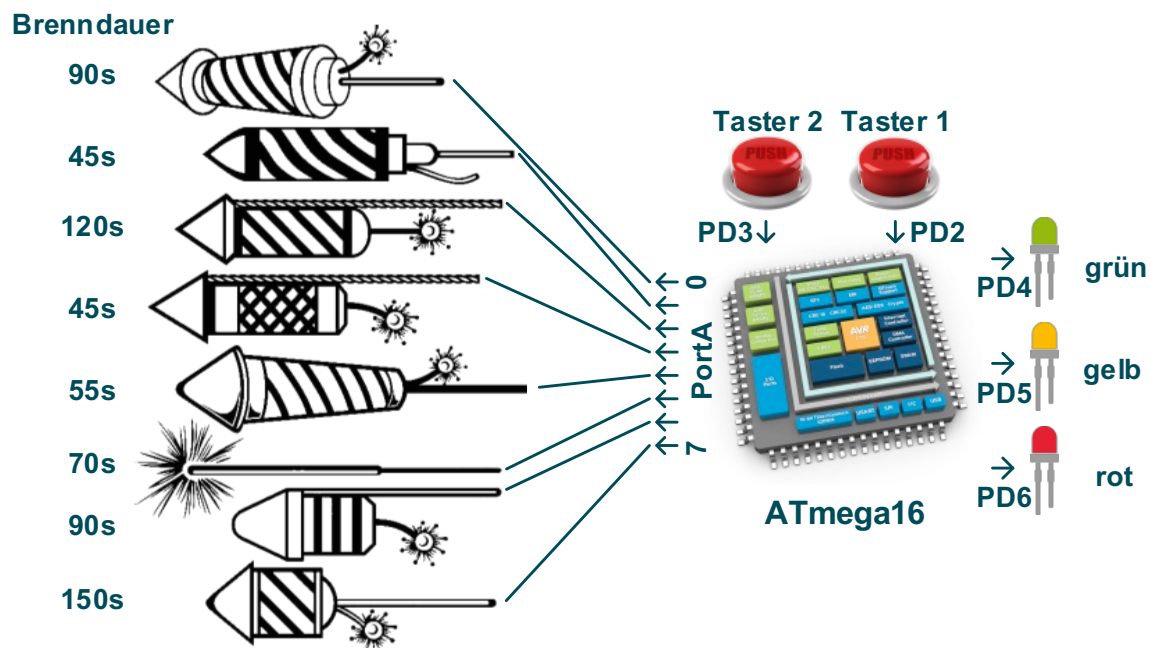
- f) Welche der nachfolgend genannten Berechnungen wird vom Steuerwerk auf dem Operationswerk durchgeführt?
- Berechnung der Hamming-Distanz von A und B
  - Zählen der im Wert von R vorkommenden 1en.
  - Zählen der im Wert von R vorkommenden 0en.
  - Berechnung der Länge (Anzahl an Bits) des längsten zusammenhängenden Blocks an 1en in R.
  - Berechnung der Länge (Anzahl an Bits) des längsten zusammenhängenden Blocks an 0en in R.
  - Berechnung der Parität
- g) Wie oft wird die im Algorithmus vorkommende Schleife (Loop) je Durchlauf des Algorithmus abgearbeitet?

$\Sigma_{A1} =$  \_\_\_\_\_ Punkte

## Aufgabe 2: Assemblerprogrammierung

### Steuerung eines Feuerwerks

(25 Punkte)



Ziel dieser Aufgabe ist die Implementierung einer vereinfachten Feuerwerksteuerung in Assembler für den aus der Übung bekannten Mikrocontroller ATmega16.

Das Feuerwerk besteht aus insgesamt 8 nacheinander zu zündenden Feuerwerksbatterien, deren Zündvorrichtungen jeweils mit einem Pin von PortA verbunden sind. Sobald an einem Pin ein High-Pegel angelegt wird, zündet die entsprechende Batterie. Solange das Feuerwerk noch nicht gezündet wurde oder wenn es abgebrannt ist, leuchtet die grüne LED, um zu verdeutlichen, dass sich dem Feuerwerk gefahrlos genähert werden kann. Aus Sicherheitsgründen müssen zum Start die beiden Taster an PinD2 und PinD3 in beliebiger Reihenfolge betätigt werden. Ist ein Taster betätigt worden, so erlischt die grüne LED und es leuchtet die gelbe LED. Wurde das Feuerwerk gestartet leuchtet für die gesamte Brenndauer nur die rote LED. Nach dem Start zündet der ATmega sequentiell die einzelnen Batterien beginnend bei der an PinA0 angeschlossenen bis hin zu der mit PinA7 verbundenen. Jede Batterie darf erst gezündet werden, wenn die vorherige vollständig abgebrannt ist.

### Aufgaben:

Die Bearbeitung der Aufgabe erfolgt in mehreren Unteraufgaben. Nutzen Sie die einzeln vorgegebenen Bereiche und schreiben Sie aussagekräftigen Assembler Code beziehungsweise kommentieren Sie gegebenenfalls den Code sinnvoll.



- a) Beginnen Sie die Programmierung mit den einzelnen Registernamensdefinitionen. Sorgen Sie dafür, dass das Register **R17** unter dem Namen **Start**, das Register **R18** unter dem Namen **Time** und das Register **R19** unter dem Namen **Duration** angesprochen werden können. Weitere von Ihnen eventuell genutzte Namen für einzelne Register müssen hier ebenfalls definiert werden.

- b) Initialisieren Sie die Interrupt-Vektor-Tabelle, sodass nach einem **RESET** zum Label **INIT**, nach dem **External Interrupt Request 0** zur **ISR\_INT0**, beim **External Interrupt Request 1** zur **ISR\_INT1** und nach einem **Timer/Counter 1 Compare Match Interrupt** zur **ISR\_TIMER1** gesprungen wird.

- c) Jede Batterie hat eine definierte Brenndauer, die der obigen Abbildung entnommen werden kann. Legen Sie die 8 Brenndauern direkt hinter dem letztmöglichen Eintrag der Interrupt-Vektor-Tabelle im ROM unter dem Namen **DURATIONS** ab.

- d) Konfigurieren Sie unter dem Label **INIT** die Ein- und Ausgabeports und initialisieren Sie den Stackpointer. Beachten Sie eventuelle Initialwerte für die Ausgänge. Initialisieren Sie das Register **Start** mit dem Wert 0.

- e) Konfigurieren Sie die externen Interrupts in der Art, dass **INT0** bei einer durch Taster 1 hervorgerufenen **fallenden Flanke** an PD2 und **INT1** bei einer durch den Taster 2 hervorgerufenen **steigenden Flanke** an PD3 ausgelöst wird.

- f) Die Messung der verstrichenen Zeit nach dem Zünden einer Batterie erfolgt mit Hilfe des **Timer/Counter 1**. Konfigurieren Sie ihn so, dass er **einmal pro Sekunde** einen Compare Match Interrupt auslöst. Verwenden Sie den **CTC-Modus** und nutzen Sie einen **Prescaler** von **256**. Berechnen Sie den entsprechenden Vergleichswert. Aktivieren Sie zudem die Interrupts global.

- g) Implementieren Sie die Interrupt-Service-Routine **ISR\_INT0** die den Wert im Register **Start** beim Aufruf inkrementiert, den Interrupt **INT0** deaktiviert und dafür sorgt, dass die grüne LED aus- und die gelbe LED eingeschaltet wird.

- h) Die **ISR\_INT1** weist das gleiche Verhalten auf, wie die **ISR\_INT0**, deaktiviert jedoch den Interrupt **INT1**. Welche Änderungen an der **ISR\_INT0** sind notwendig, um aus ihr die **ISR\_INT1** zu machen?

- i) Warum ist es in diesem Fall notwendig, INT0 beziehungsweise INT1 zu deaktivieren.

- j) Implementieren Sie die **ISR\_TIMER1** Interrupt Service Routine, welche den aktuellen Wert des Registers **Time** bei jedem Aufruf inkrementiert.

- k) Implementieren Sie nun die in einer Endlosschleife laufenden Grundfunktionalität unter dem Label **MAIN**. Warten Sie in einer Schleife darauf, dass beide Taster betätigt wurden (**Start** = 2). Schalten Sie die gelbe LED aus und die rote LED ein. Starten Sie nun in einer Schleife nacheinander die einzelnen Batterien beginnend bei der an PinA0. Laden Sie die jeweilige Brenndauer aus dem **ROM** in das Register **Duration**, setzen Sie nach dem Zünden der jeweiligen Batterie das Register **Time** zurück auf den Wert 0 und warten Sie entsprechend, bis die Batterie abgebrannt ist, bevor Sie die nächste Zünden. Wurde das komplette Feuerwerk abgebrannt, wird die rote LED wieder aus- und die grüne LED angeschaltet. Anschließend soll in einer Endlosschleife verharrt werden.

$\Sigma_{A2} =$  \_\_\_\_\_ Punkte