Musterlösungen

zu den Übungsaufgaben vom 22. 1. 07

1. Es ist zyklisch (mit Takt CLK) ein Impulsmuster gemäß Abb. 1 zu erzeugen. Entwerfen Sie eine entsprechende Schaltung zur Implementierung in einem programmierbaren Logikschaltkreis.

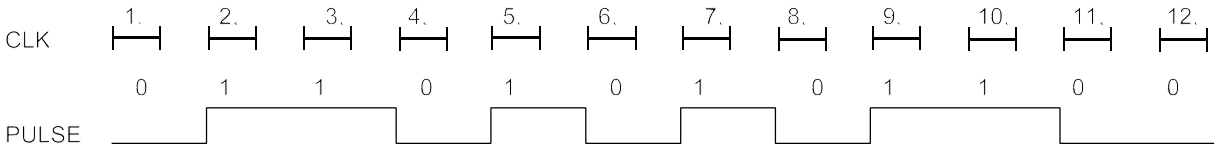
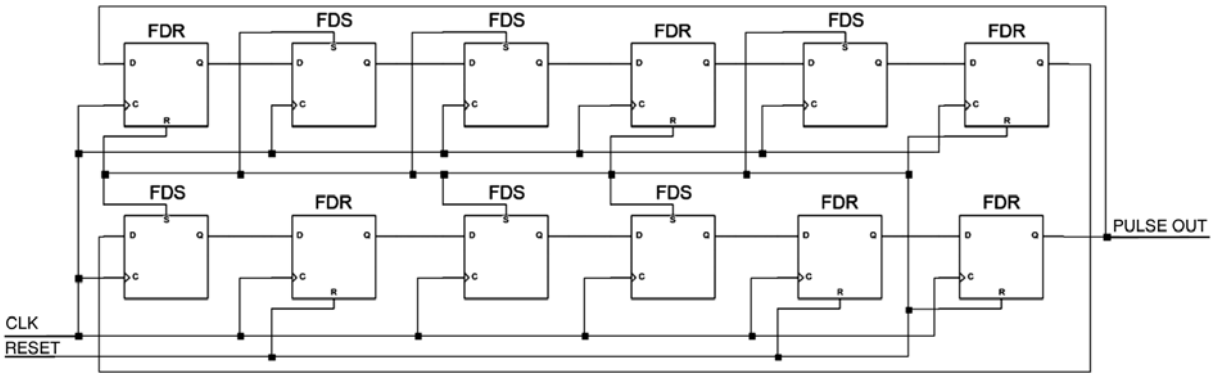


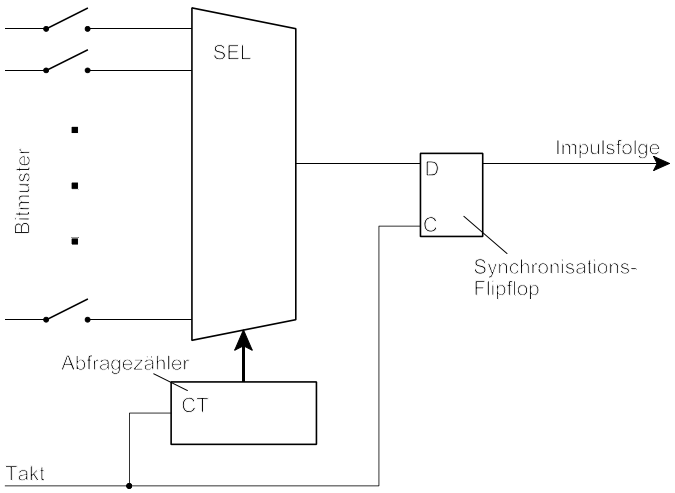
Abb. 1

In Abb. 1 haben wir schon eingezeichnet, was eigentlich zu liefern ist. Die Impulsfolge ist 12 Takte lang. Es gibt zwei einfache Grundvarianten:

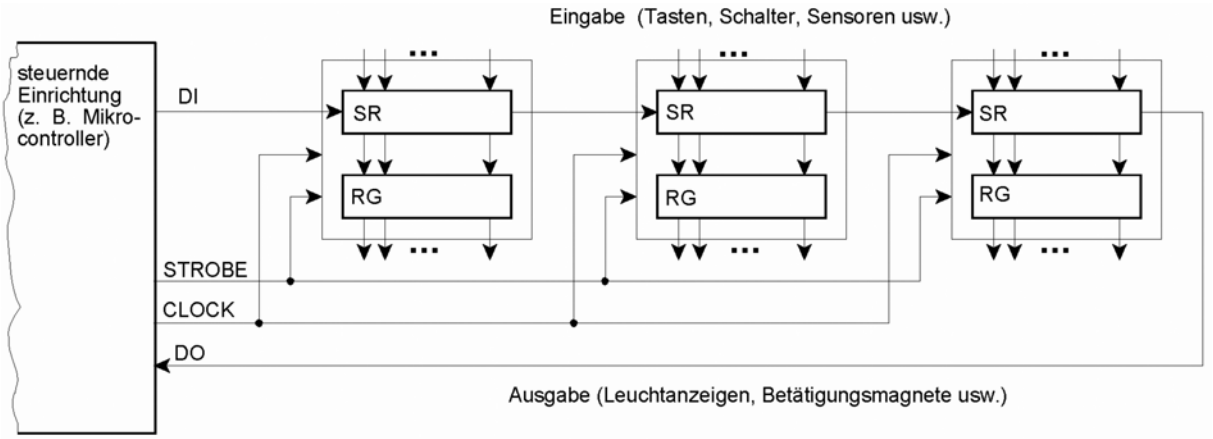
1. Schiebeprinzip. Ein als Ring geschaltetes Schieberegister mit 12 Flipflops bauen. Das Bitmuster beim Einschaltrücksetzen passend einstellen:



2. Abfrageprinzip, z. B. mit Multiplexer und Zähler. Der Zähler muß hier modulo 12 zählen. Die genaue Ausgestaltung sei offen gelassen (als weitere Übungsgelegenheit) :

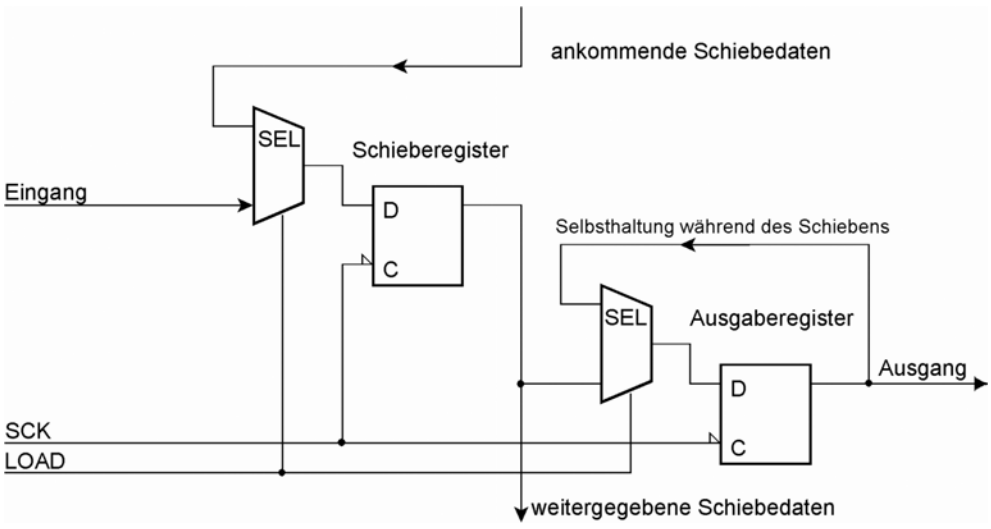


2. Erläutern Sie kurz (mit Skizze), wie ein einfaches Schieberegister-Interface (für parallele Ein- und Ausgabe) aufgebaut ist. Welchen Vorteil hat dieses Prinzip? Nennen Sie wenigstens zwei Schieberegister-Interfaces, die als Industriestandard anzusehen sind. Skizzieren Sie eine Schaltungslösung (eine Bitposition genügt), die sich zur Implementierung mit CPLDs eignet. Wieviele CPLD-Zellen brauchen Sie für eine Bitposition?



Vorteile: Man braucht nur wenige Signalleitungen (z. B. Dateneingang, Datenausgang, Takt, Strobe), um beliebig viele Bits ein- und auszugeben. Der Steuerungsaufwand ist sehr gering; man kann ggf. die Signalspiele direkt mittels Software steuern.

Beispiele: I^2C-Bus Microwire, SPI.



Eine Zelle enthält ein Flipflop. Die vorgeschaltete Kombinatorik (eine 2:1-Auswahl) läßt sich mit dem UND-ODER-Knoten der Zelle ihne weiteres bauen. Also genügen ***zwei Zellen***. Wichtig: alles mit einem einzigen Takt erledigen (vollsynchone Arbeitsweise). Als weitere Übungsgelegenheit: weshalb?

3. Erläutern Sie kurz, worin sich GALs und CPLDs voneinander unterscheiden.

Beide Schaltkreisarten haben vergleichsweise wenige große Zellen (Makrozellen) die UND-ODER-Strukturen mit nachgeschalteten Flipflops darstellen.

GAL: vergleichsweise wenige Zellen (z. B. 10), die einzeln direkt auf die Schaltkreisanschlüsse geführt sind. Komplexe Funktionen sind durch Verbindungen auf der Leiterplatte zu realisieren.

CPLD: mehr Zellen (von typischerweise 32...36 an aufwärts bis zu einigen hundert). Trennung zwischen Zellen und E-A-Koppelstufen. Interne Verbindungsnetzwerke. Komplexe Funtionen können innerhalb des Schaltkreises realisiert werden.

4. Nennen Sie wenigstens zwei Prinzipien der Parameterübergabe an Unterprogramme. Veranschaulichen Sie diese Prinzipien anhand kurzer Programmstücke (AVR-Assemblerprogrammierung). Das Unterprogramm soll SPECIAL heißen. Es sollen drei Parameter P1, P2, P3 (jeweils ein Byte) übergeben werden. P1 ist ein Direktwert, P2 steht in Register r12 und P3 steht im SRAM unter der symbolischen Adresse EX_P.

Stack	Register	RAM

ldi temp, p1
push temp
push r12
lds temp, EX_P
push temp
	ldi pr1, p1
	mov pr2, r12
	lds pr3, EX_P
		ldi temp, p1
		sts P_1, temp
		sts P_2, r12
		lds temp, EX_P
		sts P_3, temp

Wir brauchen stets ein Hilfsregister (temp). Die Übergaberegister heißen hier pr1, pr2 usw., die Übergabezellen im RAM P_1, P_2 usw.

5. Es geht um einen Testprogrammablauf. Über Port A eines AVR-Mikrocontrollers ist ein Datenbyte auszugeben und zur Kontrolle wieder zurückzulesen. Das auszugebende Datenbyte steht in r16. Der Kontrollwert soll nach r20 gebracht werden. Geben Sie eine geeignete Befehlsfolge an. Worauf ist hierbei besonders zu achten?

```
OUT     porta,r16
NOP
IN      r20,pina
```

Vom Pin zurücklesen, nicht vom Port. Nicht sofort nach dem OUT wieder zurücklesen. Zeit zur Synchronsation beim Rücklesen und ggf. zum Schalten des Signals (Stichwort: kapazitive Belastung) einräumen.

6. Parallele Bussysteme kann man auf vielfältige Weise aufbauen. In der Praxis sind zwei Auslegungen weit verbreitet, die man als Intel-Ausführung und als Motorola-Ausführung bezeichnen könnte. Erläutern Sie kurz. worin die kennzeichnenden Merkmale beider Ausführungen bestehen. Nennen Sie jeweils einen typischen Anwendungsfall.

Intel-Ausführung	Motorola-Ausführung
zwei Strobes; beide aktiv Low: RD# zum Lesen, WR# zum Schreiben. Beispiel: Speicherschaltkreise (ROM, SRAM)	ein Steuersignal, ein Strobe-Signal: R/W: Low = Schreiben, High = Lesen, E: Enable (Zugriff ausführen). Beispiel: LCD-Anzeigen

7. Die Register r2 bis r5 enthalten zwei 16 Bits lange Binärzahlen X und Y. Schreiben Sie einen Programmablauf (AVR Assembler) für die Subtraktion X-Y. Hierbei soll das Ergebnis in den Registern r2 und r3 zu stehen kommen.

r2	LO_X
r3	HI_X
r4	LO_Y
r5	HI_Y

```
SUB     r2,r4
SBC     r3,r5
```

8. Wir beziehen uns auf Aufgabe 7. Jetzt ist aber die Subtraktion X-Y so auszuführen, daß das Ergebnis in den Registern r18 und r19 zu stehen kommt. Die Inhalte der Register r2 bis r5 sollen unverändert erhalten bleiben.

```
MOV     r18,r2
MOV     r19,r3
SUB     r18,r4
SBC     r18,r5
```

9. Schreiben Sie einen Programmablauf, der über Port C, Bitpositionen 1 und 0 das in Abb. 2 gezeigte Bitmuster ausgibt.

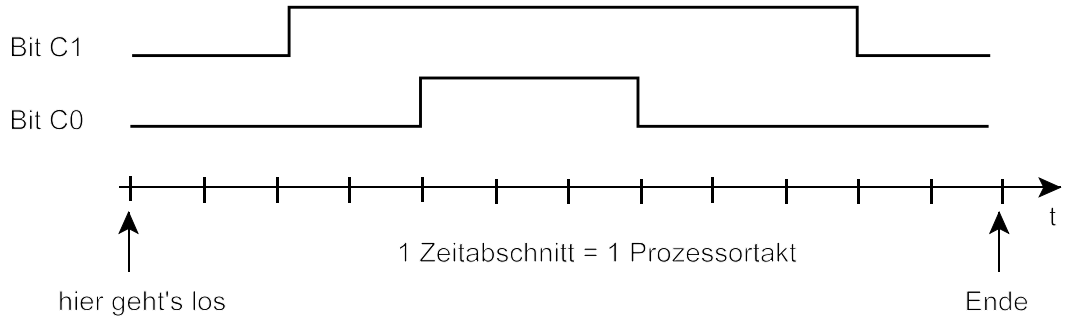


Abb. 2

```
LDI     temp,0          ; anfängliche Einstellung
OUT     portc,temp      ; 2 Takte
SBI     portc,1         ; 2 Takte
SBI     portc,0
NOP                     ; insgesamt 3 Takte
CBI     portc,0
NOP                     ; insgesamt 3 Takte
CBI     portc,1
```

10. Schreiben Sie eine Interruptroutine zur Bedienung des Analog-Digital-Wandlers. Die Datenregister sind einzulesen. Aus den 10 Bits ist ein 8-Bit-Wert zu bilden (Abb. 3) und über Port C auszugeben. Achten Sie darauf, daß das unterbrochene Programm nicht beeinträchtigt wird.

ADC Data Register ADCL und ADCH:

Register	7	6	5	4	3	2	1	0
ADCH	-	-	-	-	-	.	ADC9	ADC8
ADCL	ADC7							ADC0

Das gewünschte Ergebnis:

7	6	5	4	3	2	1	0
ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2

Abb. 3

Wir brauchen zwei Arbeitsregister, hier mit temp und temp1 bezeichnet. Da es sein kann, daß sie vom unterbrochenen Program verwendet werden, sind beide zu retten.

Beispiel mit verkürztem Schiebeablauf	**ein schulmäßiger Schiebeablauf**
push temp push temp1 in temp, sta push temp in temp1, adcl in temp, adch	ror temp ror temp ror temp andi temp, 0xc0 lsr temp1 lsr temp1 or temp1, temp
ror temp ror temp1 ror temp ror temp1	
out portc, temp1 pop temp out sta, temp pop temp1 pop temp reti	

11. Schreiben Sie ein Assemblerprogrammstück, das folgenden Ablauf implementiert (Abb. 4).

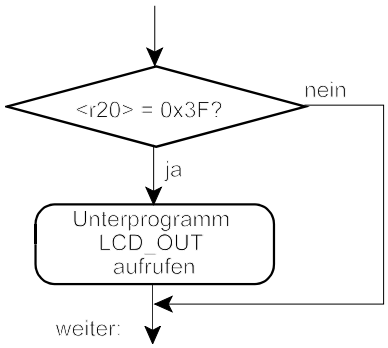


Abb. 4

```
        CPI     r20, 0x3f
        BRNE    weiter
        call    LCD_OUT
weiter:
```