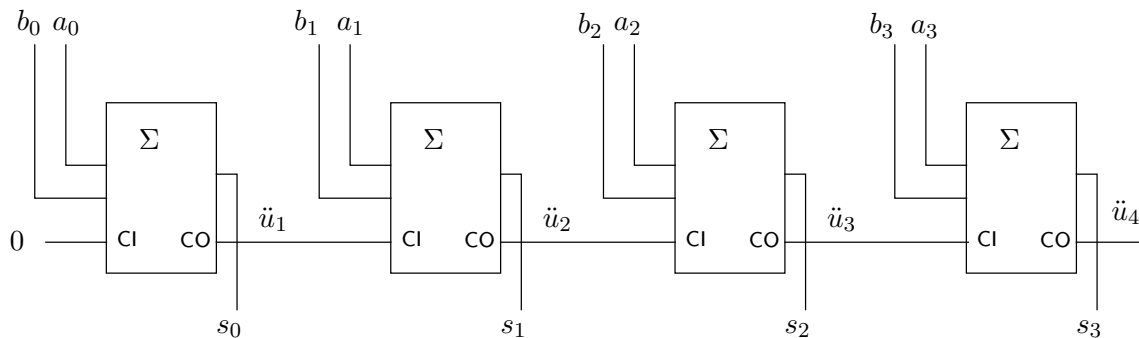




Lösung 1

1. 4-Bit-Volladdierer als *carry-ripple*-Addierer aus 1-Bit-Volladdierern:



2. Additionszeit:

- Die benötigte Zeit bis s_3 berechnet wird:
 $(3 + 2 + 2 + 1) ns = 8 ns$
- Die benötigte Zeit bis $ü_4$ berechnet wird:
 $(3 + 2 + 2 + 2) ns = 9 ns$

3. Gleichungen für einen *carry-look-ahead*-Addierer:

$$\ddot{u}_{i+1} = a_i b_i \vee (a_i \oplus b_i) \ddot{u}_i = g_i \vee p_i \ddot{u}_i$$

$$s_i = (a_i \oplus b_i) \oplus \ddot{u}_i = p_i \oplus \ddot{u}_i$$

$$\ddot{u}_1 = g_0 \vee p_0 \ddot{u}_0$$

$$\ddot{u}_2 = g_1 \vee p_1 g_0 \vee p_1 p_0 \ddot{u}_0$$

$$\ddot{u}_3 = g_2 \vee p_2 g_1 \vee p_2 p_1 g_0 \vee p_2 p_1 p_0 \ddot{u}_0$$

$$\ddot{u}_4 = g_3 \vee p_3 g_2 \vee p_3 p_2 g_1 \vee p_3 p_2 p_1 g_0 \vee p_3 p_2 p_1 p_0 \ddot{u}_0$$

Lösung 2

1. Die Schaltung besteht aus

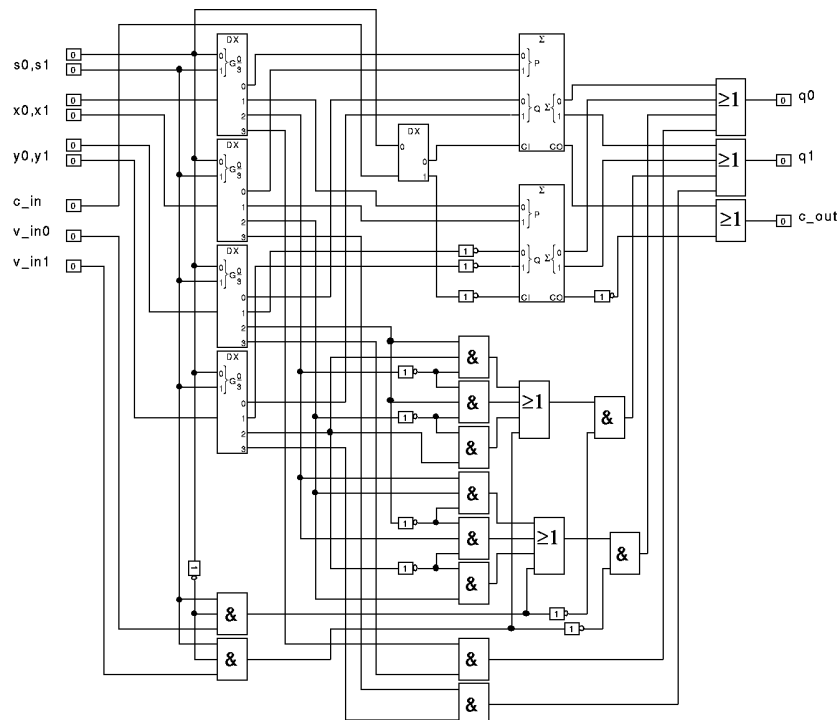
- einem 2-Bit Addierer (LOKON-Bibliothekselement),
- einem 2-Bit Subtrahierer (2-Bit Addition des Zweierkomplements)
- je einem Schaltnetz für den 2-Bit Vergleich $X > Y$ und $X < Y$ unter Berücksichtigung der Übertragseingänge v_{in1} und v_{in0}

v_{in1}	v_{in0}	x_1	x_0	y_1	y_0	q_0	v_{in0}	v_{in1}	y_1	y_0	x_1	x_0	q_1
0	0	0	0	-	-	0	0	0	0	-	-	0	
0	0	0	1	0	0	1	0	0	1	0	0	1	
0	0	0	1	0	1	0	0	0	1	0	1	0	
0	0	0	1	1	-	0	0	0	1	1	-	0	
0	0	1	0	0	-	1	0	0	1	0	0	-	1
0	0	1	0	1	-	0	0	0	1	0	1	-	0
0	0	1	1	0	-	1	0	0	1	1	0	-	1
0	0	1	1	1	0	1	0	0	1	1	1	0	1
0	0	1	1	1	1	0	0	0	1	1	1	1	0
0	1	-	-	-	-	1	0	1	-	-	-	-	1
1	-	-	-	-	-	0	1	-	-	-	-	-	0

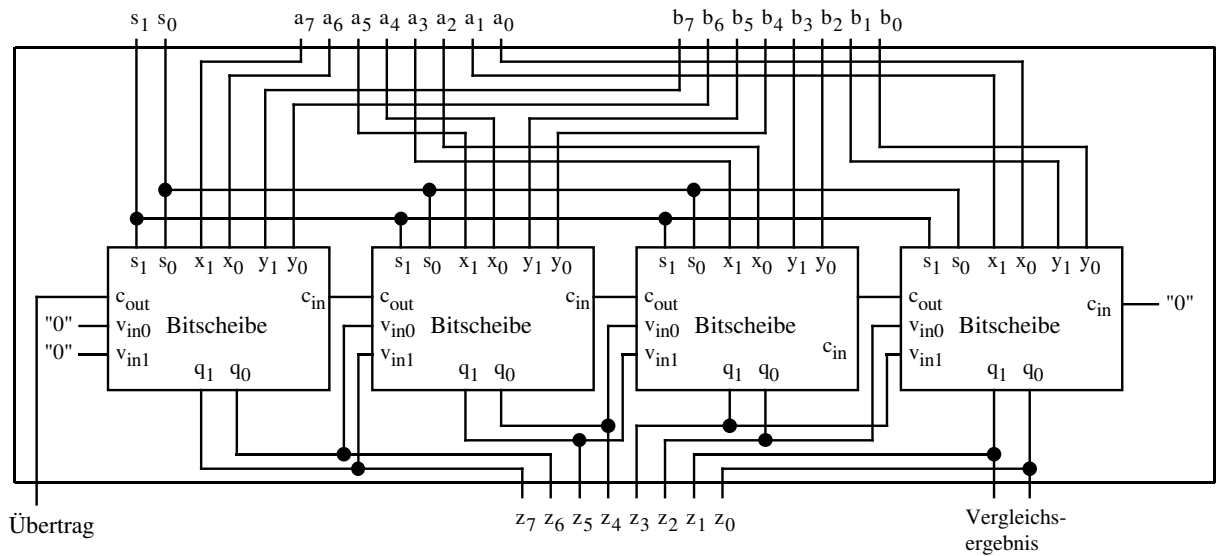
$$q_0 = (x_0 \bar{y}_1 \bar{y}_0 \vee x_1 x_0 \bar{y}_0 \vee x_1 \bar{y}_1 \vee v_{in0}) \bar{v}_{in1}$$

$$q_1 = (\bar{x}_0 y_1 y_0 \vee \bar{x}_1 \bar{x}_0 y_0 \vee \bar{x}_1 y_1 \vee v_{in1}) \bar{v}_{in0}$$

Die Datensignale werden über vier 1:4-Demultiplexer an die Teilschaltungen herangeführt. Der Übertrag c_{in} wird in Abhängigkeit von s_0 auf den Addierer oder den Subtrahierer geschaltet. Die Übertragseingänge v_{in0} und v_{in1} werden nur aktiv, wenn die Vergleichsoperation mit $s_1 = 1$ und $s_0 = 0$ ausgewählt ist. Ein Multiplexer am Ausgang ist nicht notwendig, da die nichtselektierten Demultiplexerausgänge gleich Null sind und damit das Ergebnis der Addition bzw. Subtraktion bei Durchführung einer Subtraktion bzw. Addition (einschließlich des Übertrags) ebenfalls Null ist. Dazu muss jedoch bei der Subtraktion ein negierter Übertrag verwendet werden und der Übertragsausgang auch negiert werden.



2. Die unterschiedliche Art und Weise, in der die Übertrage c_{in} bzw. v_{in0} und v_{in1} an eine weitere Stufe weitergegeben werden, ist hierbei zu beachten.



Lösung 3

1. $X \rightarrow Q, Y \rightarrow M, A := 0$. Vorzeichen: $p_0 = x_0 \leftrightarrow y_0$. Im folgenden bedeutet:

- Addiere M zu A: $A(0:7) := A(1:7) + M(1:7)$ und
- Rechtsschieben: $A(1:7).Q := A.Q(0:6)$

(a) $01101001 \times 11000110 = 1011100101101100$

M			
0110 1001			
A	Q		
0000 0000	1100 0110		
0000 0000	1100 0110	Addiere Null zu A	
0000 0000	0110 0011	Rechtsschieben	
0110 1001			
0110 1001	0110 0011	Addiere M zu A	
0011 0100	1011 0001	Rechtsschieben	
0110 1001			
1001 1101	1011 0001	Addiere M zu A	
0100 1110	1101 1000	Rechtsschieben	
0000 1001	1101 1011	3x Rechtsschieben	
0110 1001			
0111 0010	1101 1011	Addiere M zu A	
0011 1001	0110 1101	Rechtsschieben	
1011 1001	0110 1100	$A(0) := p_0 = 1, Q(7) := 0$	

(b) $10010111 \times 11100010 = 0001000110011100$

M		
1001 0111		
A	Q	
0000 0000	1110 0010	Addiere Null zu A
0000 0000	0111 0001	Rechtsschieben
001 0111		
0001 0111	0111 0001	Addiere M zu A
0000 1011	1011 1000	Rechtsschieben
0000 0001	0111 0111	3x Rechtsschieben
001 0111		
0001 1000	0111 0111	Addiere M zu A
0000 1100	0011 1011	Rechtsschieben
001 0111		
0010 0011	0011 1011	Addiere M zu A
0001 0001	1001 1101	Rechtsschieben
0001 0001	1001 1100	$A(0) := p_0 = 0, Q(7) := 0$

2.

(a) $01000110 \times 01101001 = 0011100101101100$

M		
0100 0110		
A	Q	
0000 0000	0110 1001	
0100 0110	0110 1001	add
0010 0011	0011 0100	shift
0000 1000	1100 1101	2x shift
0100 0110		
0100 1110	1100 1101	add
0010 0111	0110 0110	shift
0001 0011	1011 0011	shift
0100 0110		
0101 1001	1011 0011	add
0010 1100	1101 1001	shift
0100 0110		
0111 0010	1101 1001	add
0011 1001	0110 1100	shift, $A(0) := p_0 = 0, Q(7) := 0$

(b) $11100010 \times 00010111 = 111110101001110$

M		
1110 0010		
A	Q	
0000 0000	0001 0111	
1110 0010	0001 0111	add, (M(0)=1 und Q(7)=1) → F:=true
1111 0001	0000 1011	shift
1110 0010		
1101 0011	0000 1011	add
1110 1001	1000 0101	shift
1110 0010		
1100 1011	1000 0101	add
1110 0101	1100 0010	shift
1111 0010	1110 0001	shift
1110 0010		
1101 0100	1110 0001	add
1110 1010	0111 0000	shift
1111 1010	1001 1100	2x shift, A(0) := p ₀ = 1 , Q(7) := 0

(c) $01100010 \times 10010111 = 1010111110011100$

M		
0110 0010		
A	Q	
0000 0000	1001 0111	
0110 0010	1001 0111	add
0011 0001	0100 1011	shift
0110 0010		
1001 0011	0100 1011	add
0100 1001	1010 0101	shift
0110 0010		
1010 1011	1010 0101	add
0101 0101	1101 0010	shift
0010 1010	1110 1001	shift
0110 0010		
1000 1100	1110 1001	add
0100 0110	0111 0100	shift
0001 0001	1001 1101	2x shift
0110 0010		A-M berechnen, da Q(0)=1; Q(7):=0
1010 1111	1001 1100	

(d) $11100010 \times 10010111 = 0001100010011100$

M		
1110 0010		
A		Q
0000 0000	1001 0111	
1110 0010	1001 0111	add, (M(0)=1 und Q(7)=1) → F:=true
1111 0001	0100 1011	shift
1110 0010		
1101 0011	0100 1011	add
1110 1001	1010 0101	shift
1110 0010		
1100 1011	1010 0101	add
1110 0101	1101 0010	shift
1111 0010	1110 1001	shift
1110 0010		
1101 0100	1110 1001	add
1110 1010	0111 0100	shift
1111 1010	1001 1101	2x shift
1110 0010		A-M berechnen, da Q(0)=1; Q(7):=0
0001 1000	1001 1100	

3. $01011 \div 01101 = 01101$ Rest = 00111

M		
0.1101		
A		Q
0.1011	0.0000	
1.0011		-M (also Addieren des 2-Komplements)
1.1110	0.0000	Ergebnis negativ
1.1110	0.0000	→ Q(4):= 0
1.1100	0.0000	left-shift
0.1101		+M
0.1001	0.0000	Ergebnis positiv
0.1001	0.0001	→ Q(4):= 1
1.0010	0.0010	left-shift
1.0011		-M
0.0101	0.0010	Ergebnis positiv
0.0101	0.0011	→ Q(4):= 1
0.1010	0.0110	left-shift
1.0011		-M
1.1101	0.0110	Ergebnis negativ
1.1101	0.0110	→ Q(4):= 0
1.1010	0.1100	left-shift

Jetzt ist der Rest negativ, also Korrekturschritt:

0.1101	0.0001	+M, +0.0001
0.0111	0.1101	ergibt: 13 Rest 7

Kontrolle:

$$\text{Ergebnis: } 0.1101 = \frac{13}{16} = 0,8125. \text{ Rest: } 0.0111 = \frac{7}{16} \cdot \frac{1}{13} = \frac{7}{208} = 0,033765$$

$$\frac{13}{16} + \frac{7}{13 \cdot 16} = \frac{13 \cdot 16 + 7}{13 \cdot 16} = \frac{11}{13}$$

Man könnte sich auch überlegen, dass der erste Addier- und Schiebeschritt nur ein Ausrichtungsschritt ist, so dass der Korrekturschritt dem 4. Schritt des Algorithmus entspricht. Allerdings wird bei diesem auf das *left-shift* verzichtet.

Anmerkung: vom Ergebnis der Addition (Subtraktion) hängt das nächste Addieren (Subtrahieren) ab, nicht davon, ob nach dem *left-shift* die Zahl auf einmal negativ geworden ist. (Nach dem ersten *left-shift* wird das vormalig positive Ergebnis „negativ“, es wird trotzdem anschließend subtrahiert).