Aufgabe 1.2 Umrechnung von Zahlen

Vervollständigen Sie folgende Tabelle

binär	hexadezimal
1000 1100 1110 0001	
1010 0101 1001 0110	
1010 0001 0000 0110	A106
	1248

Dezimal	Hexa
0	0
1	2
⋮	
9	9
10	A
11	B
12	C
13	D
14	E
15	F
16	10

$10 = 1 \cdot 16 + 0 \cdot 16^0$

$A106 =$

$= \underline{1010}\ \underline{0001}\ \underline{0000}\ \underline{0110}$

$A = 10 = \underset{2^3}{8} + 2 = 1010$

$6 = \underset{2^2}{4} + 2 = 0110$

$2^3 + 2^2 + 2^1 = (1110)_2$

Ergänzen Sie die Tabelle (alle Angaben im **Hexadezimalsystem** und vom Typ **integer**)

A	-A (Vorzeichen/Betrag)	-A (Einerkomplement)	-A (Zweierkomplement)
0001	8001	FFFE	FFFF
FFF0			

$A = (0001)_{hex} = (\underline{0000}\ \underline{0000}\ \underline{0000}\ \underline{0001})_2$

$-A = (1000\ 0000\ 0000\ 0001)_2$

Vorzeichen

$= (8\ 0\ 0\ 1)_{hex}$

$-A = (1111\ 1111\ 1111\ 1110)_2$

Einer-komplement

$= (F\ F\ F\ E)_{hex}$

$-A = (1111\ 1111\ 1111\ 1111)_2$

$= (F\ F\ F\ F)_{hex}$

3.1 Darstellung durch Vorzeichen und Betrag

Dies ist die Darstellung, an die man spontan denken würde, weil si
bekannt ist. Übertragen auf das Binärsystem bedeutet dies, dass das I
zeichen interpretiert wird und die restlichen drei Bit den Betrag der
dann also folgende Tabelle:

Zahl	binär	Zahl	binär
0	0000	(-0)	1000
1	0001	-1	1001
2	0010	-2	1010
3	0011	-3	1011
4	0100	-4	1100
5	0101	-5	1101
6	0110	-6	1110
7	0111	-7	1111

3.3 Darstellung im Zweierkomplement

Negative Zahlen werden hier so dargestellt, dass sie das sog. Zweierkomplement der entsprechenden positiven Zahl sind (Invertieren aller Bits und Addition von Eins oder Subtraktion von Eins und dann Invertieren aller Bits). Man erhält dann folgende Codierung.

Zahl	binär	Zahl	binär
0	0000	-1	1111
1	0001	-2	1110
2	0010	-3	1101
3	0011	-4	1100
4	0100	-5	1011
5	0101	-6	1010
6	0110	-7	1001
7	0111	-8	1000

$1111 = (-1) \cdot 2^3 + 2^2 + 2^1 + 2^0$

$11101 = (-1) \cdot 2^4 + 2^3 + 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$

A	-A (Vorzeichen/Betrag)	-A (Einerkomplement)	-A (Zweierkomplement)
0001			
FFF0	7FF0	000F	0010

$A = (FFF0)_{hex} = 1111 \quad 1111 \quad 1111 \quad 0000 =$

$-A = 0111 \quad 1111 \quad 1111 \quad 0000$

$-(-5) = 5 \quad = \quad 7 \quad F \quad F \quad 0$

$-A = 0000 \quad 0000 \quad 0000 \quad 1111$

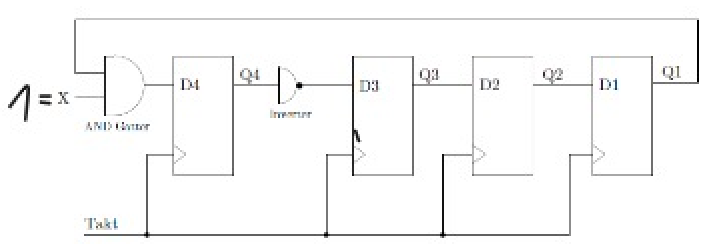
$= 0 \quad 0 \quad 0 \quad F$

$-A = 0000 \quad 0000 \quad 0001 \quad 0000$

$= 0 \quad 0 \quad 1 \quad 0$

$$\begin{array}{r} 1011 \\ +\ 1111 \\ \hline 11010 \end{array}$$

$$\begin{array}{r} 001111 \\ +\ 000001 \\ \hline 10000 \end{array}$$



A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

Aufgabe 1.5 Zustände eines Schaltwerks

Bestimmen Sie die Zustände, die durchlaufen werden, wenn dieses Schaltwerk zu Beginn den Zustand 0000 hat und dann getaktet wird. Beachten Sie dabei, dass die Flipflops gleichzeitig schalten, wenn auf dem Takteingang eine Vorderflanke kommt!

X=0

Takt	Zustand Z Q4	Q3	Q2	Q1
0	0	0	0	0
1	0	1	0	0
2	0	1	1	0
3	0	1	1	1
4	0	1	1	1
5	0	1	1	1
6	0	1	1	1
7	0	1	1	1
8	0	1	1	1
9	0	1	1	1
10	0	1	1	1

X=1

Takt	Zustand Z Q4	Q3	Q2	Q1
0	0	0	0	0
1	0	1	0	0
2	1	1	1	0
3	0	1	1	1
4	1	1	1	1
5	1	0	1	1
6	1	0	0	1
7	1	0	0	0
8	0	0	0	0
9	0	0	0	0
10	0	1	1	0

Aufgabe 1.7 ALU-Operationen

Tragen Sie in die Tabelle ein, was die ALU für folgende Eingaben am Ausgang liefern würde. Alle Zahlen sind hexadezimal zu verstehen und die zugrunde liegende Zahlendarstellung ist das Zweierkomplement:

ALU_OPC	A	B / imm4	Cin	Result	Cout
00	FFFF	0000	0	0000	
01	FFFF	0002	0	0001	
01	000A	0006	1	0010	
02	000A	0006	1	0011	0011
03	0001	0002	0		
04	1234	000F	1		
05	1234	000F	0		
06	0770	673B	0		
07	FFFF	0001	1		
08	0001	0001	1		

$$\begin{array}{r} FFFF \\ +\ 0002 \\ \hline 10001 \end{array} \qquad \begin{array}{r} 000A \\ +\ 0006 \\ \hline 0010 \end{array}$$

$$\begin{array}{r} DF \\ +\ 87 \\ \hline 166 \end{array}$$

ALU_OPC	Result, Cout
0 0000	B
0 0001	A + B
0 0010	A+B+Cin, update Cout
0 0011	A - B
0 0100	A AND B
0 0101	A OR B
0 0110	A XOR B
0 0111	NOT A
0 1000	A << B.<3:0>
0 1001	A >>> B.<3:0>
0 1010	A >> B.<3:0>
0 1011	undefined
0 1100	A << 1, Cout = A.<15>
0 1101	A >>> 1, Cout = A.<0>
0 1110	A >> 1, Cout = A.<0>
0 1111	undefined
1 0000	A, Cout = (A == B)
1 0001	A, Cout = (A != B)
1 0010	A, Cout = (A > B)
1 0011	A, Cout = (A < B)
1 0100	IMM4
1 0101	A + IMM4
1 0110	A - IMM4
1 0111	A AND IMM4
1 1000	A << IMM4
1 1001	A >>> IMM4
1 1010	set bit IMM4 in A
1 1011	clear bit IMM4 in A
1 11**	undefined