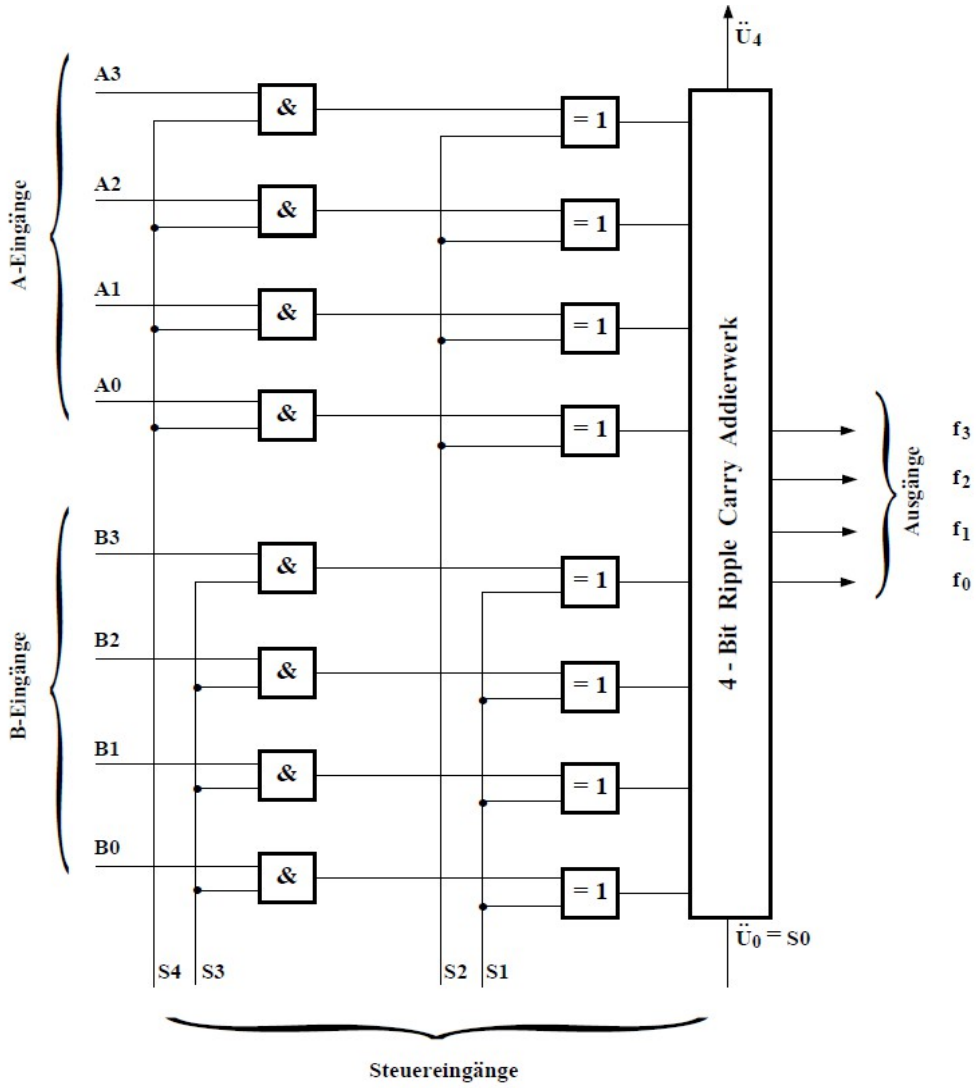
- **Das zugrundeliegende Schaltnetz: Addier/Subtrahier-Schaltnetz**

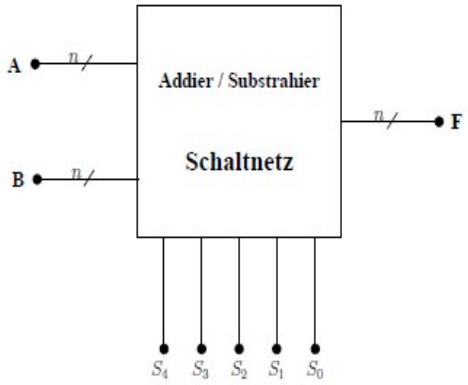
Als ein Beispiel für die Entwicklung einer einfachen ALU wird das Addier/Subtrahier-Schaltnetz im Kurstext 1608 wie folgt eingeführt:

Neben der arithmetischen Verknüpfung Addition ist die Subtraktion ebenso wichtig. Die Subtraktion kann durch ein eigenes Schaltnetz realisiert werden oder auf die Addition zurückgeführt werden. Wird die Subtraktion auf die Addition zurückgeführt, dann ist für Addition und Subtraktion nur ein Schaltnetz, z.B. ein Paralleladdierer, erforderlich.Die Funktionsfähigkeit des Paralleladdierers muss durch ein zusätzliches Schaltnetz erweitert werden (...).
Folgende Abbildung zeigt ein Schaltnetz, mit dem Additionen und Subtraktionen von Variablen A und B, mit A und B jeweils 4-Bit-Zahl, durchgeführt werden können:



In folgender Abbildung ist das Addier/Subtrahier-Schaltnetz zu einem Blockschaltbild zusammengefasst. Das n mit dem Schrägstrich bei A, B und F deutet an, dass es sich hier eigentlich um n Leitungen handelt. In folgenden Tabelle sind alle Verknüpfungen enthalten, die mit dem Addier/Subtrahier-Schaltnetz ausgeführt werden können.

				$\ddot{U}_0 = 0$	$\ddot{U}_0 = 1$
S_4	S_3	S_2	S_1	F	F
0	0	0	0	0	1
0	0	0	1	-1	0
0	0	1	0	-1	0
0	0	1	1	-2	-1
0	1	0	0	B	B + 1
0	1	0	1	$\overline{B}$	$\overline{B}$ +1 = -B
0	1	1	0	B -1	B
0	1	1	0	-B -2	-B -1 = $\overline{B}$
1	0	0	0	A	A +1
1	0	0	1	A -1	A
1	0	1	0	$\overline{A}$	$\overline{A}$ + 1 = -A
1	0	1	1	-A -2	$\overline{A}$
1	1	0	0	A +B	A +B +1
1	1	0	1	A -B -1	A -B
1	1	1	0	B -A -1	B -A
1	1	1	1	-A -B -2	-A -B -1



Quelle: Computersysteme I (2017), Kapitel 2.9 Arithmetik-Logik Einheit (ALU)

- **Die Hades Simulation:**

Der als letzte Seite angehängte Screenshot zeigt die Simulation in ihrem undefinierten Startzustand, zu erkennen an den cyanfarbenen Schaltern und Verbindungen. Links sind die 8 Eingangssignale für die Eingangsvariablen $A = A_3 A_2 A_1 A_0$ und $B = B_3 B_2 B_1 B_0$ zu sehen, unten die fünf Eingangssignale für das Steuerwort $S = S_4 S_3 S_2 S_1 S_0$. Am rechten Rand erscheinen die fünf Ausgangssignale, vier für die Ausgänge $f_0, \ldots, f_3$ und einer für den Ausgangsübertrag $\ddot{U}_4$.

Belegt man nun die Eingangsvariablen und die Steuersignale durch Klick auf die entspechenden Schalter mit 0 (grau) oder 1 (rot), kann man die UND- und ODER- Verknüpfungen nachverfolgen und die Ausgabe des 4-Bit-Ripple-Carry-Addiereres betrachten.

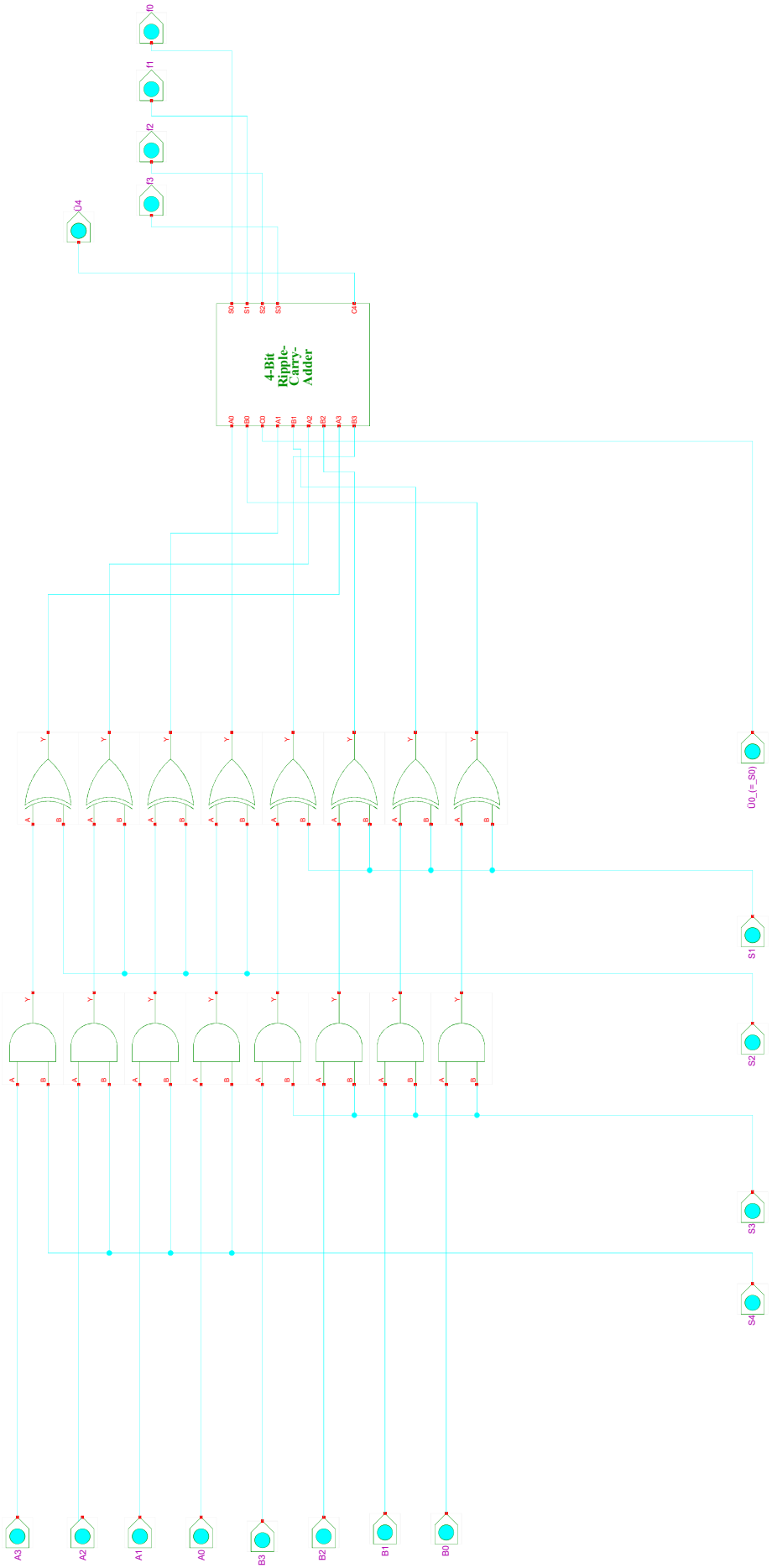
Beispiele:

Belegt man $A = A_3A_2A_1A_0 = 0010$ und $S = S_4S_3S_2S_1 = 1000$ dann erreicht man mit $S_0 = 1$ wie erwartet die Ausgabe $f_3f_2f_1f_0 = 0011 = A + 1$ und mit $S_0 = 0$ die Ausgabe $f_3f_2f_1f_0 = 0010 = A$, wie in Zeile 9 der Funktionstabelle abzulesen.

Belegt man stattdessen $A = A_3A_2A_1A_0 = 0010$ und $B = B_3B_2B_1B_0 = 1000$ und $S = S_4S_3S_2S_1 = 1101$ dann produziert man mit $S_0 = 1$ die Ausgabe $\ddot{U}_4f_3f_2f_1f_0 = 00011 = A - B - 1$ und mit $S_0 = 0$ die Ausgabe $\ddot{U}_4f_3f_2f_1f_0 = 01001 = A - B$, wie nach der Funktionstabelle zu erwarten.

- **Die Simulation besteht aus folgenden Komponenten:**
  - 8 AND2
  - 8 XOR2
  - 5 Opins (LED)
  - 13 Ipins (switch)
  - 1 4-Bit-Ripple-Carry-Adder
- **Besonderheit**:

In dieser Simulation wurden *Subdesigns* verwendet. Das Symbol *4-Bit Ripple-Carry-Adder* versteckt die Implementierung des zugrundeliegenden 4-Bit Ripple-Carry-Addierers mit Eingangsübertrag (siehe Simulation 27 dieser Reihe) und verfügt über die volle Funktionalität des enthaltenen Subdesigns. Wie bei allen verwendeten Subdesigns gilt auch hier: Klickt man mit der rechten Maustaste auf das Symbol und wählt im erscheinenden Popup-Menu den Eintrag edit, so öffnet sich das enthaltene Subdesign im Editor.


I0
I1
I2
I3
U4
S0
S1
S2
S3
C4
4-Bit
Ripple-
Carry-
Adder
A0
B0
C0
A1
B1
A2
B2
A3
B3
Y
A
B
Ü0_(=_S0)
S1
S2
S3
S4
A3
A2
A1
A0
B3
B2
B1
B0