Realisierung von Elementen eines VN-Rechners

Name: **Vorname:** **Klasse:**

Vergleichen und Addieren von Bits – XOR, Halb- und Volladdierer

Die Realisierung des Additionsbefehl im Rechenwerk des von-Neumann-Rechners erfolgt mittels Logikgatter, aber wie?

Für das Addieren zweier Bits an den Eingängen A und B zu Y gilt folgende Wertetabelle.

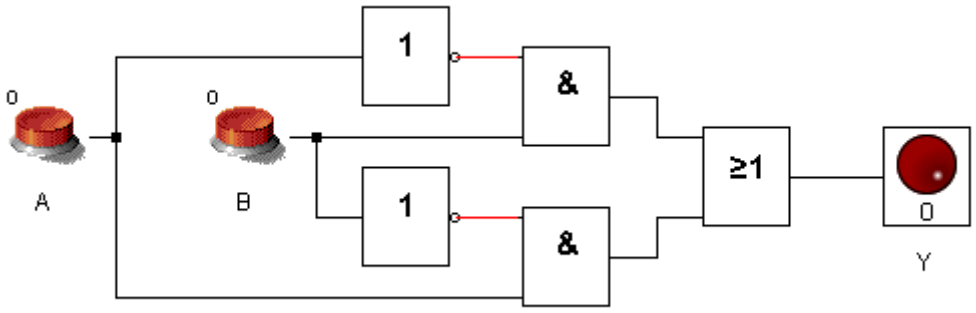
Eingang A		Eingang B		Ausgang Y	Übertrag U
0	+	0	=	0	0
0	+	1	=	1	0
1	+	0	=	1	0
1	+	1	=	0	1

Im letzten Fall wird zusätzlich ein Übertrag U erzeugt, der bei der nächsthöherwertigen Stelle berücksichtigt werden muss.

Feststellung: Kein uns bekanntes Gatter erzeugt die Wertetabelle für A, B und Y.

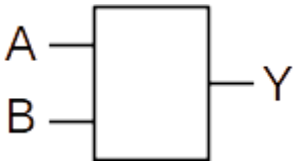
1) Zeigen Sie im Simulator, dass die nachfolgende Schaltung die Wertetabelle für A, B und Y abbildet.

   Hinweis: Leitungsverzweigungen lassen sich mit gedrückter UMSCHALT-Taste realisieren.



2) Stellen Sie eine Hypothese über das Schaltzeichen eines Gatters auf, das die gegebene Wertebelegung hat. Prüfen Sie die Hypothese im Simulator in einer neuen Schaltung. Geben Sie den Namen und das Schaltsymbol des Gatters an.

   Name des Logik-Gatters: _____ Schaltsymbol:



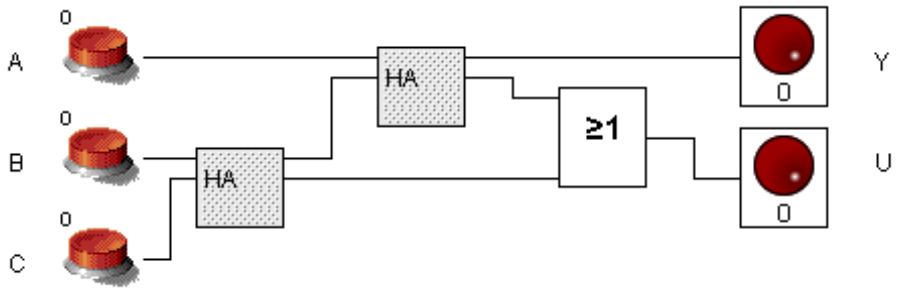
Das gefundene Logikgatter kann somit auch zum **Vergleichen von Bits** verwendet werden.

3) Erweitern Sie die Schaltung so, dass es den Ausgang U zum Anzeigen des Übertrags gibt. Man nennt eine solches Schaltnetz **Halbaddierer HA**.

4) Ein **Volladdierer** arbeitet wie ein Halbaddierer, berücksichtigt aber zusätzlich einen ggf. vorhandenen Übertrag C.
Entwickeln Sie die Wertetabelle für einen Volladdierer mit den Eingängen A, B, und C.

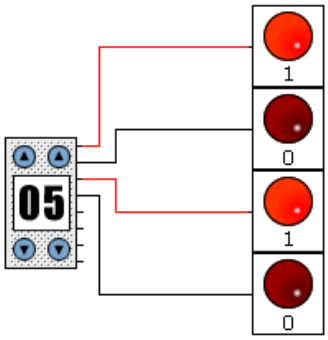
Eingang A	Eingang B	Eingang C	Ausgang Y	Übertrag U

5) Prüfen Sie, ob folgende Schaltung korrekt arbeitet, wenn das Symbol HA für Halbaddierer steht. Man nennt eine solches Schaltnetz Volladdierer, VA bzw. FullAdder.



6) Entwickeln Sie unter Verwendung des Eingabebausteins „Binäreingabe“ und des Ausgabebausteins „LCD“ ein Addierwerk zum Addieren von zwei 8-Bit-langen Zahlen unter Verwendung von Voll- und Halbaddierern.

Hinweis: Der Baustein „Binäreingabe“ nutzt das Hexadezimalsystem. Die rechte Ziffer in der Anzeige wird dabei binär von oben mit der Stelle 2^0 beginnend angezeigt.

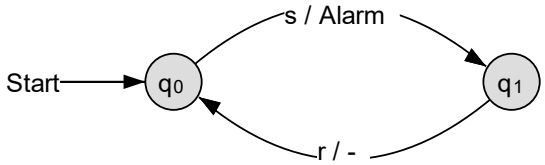


Realisierung von Elementen eines VN-Rechners

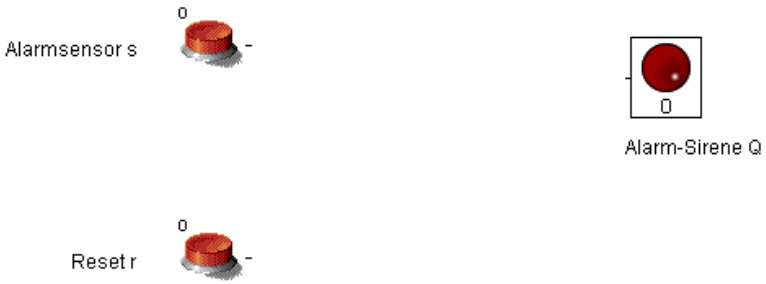
Name: **Vorname:** **Klasse:**

Bits speichern – Flip-Flop – RAM

Bei einfachen Alarmanlagen löst eines Sensors eine Sirene aus, die nur durch manuelles Rücksetzen wieder verstummt. Ein Mealy-Automat kann dies modellieren.



1) Begründen Sie, dass der Mealy-Automat unvollständig dargestellt ist.
   Vervollständigen Sie ihn.
2) Entwickeln Sie unter Verwendung eines ODER-Gatter, eines Schalters und einer LED eine sog. Selbsthalteschaltung, die beim Auslösen des Schalters S die LED dauerhaft leuchten lässt, auch wenn der Schalter S danach deaktiviert wird. Nutzen Sie die Vorgabe in der Datei Alarm.lsim.

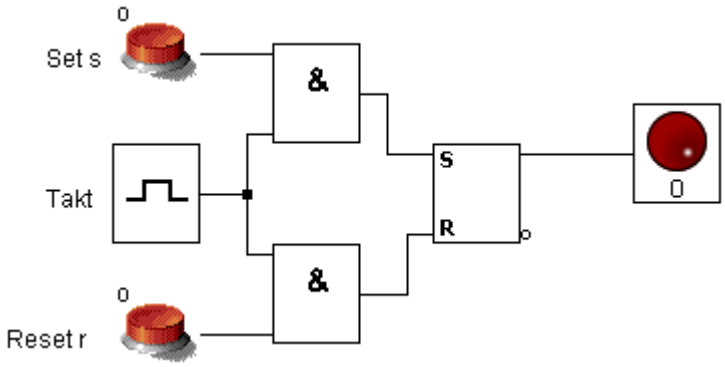


3) Die Schaltung soll um eine Möglichkeit ergänzt werden, die Alarmanlage über ein Signal am Eingang R zurückzusetzen. Dazu müssen die Eingänge des ODER-Gatters den Wert 0 haben, wobei der Wert am unteren ODER-Eingang von der Alarmsignalleuchte und vom Resetknopf abhängen. Vervollständigen Sie die Wertetabelle.

Alarmsignal-leuchte	Resetknopf	Signal für unteren ODER-Eingang
0	0	
0	1	
1	0	
1	1	unbestimmt

4) Erweitern Sie die Schaltung um die Reset-Funktion mithilfe zweier unterschiedlicher Gatter.
   Tipp: Wie würde die Wertetabelle aussehen, wenn Sie das Reset-Eingangssignal zunächst negieren? Welche bekannte Wertebelegung erkennen Sie dann wieder?
5) Die Schaltung der Alarmanlage stellt einen 1-Bit Speicher dar.
   Begründen Sie diese Aussage.

6) Um Speicher- und Verarbeitungsprozesse zu synchronisieren, ist eine Taktsteuerung notwendig. Ermitteln Sie den Zeitpunkt des Speicherns und Löschens eines Bits in der dargestellten Schaltung in LogicSim. Man nennt eine solche Schaltung D-Flip-Flop mit Reset (D-FF).



7) Entwickeln Sie unter Verwendung der Bausteine „D-FlipFlop“, „Binäreingabe“ und „LCD“ eine Speicherzelle für 4 Bit.

8) Vergleichen Sie den Aufbau Ihrer Speicherzelle mit der Schaltung in der Datei RAM.lsim. Untersuchen Sie die Funktionsweise des RAM-Bausteins. Ermitteln Sie die Speichergröße.

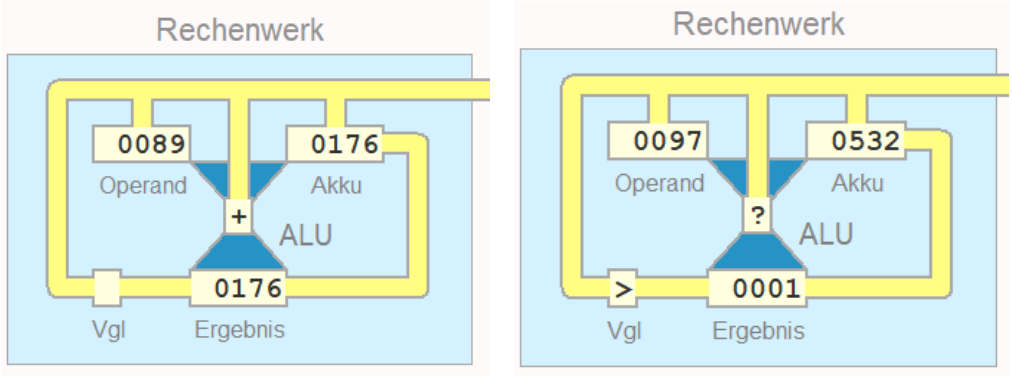
Realisierung von Elementen eines VN-Rechners

Name: **Vorname:** **Klasse:**

Entwicklung eines einfachen Rechenwerks (ALU_Vorgabe.lsim)

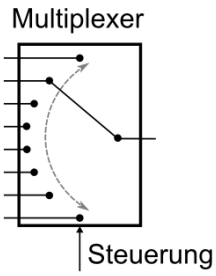
Bringt man die bisher betrachteten Schaltungen so zusammen, dass ein Operand im Akkumulator und der zweite als Parameter zugeführt wird und die Auswahl der Operation irgendwie über das Steuerwerk an Rechenwerk erfolgt, dann hätte man ein Rechenwerk!

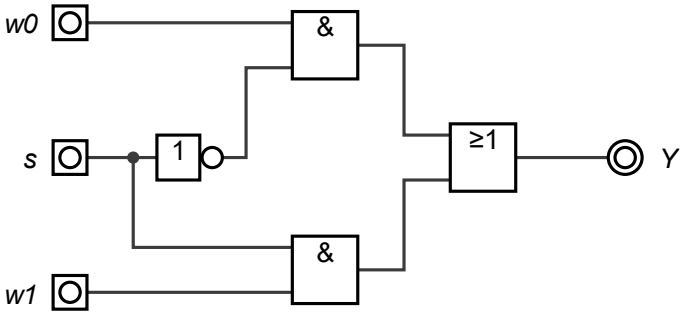
1) Geben Sie unter Verwendung der Bildschirmausschnitte aus dem von-Neumann-Simulator MOPS Operationen/Funktionen an, die ein Rechenwerk ausführen muss.



2) Ein Rechenwerk mit einer 1-Bit-Verarbeitungsbreite soll für die Eingabe A (Akkumulator) und B (Parameter) die Funktionen and(A, B), or(A, B), not(A) sowie add(A, B, C), wobei C ein vorliegender Übertrag ist, realisieren können.
Entwickeln Sie ein Rechenwerk, dass alle Funktionen *gleichzeitig* ausführt und das jeweilige Ergebnis auf einer eigenen LED abbildet. Der bei Addition ggf. entstehenden Übertrag ist durch eine eigenständige LED für das sog. Carry-Flag abzubilden.

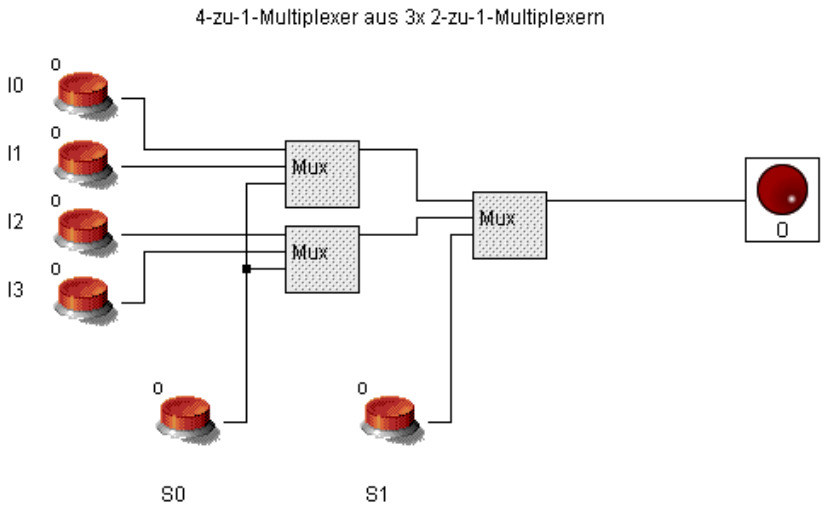
3) Das Durchschalten des gewünschten Ergebnisses wird mithilfe eines sog. Multiplexers realisiert.
Zeigen Sie mithilfe einer Wertetabelle, dass eine zweiseitige Auswahl der Art „WENN s DANN w1 SONST w0“ mit nachfolgende Logikschaltung – einem 2-zu-1-Multiplexer realisierbar ist.





4) Der 2-zu-1-Multiplexer liegt in LogicSim als Modul vor. Prüfen Sie seine Korrektheit.

5) Kaskadiert man drei 2-zu-1-Multiplexer durch nachfolgende Schaltung, so entsteht ein 4-zu-1-Multiplexer. Dieser verfügt über 4 Eingänge, von denen genau einer auf den Ausgang geschalten wird. Die Auswahl erfolgt binär codiert mithilfe zweier Steuersignale s0 und s1.



Erweitern Sie Ihr Rechenwerk um einen 4-zu-1-Mulitplexer (unter Verwendung der obigen Schaltung oder des Moduls 4-zu-1-Multiplexer). In Anhängigkeit von den Steuersignalen S0 und S1 soll die ALU das Ergebnis einer Funktion auf die LED schalten.

Testen Sie Ihre Schaltung.

Geben Sie die Logiktabelle für die Funktionsauswahl der Steuerung an.