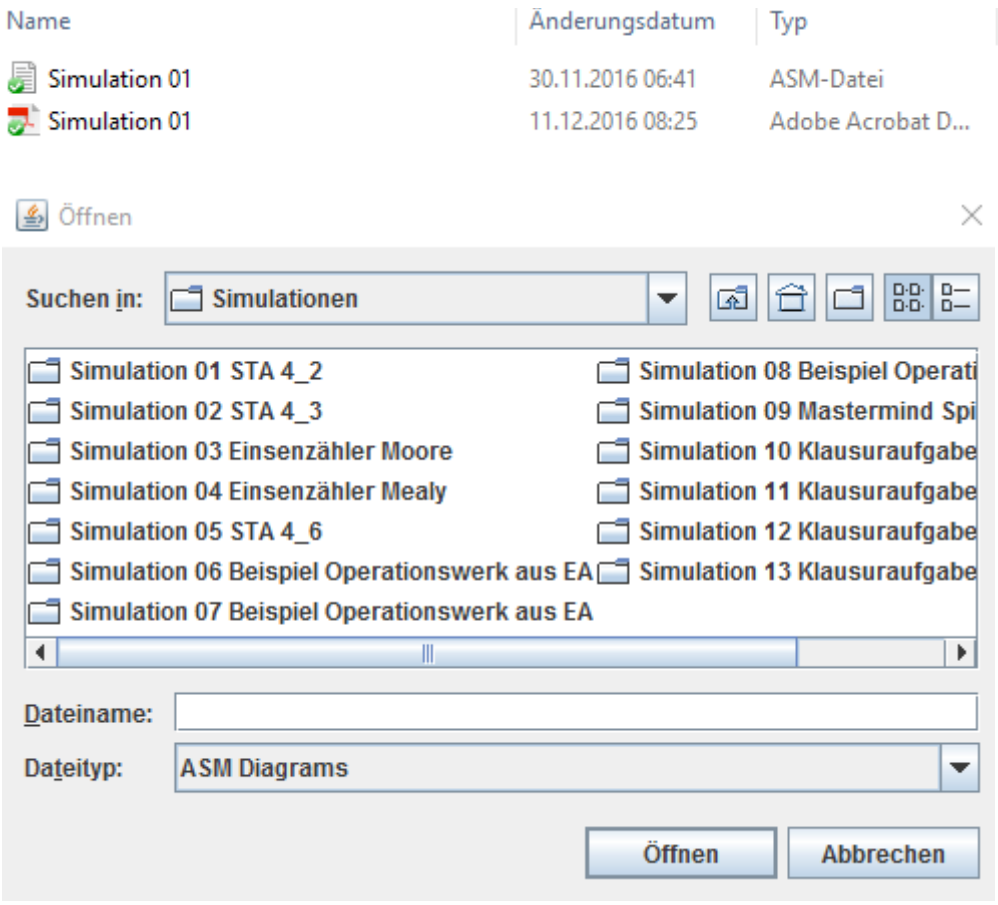
- **Der ASM-Simulator – Grundsätzliches & Wichtiges**
  - Der Simulator ist javabasiert, sodass also zwingend erforderlich ist, dass ein (aktuelles) JDK/JRE (oder OpenJDK) auf Ihrem System vorhanden ist.
  - Die Software ist im Rahmen einer Bachelorarbeit am Lehrgebiet Rechnerarchitektur entstanden, Sie werden also, von dieser Einführung hier abgesehen, keine weitergehenden Informationen im Netz darüber finden.
  - Diese Kurzanleitung soll einen kurzen Einblick in die Funktionsweise des Simulators geben, sodass er Unterstützung beim Anfertigen und Verstehen von ASM-Diagrammen sein kann.
- **Erste Schritte → vorhandene Simulationen**
  - über die Konsole: wechseln Sie in das Verzeichnis, in dem die .jar Datei liegt und öffnen Sie sie mit: java -jar „ASM-Simulator.jar“, abhängig von Ihren Systemeinstellungen reicht eventuell ein Doppelklick.
  - Über das Menü *File → Open* können Sie im Ordner *ASM/Simulationen* die 13 verfügbaren Simulationen finden und öffnen. In jedem der Ordner liegt sowohl die .asm Datei mit der Simulation als auch eine pdf-Kurzbeschreibung. Es ist sinnvoll, diese pdf ebenfalls zu öffnen:

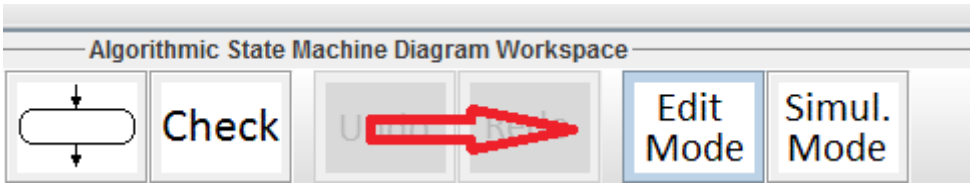


- die vorgefertigten Simulationen wurden anhand von Kurstextbeispielen, Selbsttestaufgaben, Einsendeaufgaben und Klausuraufgaben vergangener Semester erstellt. Sie können eventuell dabei helfen, die Funktionsweise und grundlegenden Eigenschaften von ASM-Diagrammen leichter nachzuvollziehen, ersetzen aber natürlich nicht das gründliche Auseinandersetzen mit dem Kurstext!

- Im Folgenden wird nach der Vorstellung einiger Grundfunktionen ein kleines ASM-Diagramm entworfen, um den Umgang mit dem Simulator zu zeigen. Die meisten Handgriffe sind selbsterklärend, und wer schon einmal ein ASM-Diagramm mit der Hand oder einem beliebigen Zeichenprogramm gezeichnet hat, wird die unkomplizierte Herangehensweise zu schätzen wissen.

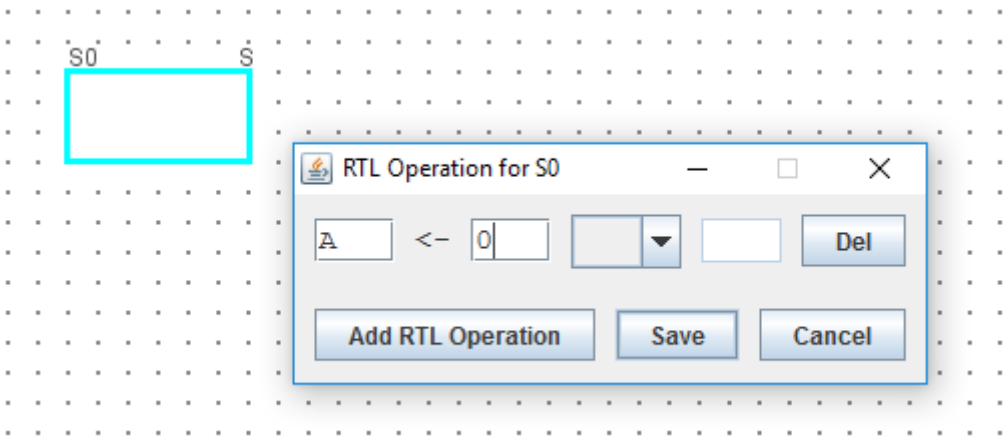
• Erste Schritte → der Simulator

- Bei Aufruf befindet sich der Simulator im Editiermodus, zu erkennen am farblich hinterlegten Button *Edit Mode* in der obigen Menüzeile:



- In diesem Modus werden die Diagramme erstellt und nach einem Klick auf *Check* (s.o.) wechselt man in den Simulationsmodus durch Klick auf *Simul.Mode*, um dann das komplexe Schaltwerk schrittweise zu simulieren.

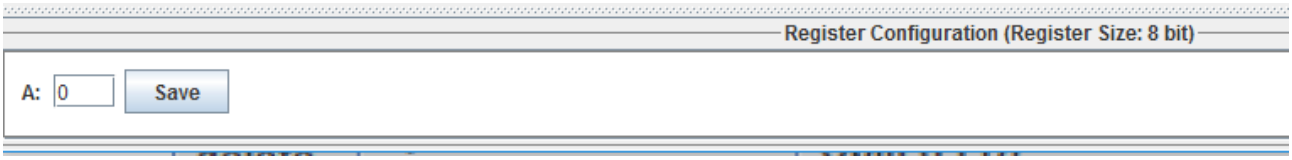
- Doppelklick auf den bei Simulator-Aufruf automatisch erschienenen Startzustand S_0 öffnet den Eingabedialog, über den nun die für diesen Zustand vorgesehenen Operationen in RTL-Notation eingegeben werden können:



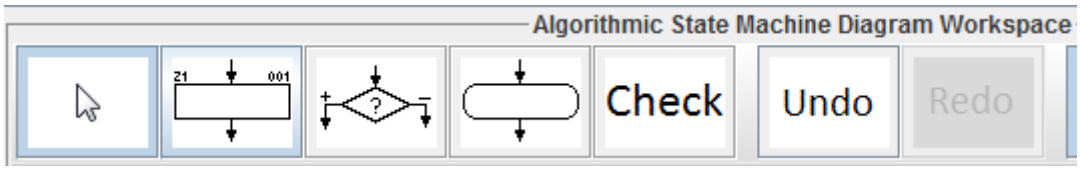
- Mit Klick auf *Save* wird (in diesem Fall) die Initialisierung des Registers A mit dem Wert 0 bestätigt und im unteren Bereich *Register Configuration* erscheint das Register A:

  - Vorher:

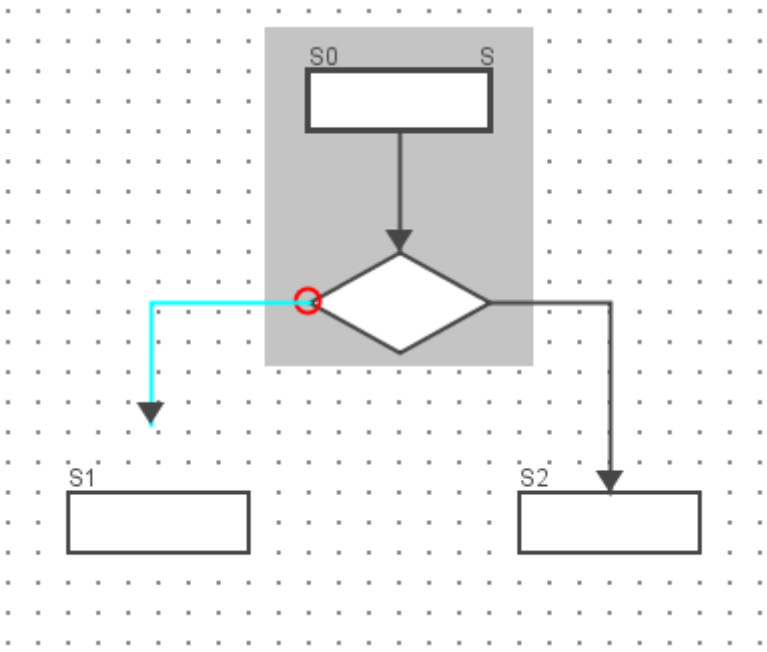
  - Nachher:



- Über die Buttons *Statebox*, *Condition Box* und *Conditional Output* können nun die Boxen ausgewählt werden, die man für sein Diagramm benutzen möchte, Klick auf *Undo* macht den letzten Schritt rückgängig:



- Durch einfaches Klicken & Ziehen werden die Boxen miteinander verbunden:

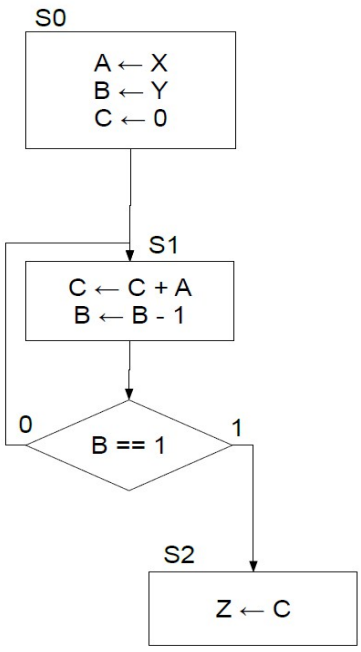


- rechts im Bereich *ASM Simulation* befindet sich eine Tabelle mit den verwendeten Registern, ihrer Belegung in jedem Schritt und ebenfalls eine Anzeige, in welchem Takt und Zustand die Simulation sich befindet. Folgende Abbildung zeigt die Registerbelegung nach einmaligem Durchlauf eines komplexen Schaltwerkes für die Multiplikation zweier Binärzahlen, in diesem Fall wurde die Multiplikation von 2 und 3 simuliert:

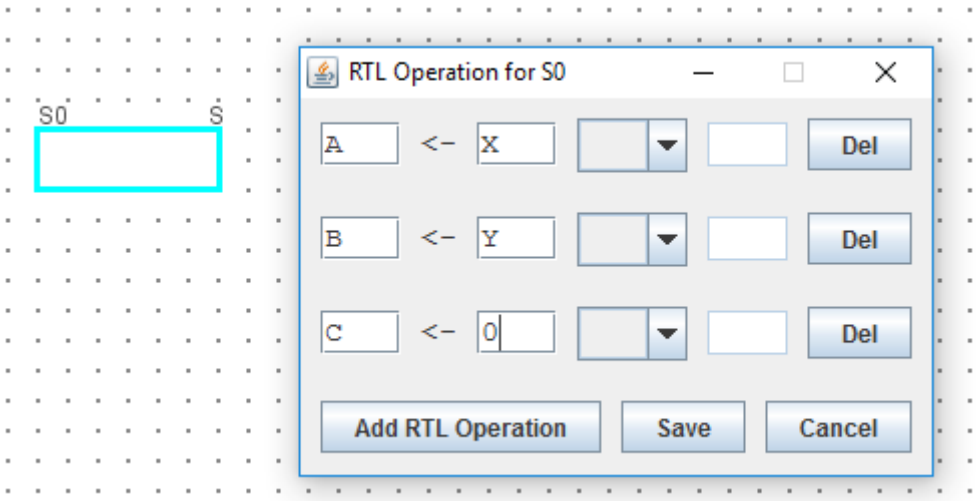
register	cycle # : state id					
	1 / S1	2 / S1	3 / S1	4 / S2	5 / S0	
A	2	2	2	2	2	
B	3	2	1	0	0	
C	0	2	4	6	6	
X	2	2	2	2	2	
Y	3	3	3	3	3	
Z	0	0	0	0	6	

- **Erste Schritte → ASM-Diagramm erstellen**

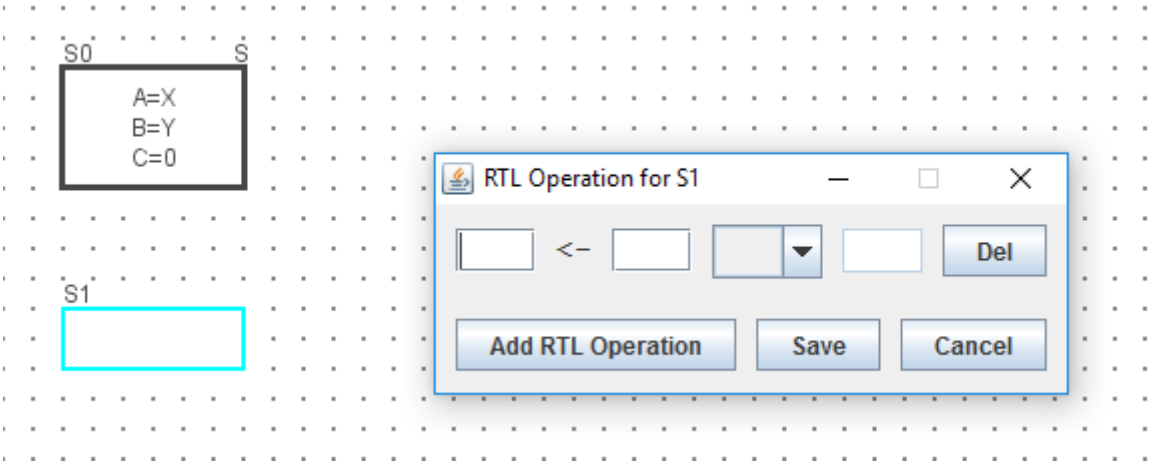
  - Editor öffnen
  - Simuliert werden soll folgendes ASM-Diagramm:



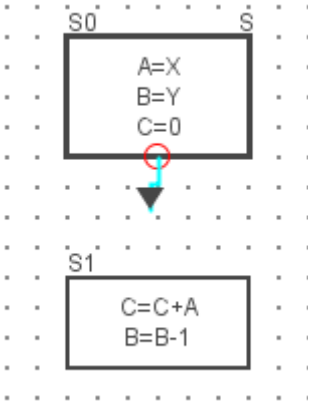
- Wir beginnen mit den RTL-Operationen für den Startzustand S_0



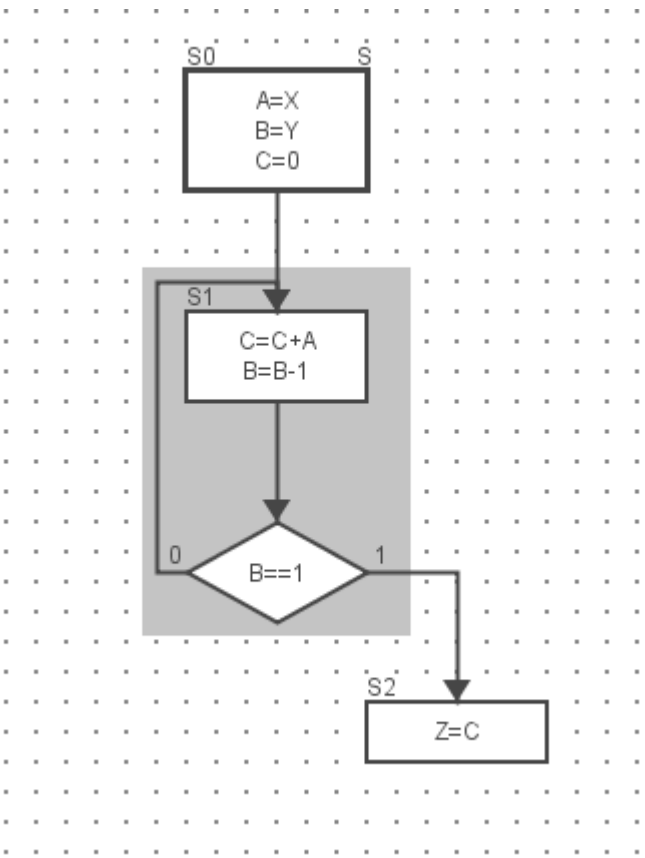
- und mit einfachem Klick auf die Zustandsbox in der oberen Menüzeile können wir einen weiteren Zustand, also S_1, im Zeichenbereich platzieren und seinen Eingabedialog ebenfalls durch Doppelklick öffnen:



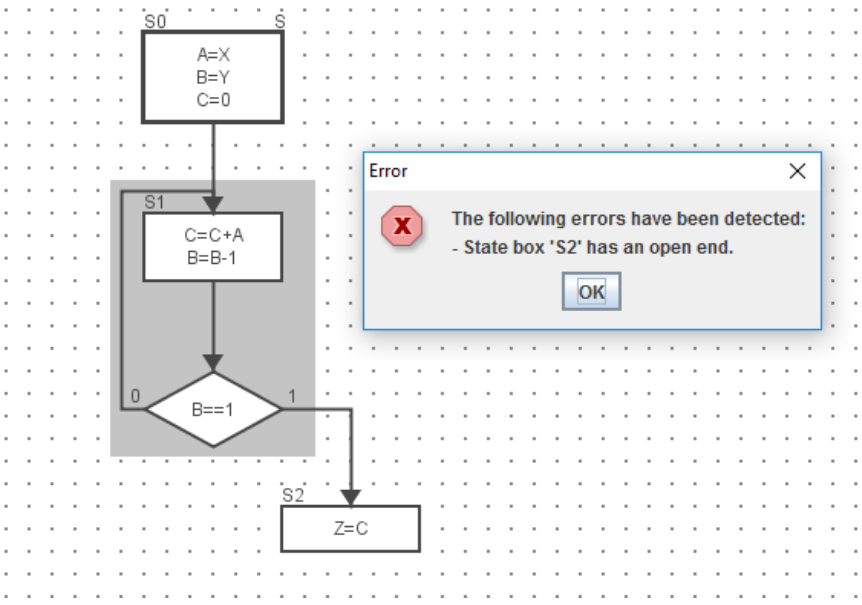
- Anschließend werden die beiden Zustandsboxen von S0 ausgehend verbunden:



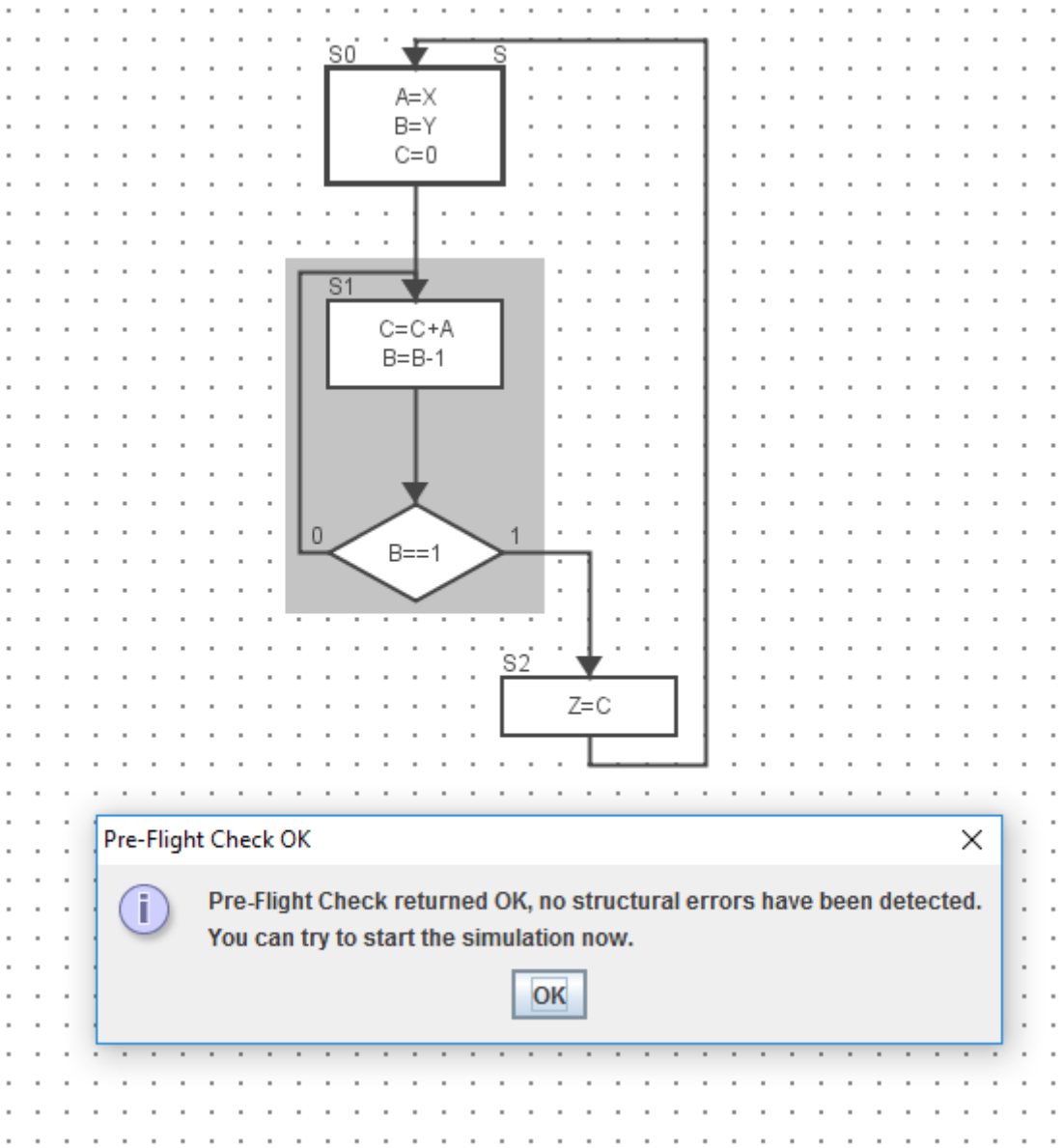
- Ebenso wird für die Entscheidungsbox und den Endzustand S_2 verfahren:



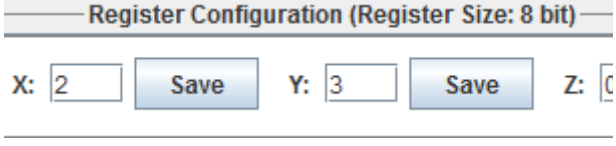
- Nun entspricht das Diagramm dem der Vorgabe. Klick auf *Check* überprüft das Design auf strukturelle Fehler und ergibt folgendes Problem:



- Der Simulator unterstützt Zustände ohne Folgezustände nicht, sodass obige Fehlermeldung erscheint. Dieses Problem wird einfach gelöst, indem Endzustände erneut in die Startzustände geführt werden:



- Nun müssen den Eingangsvariablen Werte zugewiesen werden, die Faktoren X und Y werden über das Feld *Register Configuration* gesetzt, indem das Feld neben der jeweiligen Registerbezeichnung gefüllt und mit Klick auf *Save* bestätigt wird:



- Damit ist die Simulation bereit für einen Testdurchlauf, in diesem Fall mit X=2 und Y=3. Klick auf *Simul.Mode* lässt den Simulator in den Simulationsmodus wechseln, zu erkennen an den nun aktiven Steuerbuttons im rechten Bereich *ASM Simulation*:

- Editiermodus:

- Simulationsmodus:

• Erste Schritte → Simulationsdurchlauf

Takt 0:

Zu Beginn befinden wir uns im Startzustand S_0, die Registerbelegung entspricht den Eingaben im unteren Bereich *Register Configuration.* A und B werden mit den Werten der Variablen X und Y beschrieben, C wird mit 0 initialisiert.

Gemäß Regel 3 der *wesentlichen Regeln zum Arbeiten mit ASM-Diagrammen (sh. Kurstext, Kapitel 4.4.4)* werden diese Zuweisungen erst bei Eintritt in den Folgezustand gültig:

register	cycle # : state id	
	0 / S0	
A	0	
B	0	
C	0	
X	2	
Y	3	
Z	0	

Takt 1:

Wir sind im ASM Block S_1 angekommen, die Zuweisungen aus S_0 sind gültig, die Schleife beginnt. A wird zum Inhalt von C addiert, während B dekrementiert wird:

register	cycle # : state id		
	0 / S0	1 / S1	
A	0	2	
B	0	3	
C	0	0	
X	2	2	
Y	3	3	
Z	0	0	

Takt 2:

Wir sind wieder in S_1, weil die Entscheidungsbox zu *nein* ausgewertet wurde. Die Subtraktion von 1 von B wurde erst jetzt, beim Wiedereintritt in S_1 gültig, in der Entscheidungsbox hatte B noch den Wert 3, der in S_0 zugewiesen wurde. Jetzt ist B=2 und wird erneut dekrementiert, was aber in der Abfrage noch nicht berücksichtigt werden kann (Regel 3!):

register	cycle # : state id			
	0 / S0	1 / S1	2 / S1	
A	0	2	2	
B	0	3	2	
C	0	0	2	
X	2	2	2	
Y	3	3	3	
Z	0	0	0	

Takt 3:

Wir sind wieder in S_1, weil die Entscheidungsbox zu *nein* ausgewertet wurde. Bei Wiedereintritt in S_1 hat B nun den Wert 1.

register	cycle # : state id				
	0 / S0	1 / S1	2 / S1	3 / S1	
A	0	2	2	2	
B	0	3	2	1	
C	0	0	2	4	
X	2	2	2	2	
Y	3	3	3	3	
Z	0	0	0	0	

Takt 4:

Nun sind wir in S_2, denn obwohl B in S_1 noch einmal dekrementiert wurde und jetzt, bei Eintritt in S_2, den Wert 0 hat, stand zum Zeitpunkt der Abfrage, nämlich in der S_1 ASM-Box, noch 1 in B, sodass die Entscheidungsbox zu *ja* ausgewertet wurde. A wurde drei mal (also B-mal) zu C addiert, es folgt C = 6.

register	cycle # : state id				
	0 / S0	1 / S1	2 / S1	3 / S1	4 / S2
A	0	2	2	2	2
B	0	3	2	1	0
C	0	0	2	4	6
X	2	2	2	2	2
Y	3	3	3	3	3
Z	0	0	0	0	0

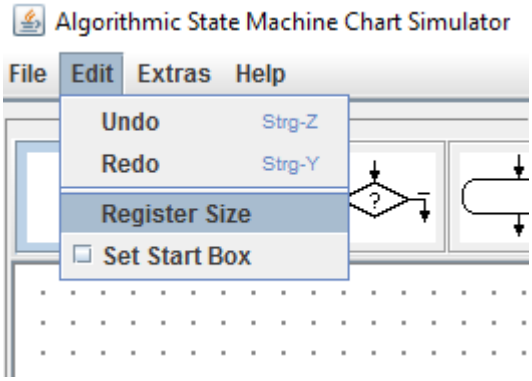
Takt 5:

In S_2, dem eigentlichen Endzustand, wird das berechnete Produkt A * B, das in C steht, in Z geschrieben und liegt somit am Ende von Takt 4 am Ausgang Z an, gültig und sichtbar in Takt 5.

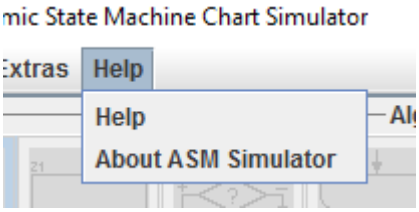
Damit ist ein Durchlauf abgeschlossen, das Produkt korrekt berechnet und es können bei Bedarf anderen Werte für X und Y eingegeben werden.

register	cycle # : state id				
	1 / S1	2 / S1	3 / S1	4 / S2	5 / S0
A	2	2	2	2	2
B	3	2	1	0	0
C	0	2	4	6	6
X	2	2	2	2	2
Y	3	3	3	3	3
Z	0	0	0	0	6

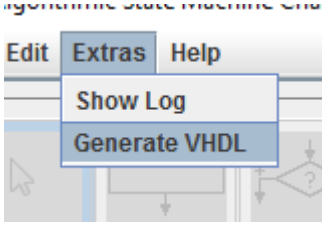
- **Tipps**
  - Ist es für ein komplexes Schaltwerk nötig, die Registerbreite von (den standardmäßig voreingestellten) 8 Bit zu ändern, kann das über die Menüzeile *Edit* → *Register Size* gemacht werden, Werte von 0 bis 63 sind möglich:



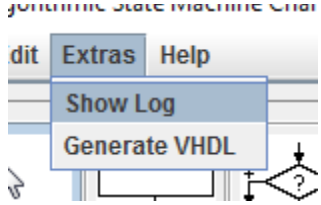
- Über die Menüzeile erreicht man mit *Help* → *Help* eine kurze, englischsprachige Anleitung:



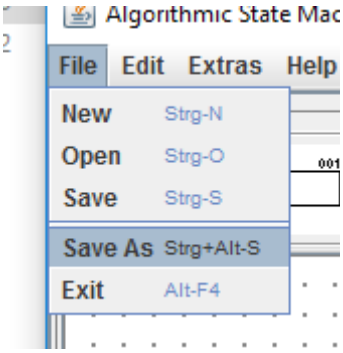
- Fertige Diagramme können über die Menüzeile mit *Extras* → *Generate VHDL* in VHDL Code exportiert werden:



- Ebenfalls über die Menüzeile, mit *Extras* → *Show Log,* kann ein Logfile eingesehen werden, das jeden Takt protokolliert:



- Fertig erstellte Diagramme werden über File → Save As unter einem beliebigen Namen an einem beliebigen Ort als .asm Datei gespeichert:



Viel Spaß & Erfolg mit den ASM-Diagrammen!