- **Das zugrundeliegende Schaltnetz: Ehemalige Einsendeaufgabe**

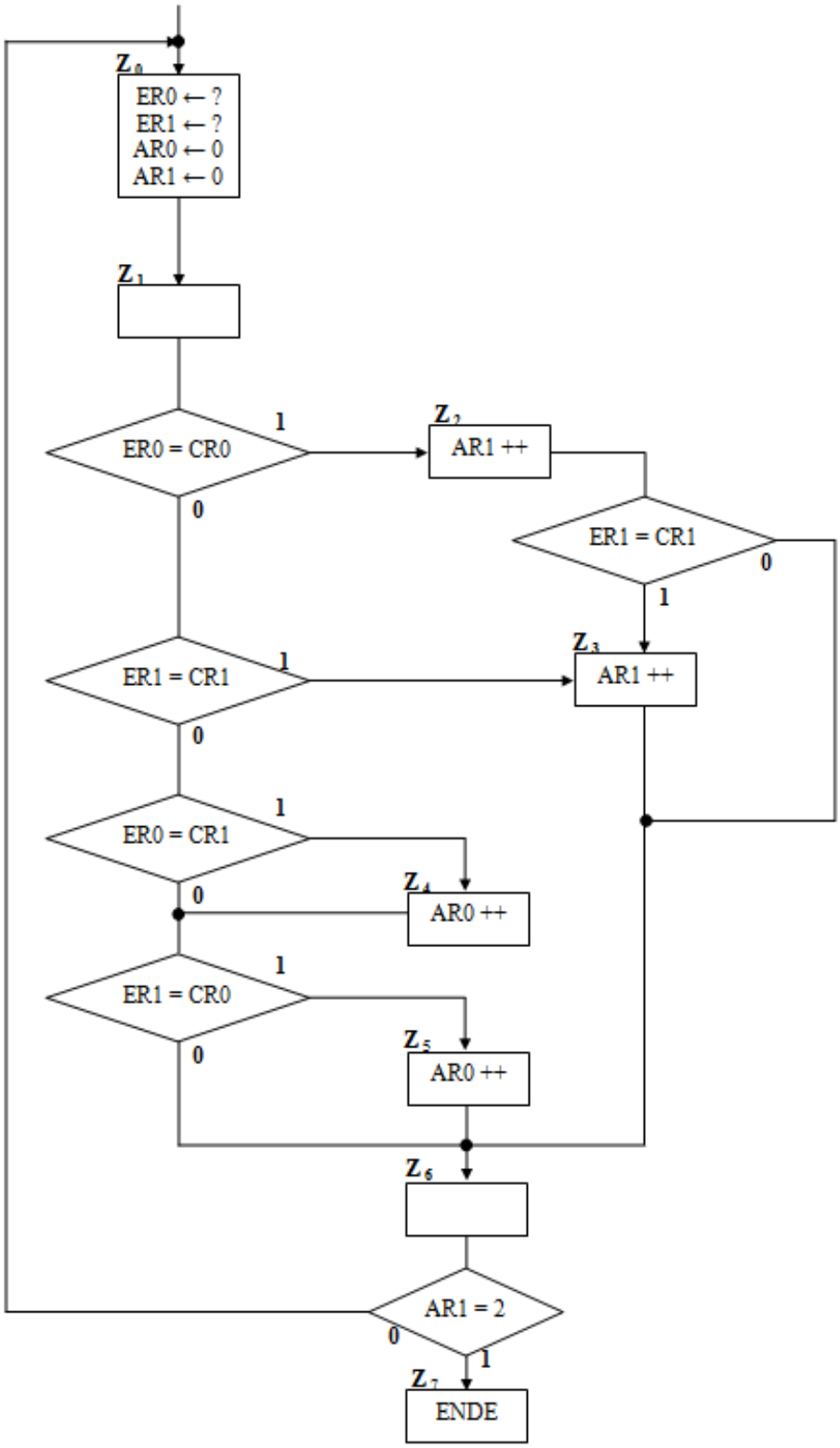
Entwerfen Sie ein komplexes Schaltwerk, das ein vereinfachtes Mastermind-Spiel realisiert. Bei dem Spiel gibt es lediglich drei Farben (rot, grün und blau) und 2 Positionen (rechts und links). Für das Schaltwerk werden die folgenden sechs 2-Bit-Register benötigt:

2 Coderegister (CR0, CR1),
2 Eingaberegister (ER0, ER1) und
2 Ausgaberegister (AR0, AR1).

Die Farben sind wie folgt codiert:
rot = 00,
blau = 01,
grün = 10.

In den beiden Coderegistern stehen schon zu Beginn die ”geheimen“ Farbwerte für die Positionen (CR0 = links, CR1 = rechts). Nach erfolgter Benutzereingabe der geschätzten Farben in den beiden Eingaberegistern werden diese mit den Farben in den Coderegistern verglichen. Es gibt zwei Ausgaberegister:
AR0 enthält die Anzahl der richtigen Farben, die sich jedoch an der falschen Position befinden und AR1 enthält die Anzahl der richtigen Farben, die auch an der richtigen Position sind. Falls AR1mit der Anzahl der Positionen überein-stimmt, ist das Spiel beendet, ansonsten wird eine neue Eingabe erwartet. Die ”geheimen“ Farbwerte und die Eingabe können sowohl verschiedene als auch gleiche Farben für beide Positionen enthalten. Die Überprüfung einer Farbe kann sich in jeder Runde höchstens auf ein Ausgaberegister AR1 oder AR0 und auch nur in dieser Reihenfolge auswirken! Das heißt, falls z.B. zwei gleiche Farben als Eingabe gewählt wurden, die jedoch nur einmal im ”geheimen“Code vorkommen, wirkt sich dies nur einmal auf das Register AR1 aus, aber nicht auf das Register AR0.
Skizzieren Sie ein ASM-Diagramm für ein komplexes Moore-Schaltwerk, dass das Mastermind-Spiel mit möglichst geringer Anzahl an Taktzyklen ausführt.

- **Die Musterlösung:**



Zunächst werden in Zustand Z0 die Eingaberegister geladen und die Ausgaberegister mit 0 initialisiert. Danach werden als erstes die Farben auf der linken Position (also die Register ER0 und CR0) verglichen (Zustand Z1). Falls diese identisch sind, erhöht sich im Zustand Z2 das Register AR1 um eins. Nun muss nur noch überprüft werden, ob die Farben auf der rechten Position (also die Register ER1 und CR1) ebenfalls identisch sind, denn der Fall, dass eine Farbe zwar richtig ist, aber auf der falschen Position liegt, kann hier laut Aufgabenstellung nicht mehr eintreten.
Falls ja, führt dies zu einer weiteren Erhöhung des Ausgaberegisters AR1 (im Zustand Z3) und das Spiel ist beendet, da die Überprüfung im Zustand Z6 positiv ausfällt. Falls nein, wird die nächste Runde mit einer neuen Eingabe im Zustand Z0 gestartet. Sind die Farben auf der linken Position nicht identisch (Zustand Z1), wird zunächst überprüft, ob die Farben auf der rechten Position identisch sind. Wenn ja, wird das Ausgaberegister AR1 im Zustand Z3 um eins erhöht. In diesem Fall muss aufgrund der Aufgabenstellung auch nichts weiter unternommen werden und die nächste Runde mit einer neuen Eingabe kann gestartet werden.
Sind jedoch im Zustand Z1 die Farben auf den entsprechenden Positionen nicht identisch muss noch überprüft werden, ob die Farben zwar richtig, aber auf der jeweils falschen Position liegen. Falls dem so ist wird das Ausgaberegister im Zustand Z4 und ggf. in Zustand Z5 entsprechend erhöht und eine neue Runde gestartet, ansonsten wird direkt eine neue Runde gestartet. Somit kann eine Farbe, egal auf welcher bzw. auf welchen Positionen sie sich befindet immer nur höchstens ein Ausgaberegister AR1 oder AR0 und auch nur in dieser Reihenfolge verändern.

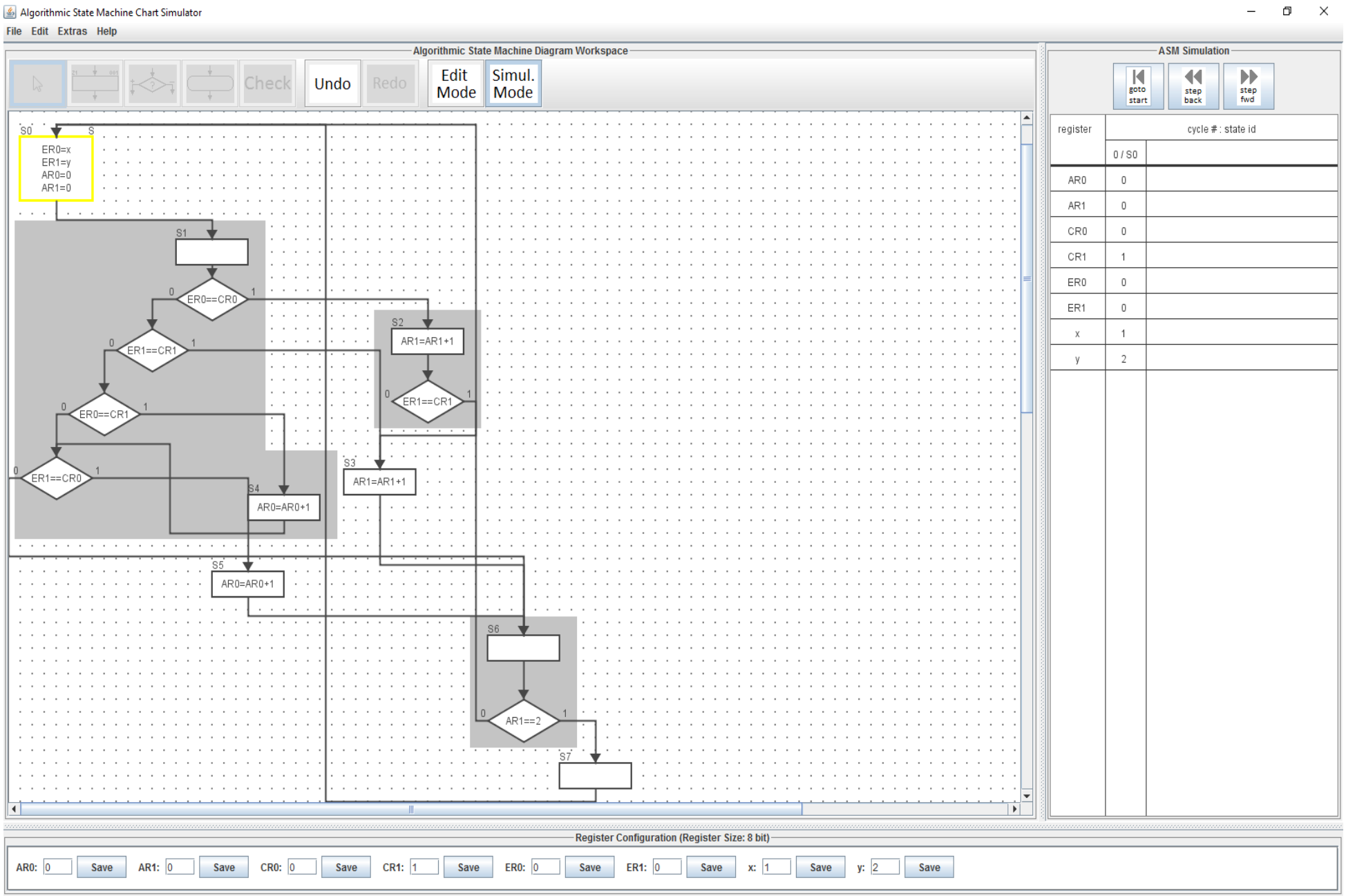
Quelle: Computersysteme I (2016), Einsendeaufgabe 4, Aufgabe 6

- **Die ASM Simulation:**

Der als letzte Seite angehängte Screenshot zeigt die Umsetzung des ASM-Diagramms aus der Musterlösung mit dem ASM-Simulator.
Im Startzustand werden die Ausgaberegister mit 0 initialisiert und die Benutzereingabe, also die „geratenen“ Farben, in die Eingaberegister geschrieben.
Abweichend von der Zeichnung des ASM-Diagramms der Musterlösung werden in der Simulation die Werte nicht direkt, sondern über die Variablen x und y in die Eingaberegister geschrieben, weil es für den Benutzer der Simulation einfacher ist, die Variablen mit gewünschten Werten zu belegen, als dies im Editiermodus direkt in der Zustandsbox über die RTL-Notation zu tun.
Initial wurden die Coderegister mit rot und blau belegt, also CR0 = 0 und CR1 = 1, die beiden Eingaberegister, die im Startzustand Z0 beschrieben werden, raten blau und grün, also x = 1 und y = 2. Klickt man nun rechts im Feld *ASM Simulation* mehrfach auf *step fwd*, kann man den Weg durch das Diagramm verfolgen und sehen, dass wie zu erwarten AR0 einmal inkrementiert wird und AR1 gar nicht, denn es wurde zwar eine Farbe richtig geraten, aber an der falschen Position.
Um nun die geheimen Werte in den Coderegistern und/oder die geratenen Werte in den Eingaberegistern zu ändern, füllt man die entsprechenden Felder neben den Registernamen im unteren Feld *Register Configuration* und bestätigt die Eingabe jeweils durch Klick auf *Save*.

- **Die Simulation besteht aus folgenden Komponenten:**

  - 8 Zustandsboxen
  - 6 Entscheidungsboxen


Algorithmic State Machine Chart Simulator
File Edit Extras Help
Algorithmic State Machine Diagram Workspace
Check
Undo
Redo
Edit Mode
Simul. Mode
S0
S
ER0=x
ER1=y
AR0=0
AR1=0
S1
ER0==CR0
ER1==CR1
ER0==CR1
ER1==CR0
S2
AR1=AR1+1
ER1==CR1
S3
AR1=AR1+1
S4
AR0=AR0+1
S5
AR0=AR0+1
S6
AR1==2
S7
ASM Simulation
goto start
step back
step fwd
register
cycle # : state id
0 / S0
AR0 0
AR1 0
CR0 0
CR1 1
ER0 0
ER1 0
x 1
y 2
Register Configuration (Register Size: 8 bit)
AR0: 0 Save
AR1: 0 Save
CR0: 0 Save
CR1: 1 Save
ER0: 0 Save
ER1: 0 Save
x: 1 Save
y: 2 Save