Rechnertechnik **Übungszettel 4** **SS2014**

In dieser Aufgabe sollen Sie einen Automaten entwerfen und diesen mit Hilfe eines programmierbaren Logikbausteins implementieren und testen. Hierzu verwenden Sie bitte ein sogenanntes GAL, das eine DNF direkt implementieren kann (siehe VL). Minimieren müssen Sie in dieser Aufgabe nicht, da dies durch die verwendeten Werkzeuge automatisch erfolgt. Ein Datenblatt des verwendeten GALs GAL16V8 finden Sie auf Moodle zur Orientierung.

Aufgabe 1: Automaten

Bauen Sie eine Schaltung, die einen Automaten für eine Ampelschaltung mit Fußgängerüberweg implementiert. Die Ampel soll dabei die Lichter der Farben grün, gelb und rot besitzen und damit den Kfz-Verkehr auf der Straße steuern (nehmen Sie einfach 3 LEDs des Koffers und merken sich ihre Bedeutung). Verwenden Sie einen Taster zum Schalten der Ampel in den Fußgänger-Frei-Modus. Die Schaltung soll das folgende Verhalten aufweisen:

- Zu Beginn steht die Ampel auf Grün. Jede Grün-Phase muss immer mindestens 3 Sekunden anhalten, erst dann darf auf umgeschaltet werden.
- Wird der Taster während der Grün-Phase gedrückt, soll die Ampel auf Gelb wechseln, danach auf Rot. Die Gelbphase soll 1 Sekunde anhalten. Drücken des Tasters während der Gelbphase hat keinen Einfluss auf die Schaltung.
- Ist die Ampel auf Rot, soll nach 2 Sekunden auf GelbRot gewechselt werden. Drücken des Tasters während der Rot-Phase hat keinen Einfluss auf die Schaltung.
- Ist die Ampel in der GelbRot-Phase, so soll nach 1 Sekunde in die Grün-Phase gewechselt werden. Der Taster zum Schalten der Ampel in die Rotphase ist während der GelbRot-Phase schon wieder aktiv. Beachten Sie aber, dass die Ampel zunächst 3 Sekunden grün sein muss, bevor sie wieder schalten darf.

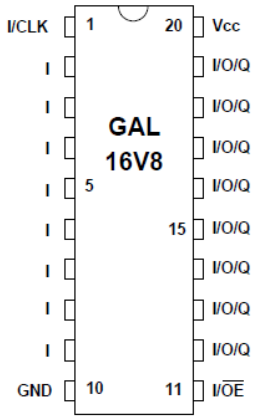
Führen Sie die folgenden Aufgaben durch:

a) Entscheiden Sie, ob Sie einen Moore- oder einen Mealy-Automaten implementieren wollen und begründen Sie Ihre Entscheidung.
b) Entwerfen Sie einen Automaten, der das erforderliche Verhalten implementiert und zeichnen Sie den Zustandsgraph.
c) Überlegen Sie sich, wann Sie welche Ausgaben wie ansteuern müssen.
d) Erstellen Sie eine Kodierung der Zustände.
e) Erstellen Sie die Zustandsübergangstabelle für den Automaten und die Wahrheitswertetabellen für die Ausgaben. Beachten Sie bitte insbesondere den Startzustand und wie Sie diesen aktivieren. Bedenken Sie hierbei, dass GALs nicht über einen asynchronen Reset auf den FlipFlops verfügen.
f) Implementieren Sie die Schaltung in Form eines Gleichungssystem für jedes Zustandsbit und jeden Ausgang unter WinCUPL (siehe Anhang I)
g) Simulieren Sie Ihr Gleichungssystem unter WinCUPL (siehe Anhang II).
h) Nach erfolgreicher Simulation erzeugen Sie bitte ein JEDEC-File und brennen Sie mittels des Programmes G540 ein GAL (siehe Anhang III).
i) Bauen Sie die Schaltung auf, indem Sie eine der kleinen Zusatzplatinen mit Sockel für die GALs nehmen, das GAL dort einsetzen, das Board mit der Spannungsversorgung eines Digitalkastens verdrahten und Eingänge und Ausgänge (LEDs, Tastern und Taktgenerator) so mit Jumpern stecken, dass die Mittel der Zusatzplatine genutzt werden. Als Takt für die Register verwendet das Zusatzboard einen 1Hz-Rechteckgenerator. Testen Sie nun Ihre Ampelschaltung.

Anhang I

Implementierung mittels WinCUPL

Erstellen Sie sich ein neues Arbeitsverzeichnis und kopieren Sie die Datei „*Ampel.pld*“ aus Moodle in dieses Verzeichnis. „*Ampel.pld*“ enthält wesentliche Voreinstellungen (allgemeine Informationen und im Besonderen den Baustein, also hier *G16V8* für das entsprechende GAL16V8), die Sie für dieses Projekt brauchen und muss von Ihnen nun erweitert werden. Starten Sie nun das Programm ***WinCUPL*** und öffnen Sie die PLD-Datei, die Sie eben angelegt haben. Im Editor kann diese Datei nun erweitert werden. Ändern Sie bitte nichts am Header. Nach dem Header finden Sie die Pin-Belegungen des Bausteins. Hier sind schon ein paar Einträge vorbereitet, so dass Sie diese übernehmen können. Zum Vergleich schauen Sie bitte auch in das Datenblatt des Bausteins (auf Moodle zu finden) und dort insbesondere auf die Pinbelegung (siehe auch nebenstehende Grafik). Wie Sie sehen, wurde der Eingang für den Taster auf Pin 2 und der für den Reset auf Pin 3 gelegt. Die Ausgänge zu den Leuchten wurden auf 12, 13 und 14 gelegt. An diesen Einstellungen bitte nichts ändern, da dies so auf der kleinen Zusatzplatine inkl. Leuchtmittel bereits aufgebaut ist.



Nun kann hier der Ampelautomat implementiert werden. Hierzu benötigen Sie die Kodierung der Zustände, insbesondere die Anzahl der Bits der Zustände. Ein GAL enthält Flipflops als Speicher, die aber den Ausgängen zugeordnet sind. Das bedeutet, dass Sie zusätzlich zu den bereits festgelegten Pins weitere nutzen müssen, die das jeweilige Bit des Zustands ausgeben (auch wenn Sie diese Bits nicht nutzen). Seien die Zustandsbits mit q0 bis q2 benannt. Dann müssen Sie diese auch in der „*Ampel.pld*“ als Pins definieren, beispielsweise so:

```
Pin 15 = q0;
Pin 16 = q1;
Pin 17 = q2;
```

Achten Sie darauf, dass die Pins I/O-Pins (siehe Abbildung oben) sein müssen, da Sie den Wert des FlipFlops ja auch wieder lesen möchten.

Erstellen Sie nun die DNF für jeden Zustandsbit und jeden Ausgang aus Ihrer Wertetabelle. Diese Gleichungen können nun in „Ampel.pld“ eingegeben werden, wobei die folgenden Sprachregeln gelten:

- Eine Zuweisung auf einen Pin erfolgt mit dem Operator „=“
- Eine Invertierung eines Signals erfolgt mit dem Operator „!“
- Eine Und-Verknüpfung erfolgt mit dem Operator „&“
- Eine Oder-Verknüpfung erfolgt mit dem Operator „#“

- Output-Pins, die kein FlipFlop beinhalten, werden mit ihrem Namen angesprochen
- Input-Pins werden mit ihrem Namen angesprochen
- FlipFlop-Inhalte werden bei Veränderung (linke Seite einer Zuweisung mit „=“) mit ihrem Namen gefolgt von der Zeichenkette „.d“ angesprochen.

Beispiel: Sie haben die folgenden beiden DNFs erstellt:

$q0 = a1 \wedge \neg a2 \vee q0 \wedge a3$

$x = a1 \wedge \neg q0$

wobei q0 ein Zustandsbit sein soll und damit in einem FlipFlop gespeichert wird, wohingegen X ein normaler Output ist. Dann wird dies unter WinCUPL zu:

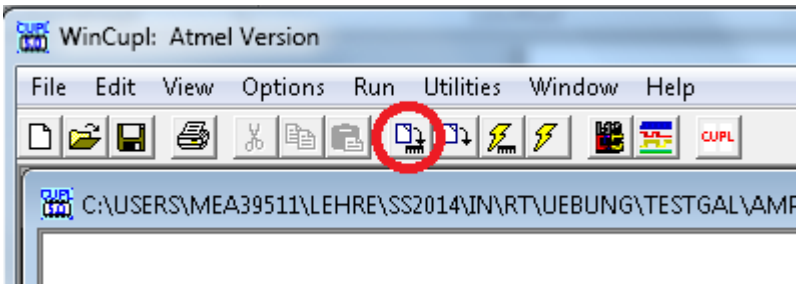
```
q0.d = a1 & !a2 # q0 & a3
x = a1 & q0
```

x und q0 müssen natürlich vorher einem Pin zugeordnet sein (s.o.).

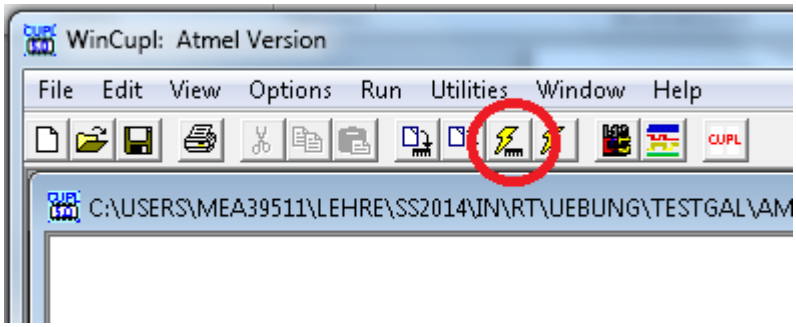
Anhang II

Simulation mittels WinCUPL

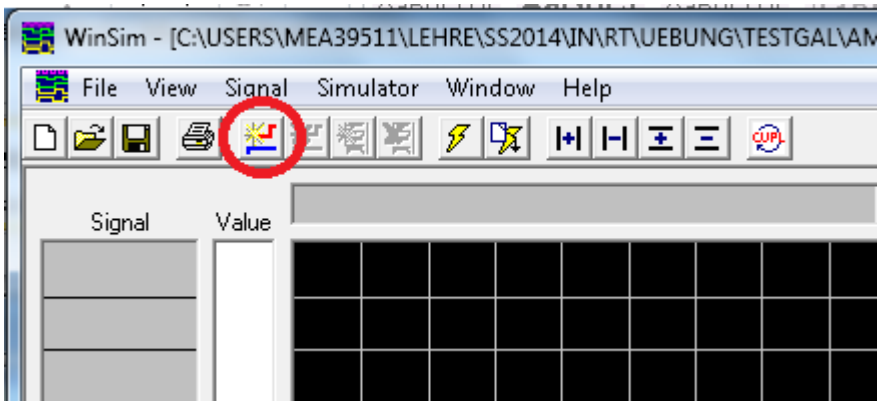
Nun können Sie Ihren Entwurf simulieren. Dazu müssen Sie zunächst einmal übersetzen:



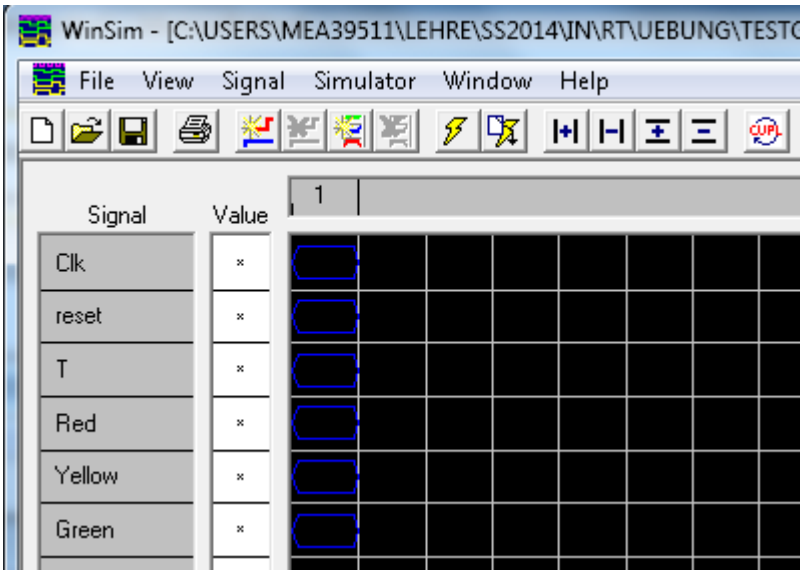
Anschließend rufen Sie den Simulator auf:



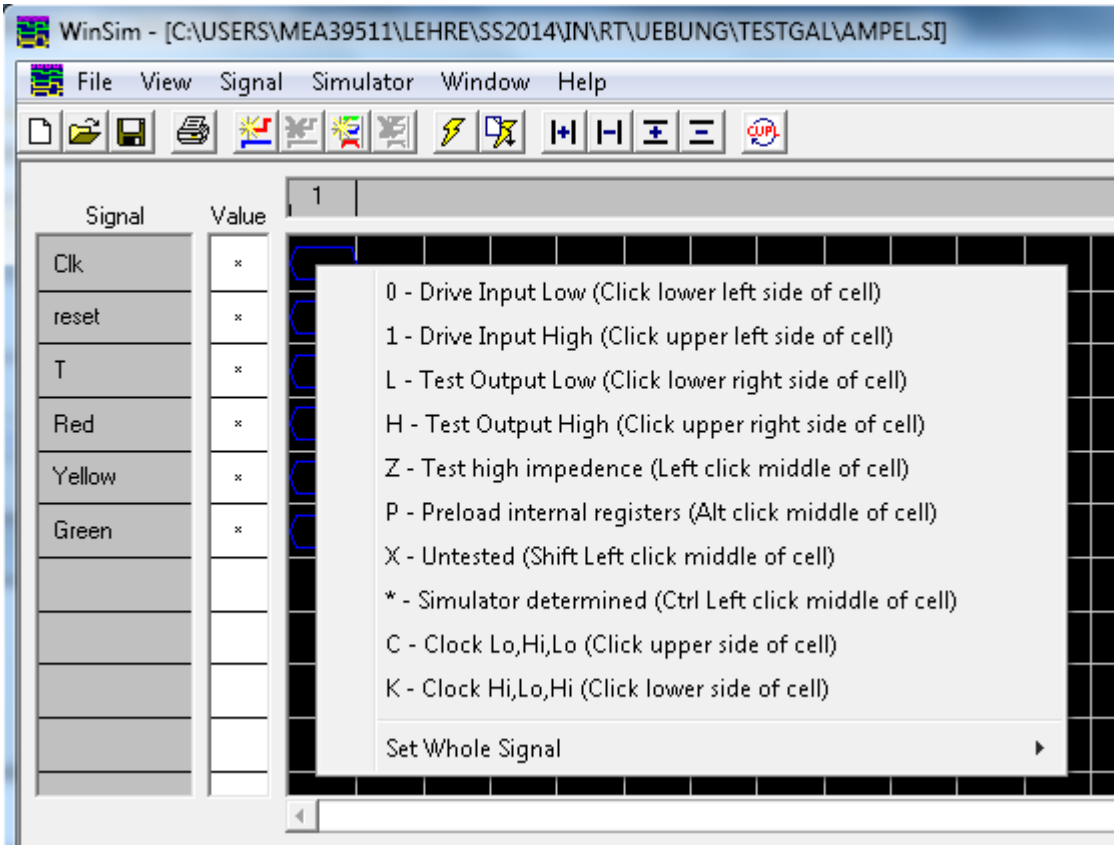
Sie bekommen eine Fehlermeldung, die Sie aber ignorieren können und erhalten dann ein leeres Simulationsfenster. Hier fügen Sie nun die zu beobachtenden Signale einfügen:



Wählen Sie nun jeweils ein Signal aus und bestätigen Sie mit „OK“. Wenn Sie alle Signale, die Sie beobachten wollen ausgewählt haben, beenden Sie den Dialog mit „Done“. Anschließend sollte Ihr Simulatorfenster etwa so aussehen:

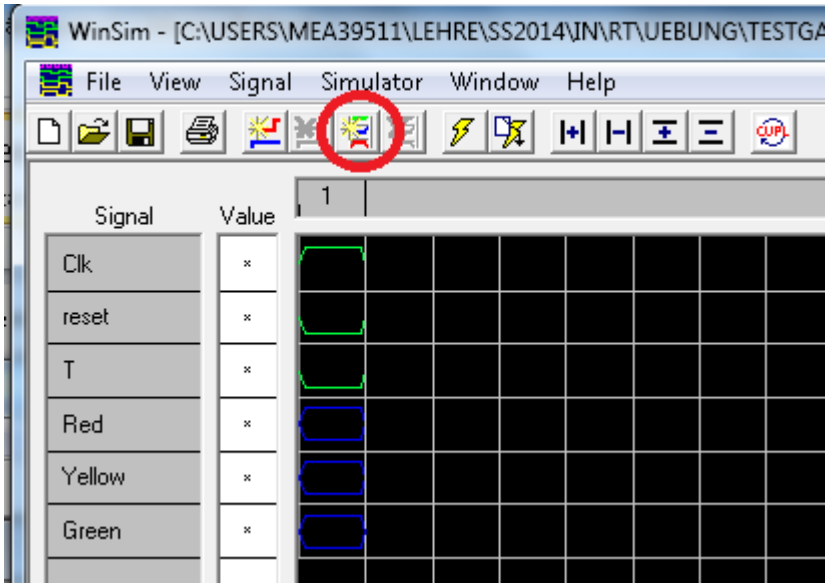


Nun müssen den Eingaben Werte zugeordnet werden. Die erreichen Sie, indem Sie für ein Signal in der Signaldarstellung den rechten Mausknopf drücken. Sie erhalten dann ein Auswahlmenü, in dem Sie den gewünschten Wert einstellen können:

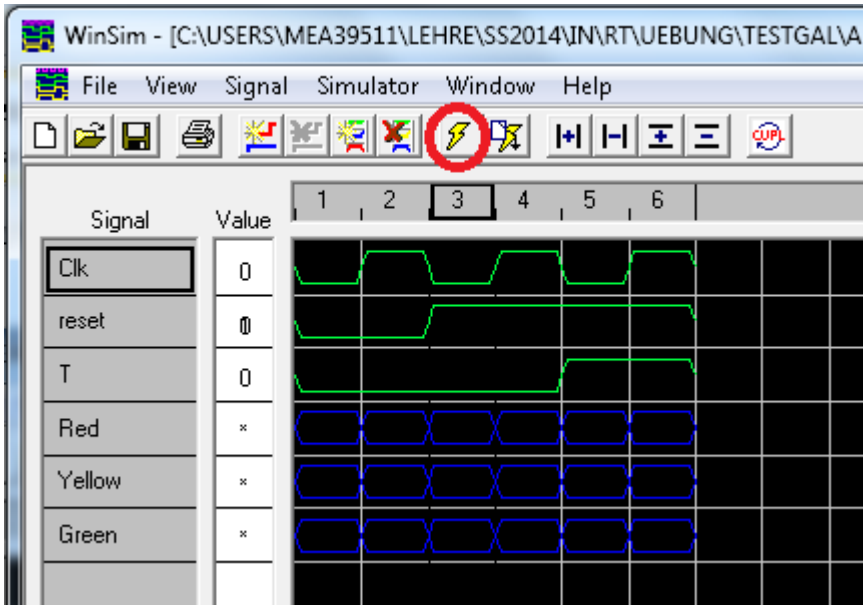


Die hier interessanten Werte sind 0 und 1 für Eingänge und * für Ausgänge.

Belegen Sie nun alle Eingänge mit den gewünschten Werten und alle Ausgänge mit *. Damit ist ein Simulationsschritt definiert. Zur Definition weiterer Schritte müssen Sie einen sogenannten Vektor hinzufügen:



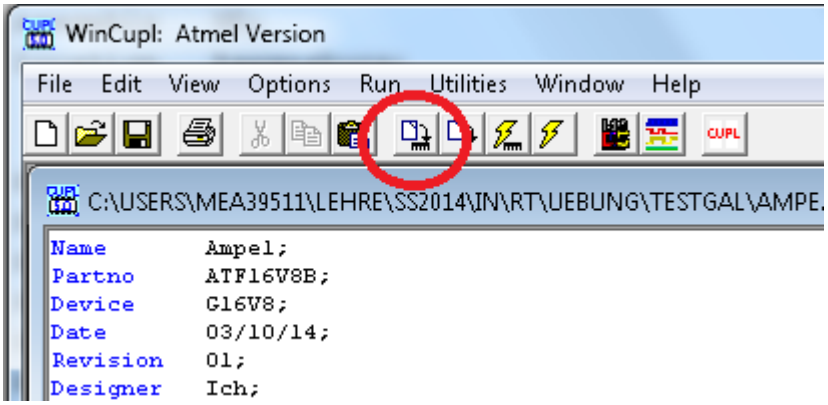
Geben Sie die für Ihre Simulation notwendige Menge an Schritten ein und bestätigen Sie. Sie werden nun im Simulationsfester weitere Schritte sehen, für die Sie die Eingangssignale bestimmen und eingeben müssen. Speichern Sie anschließend diese Eingabe (Sie können Sie dann nach Änderungen am Design wiederverwenden). Anschließend wird die Simulation aufgerufen:



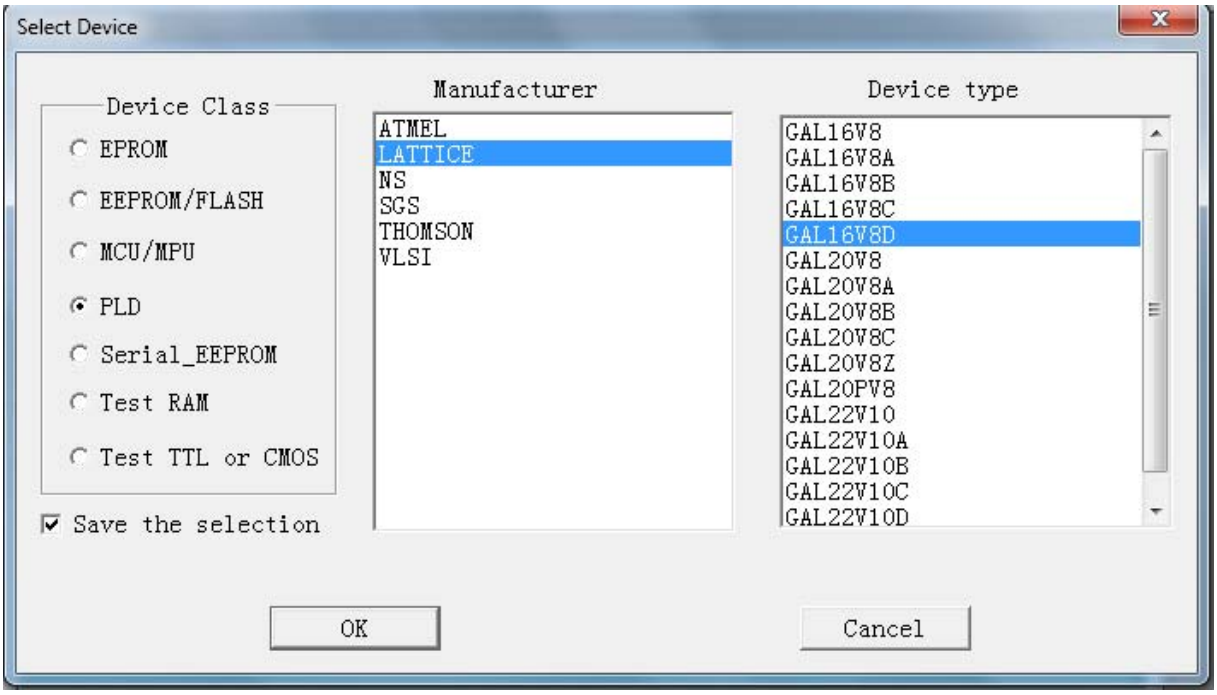
Nach Bestätigung und Ignorieren etwaiger Warnungen können Sie nun an den veränderten Outputs sehen, was Ihr Design für das angegebene Testmuster tut. Wem die Eingabe des Testmusters über die Oberfläche zu umständlich ist, der kann nach erstmaliger Erstellung auch das .si-File in einem Texteditor bearbeiten.

Anhang III

Bevor das GAL beschrieben werden kann, muss zunächst ein sogenanntes JEDEC-File erzeugt werden. Dies wird beim Übersetzen in WinCUPL generiert:

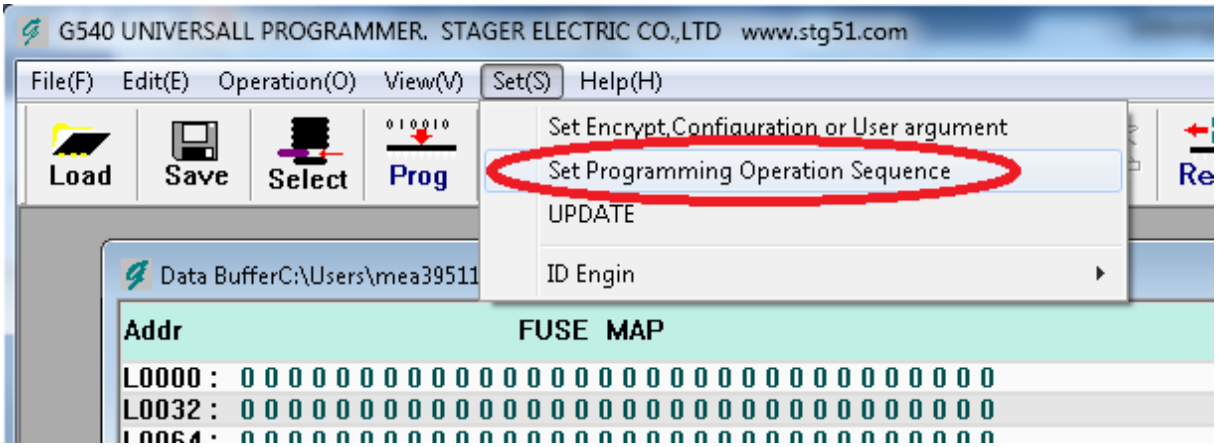


Schließen Sie nun den GAL-Brenner an Ihren PC an und starten Sie das Programm **G540**. Schließen Sie ggf. alle Anzeigen im Hauptfenster des Programms. Öffnen Sie nun in G540 über den Menüpunkt File→Load File die Datei „Ampel.jed“ aus Ihrem Arbeitsverzeichnis. Drücken Sie anschließend den Select-Button und stellen Sie das richtige Device ein, wie in der folgenden Abbildung zu sehen:

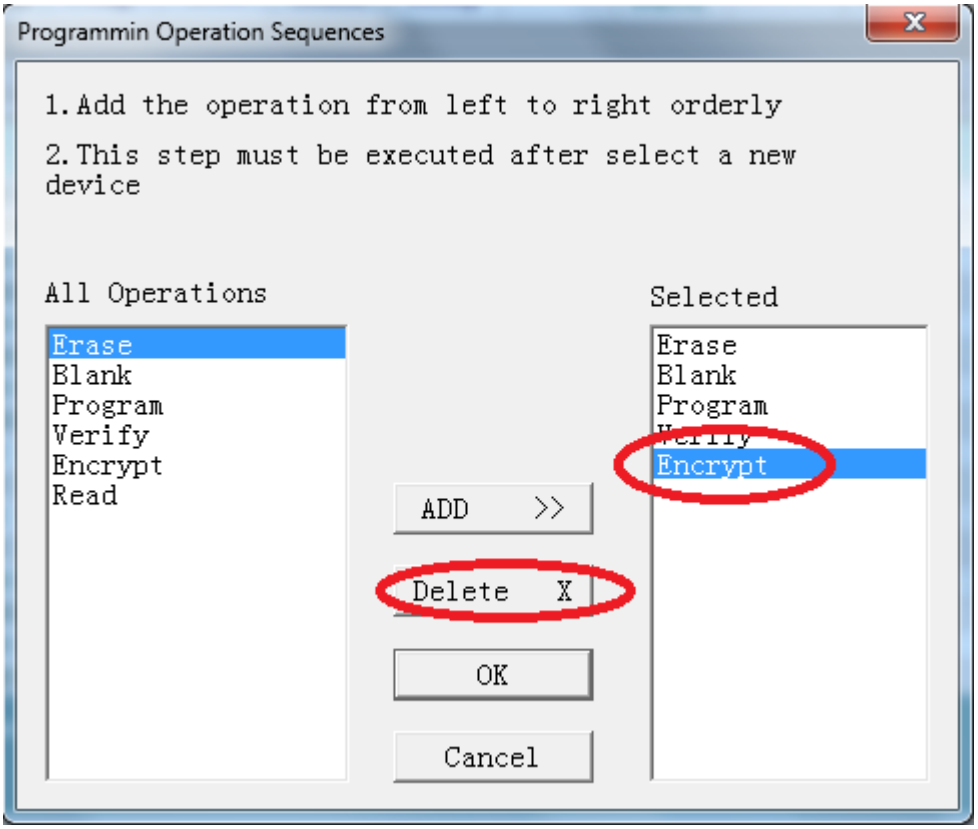


Nehmen Sie nun einen GAL16V8-Baustein und setzen ihn wie angegeben in das Programmiergerät ein (Verriegelung nicht vergessen, im Zweifel den Tutor fragen!). Der Informationsdialog kann anschließend geschlossen werden.

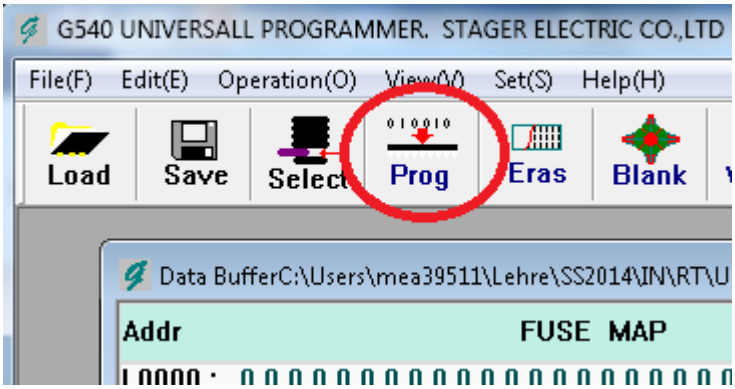
Rufen Sie nun den folgenden Menüpunkt auf:



Im daraufhin aufgehenden Dialog löschen Sie bitte die letzte Aktion („Encrypt“) auf der rechten Seite:



Drücken Sie anschließend den Button „Program“ und warten Sie, bis alle Operationen durchgelaufen sind (die letzte ist „Verify“).



Nun ist der Baustein programmiert und Sie können Ihn aus dem Brenner nehmen und in Ihre Schaltung einbauen.