Zustandsautomaten mit OHE-Zustandscodierung

11. 1. 2013

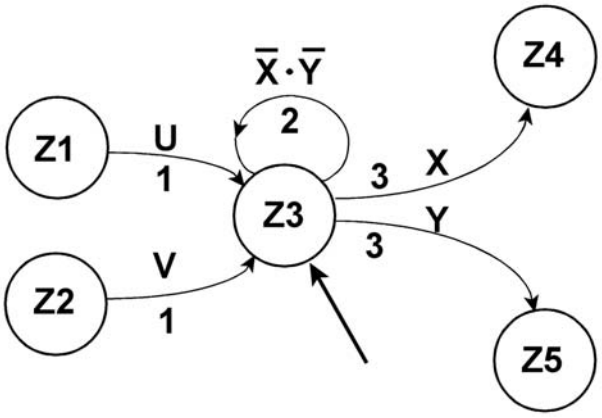
OHE = One Hot Enable = 1-aus-n-Codierung. Je Zustand ein Flipflop. Die Vorteile:

- einfaches Entwerfen,
- geringer Aufwand in der Kombinatorik,
- das Prinzip kommt den heutigen FPGA-Architekturen entgegen. Jjede Zelle hat eine kleine Lookup-Tabelle für die Kombinatorik und ein Flipflop. Alle Flipflops sind bezahlt. Es bringt aso nichts, um jeden Preis welche einzusparen.

Praktische Durchführbarkeit: bis zu einigen zehn Zuständen.

Die Grundlagen

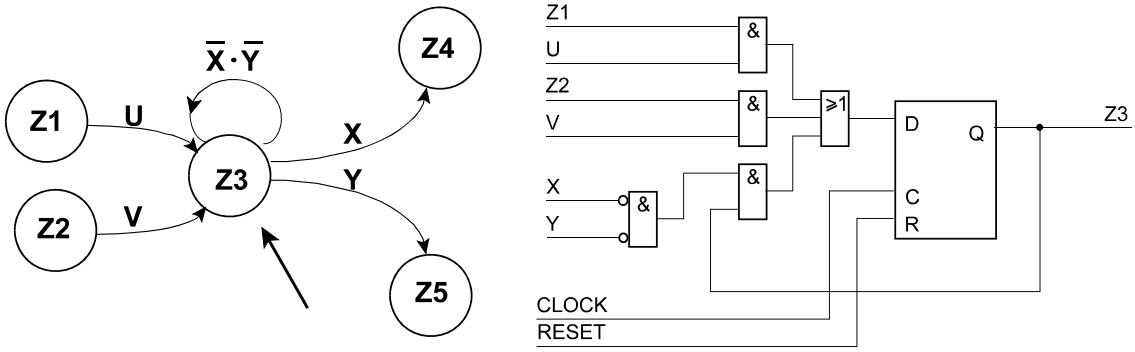
Wir betrachten zunächst einen einzelnen Zustand (im Beispiel: Z3). Er kann aus anderen Zuständen erreicht (eingeleitet) und in Richtung anderer Zustände verlassen werden. Ist er aktiv und sind keine Übergangsbedingungen in andere Zustände erfüllt, so wird er gehalten.



Die Bedingungen des Einleitens und Verlassens sind hier jeweils einzelne Signale bzw. Boolesche Variable. Eine Einleitungsbedingung ist eine Übergangbedingung zum Verlassen des jeweiligen vorhergehenden Zustandes; eine Übergangsbedingung ist eine Einleitungsbedingung für den jeweiligen nachfolgenden Zustand. Signalkombinationen müssen ggf. durch Decodierung in Einzelsignale umgesetzt werden.

Der Anfangszustand ist nach dem Einschalten hart zu erzwingen. Hierzu das Flipflop des Anfangszustandes setzen, alle anderen zurücksetzen (asynchron).

Implementierung des Zustands mittels D-Flipflop:



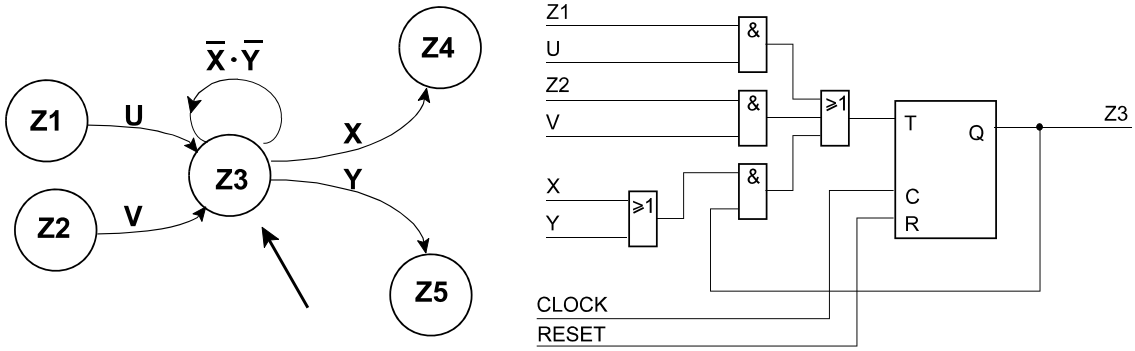
Das Flipflop ist zu setzen, wenn eine Einleitungsbedingung erfüllt und der jeweils zugehörige Zustand aktiv ist.

Beispiel: $Z1 \cdot U \vee Z2 \cdot V$

Das Flipflop ist aktiv zu halten (Rückführung), wenn keine der Übergangsbedingungen erfüllt ist:

Beispiel: $Z3 \cdot \overline{X} \cdot \overline{Y}$

Implementierung des Zustands mittels T-Flipflop:



Eine Änderung ist zu veranlassen, wenn der Zustand eingeleitet oder verlassen wird.

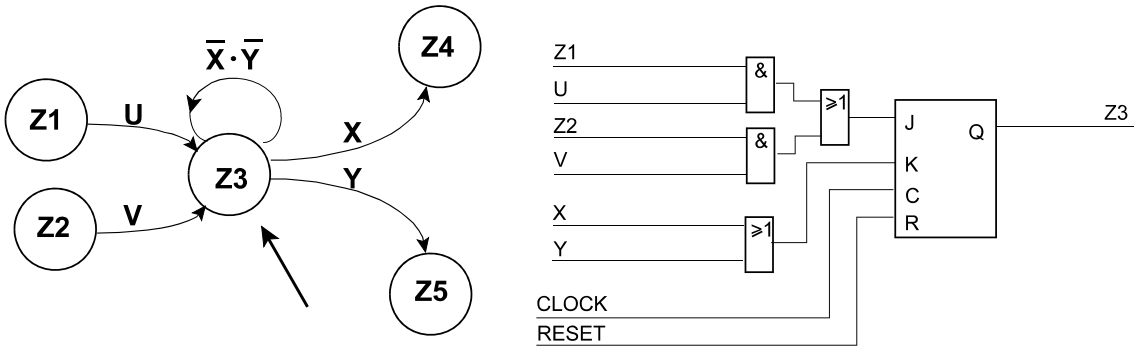
Um das Flipflop zu setzen, ist eine Änderung zu veranlassen, wenn eine Einleitungsbedingung erfüllt und der jeweils zugehörige Zustand aktiv ist.

Beispiel: $Z1 \cdot U \vee Z2 \cdot V$

Um das Flipflop zurückzusetzen, ist eine Änderung zu veranlassen, wenn der Zustand aktiv und eine der Übergangsbedingungen erfüllt ist.

Beispiel: $Z3 \cdot (X \vee Y)$

Implementierung des Zustands mittels RS-Flipflop:
(Um Verwechslungen mit dem asynchronen Setzen und Rücksetzen zu vermeiden, stellen wir hier ein JK-Flipflop dar; S = J und R = K.)



Das Flipflop ist zu setzen, wenn eine Einleitungsbedingung erfüllt und der jeweils zugehörige Zustand aktiv ist.

Beispiel: $J = Z1 \cdot U \vee Z2 \cdot V$

Das Flipflop ist zurückzusetzen, wenn eine der Übergangsbedingungen erfüllt ist.

Beispiel: $K = X \vee Y$

Denksportaufgabe: Weshalb ist beim Verlassen eine UND-Verknüpfung mit dem aktuellen Zustand unnötig?

Die UND-Verknüpfung (vgl. T-Flipflop) soll verhindern, daß das Flipflop schaltet, wenn die Übergangsbedingungen aktiv werden, während sich die State Machine in anderen Zuständen befindet. In einem solchen Fall ist das hier betrachtete Flipflop ohnehin inaktiv, so daß ein Rücksetzen nicht schadet. Tritt eine Übergangsbedingung zugleich mit einer Einleitungsbedingung auf, die das Flipflop setzen will, so schadet sie auch nicht, da ein inaktives JK-Flipflop auch mit J = K =1 aktiv wird. *Aber Achtung:* das gilt nur bei JK, nicht bei echten RS-Flopflops. Hier ist R = S = 1 verboten. Dann müßte also tatsächlich K = X v Y ersetzt werden durch R = Z3 (X v Y).

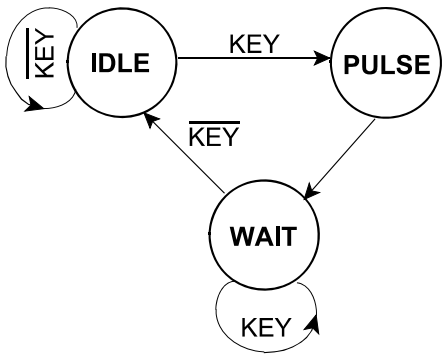
Das Verknüpfen mit dem aktuellen Zustand schadet aber auch nicht. Wenn es sich um ein echtes RS-Flipflop handelt (wo S = R = 1 verboten ist), ist es ohnehin erforderlich. Im allgemeinen Fall ist das Flipflop dann zurückzusetzen, wenn der Zustand aktiv und eine der Übergangsbedingungen zum Verlassen des Zustandes erfüllt ist.

Im Beispiel: $K = Z3 \cdot X \vee Z3 \cdot Y$

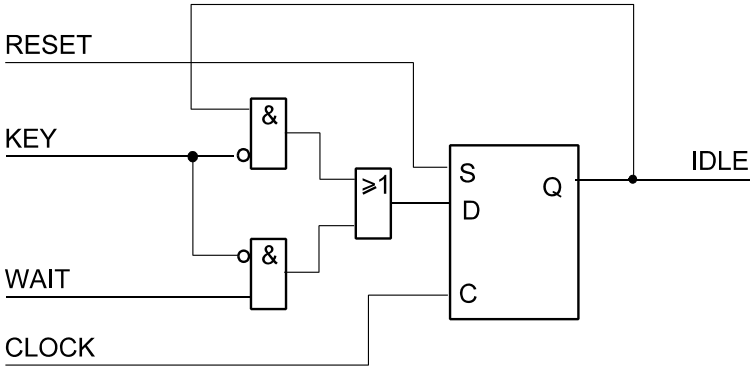
Für T- und RS- bzw. JK-Flopflops gilt allgemein:
Die Übergangsbedingung eines jeweils vorhergehenden Zustandes ist zugleich eine Setzbedingung des jeweils nachfolgenden Zustandes; die UND-Verknüpfungen von Zustand und Eingangssignalen (Übergangsbedingungen) dienen gleichzeitig zum Verlassen (Ausschalten) des aktuellen und zum Setzen des nachfolgenden Zustandes.

Beispiel: der Single-Shot-Generator

Dieser Zustandsautomat soll auf eine Tastenbetätigung (KEY) hin einen einzigen Impuls (PULSE) abgeben. Das Problem: Die Tastenbetätigung dauert viel länger als eine einzelne Taktperiode. Deshalb ist ein Wartezustand (WAIT) einzuführen, in dem auf das Loslassen der Taste gewartet wird.



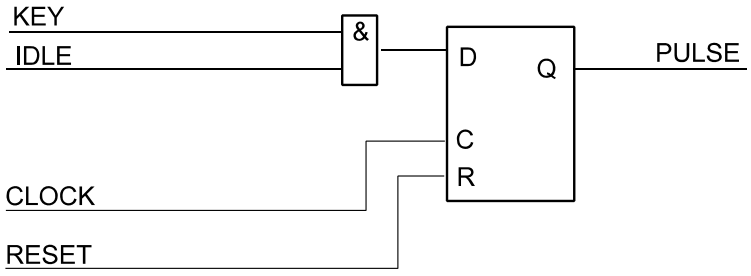
Implementierung mit D-Flipflops:



Der IDLE-Zustand wird eingeleitet, wenn im WAIT-Zustand die Taste losgelassen wird.

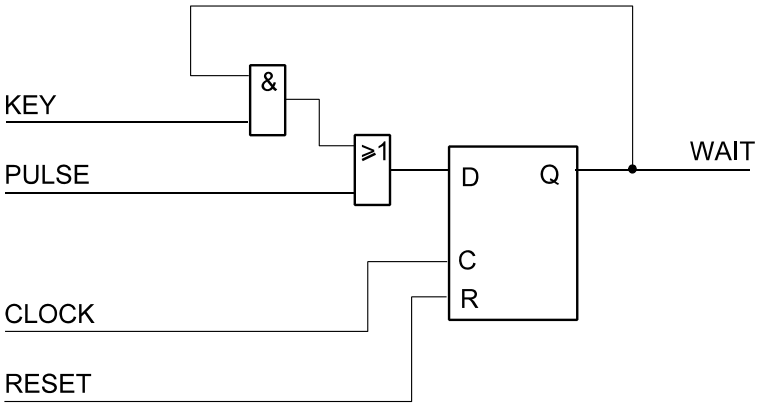
Er wird gehalten, solange die Taste nicht betätigt (= losgelassen) ist. Bei betätigter Taste wird er verlassen.

$$\mathrm{IDLE}_D = \mathrm{WAIT} \cdot \overline{\mathrm{KEY}} \vee \mathrm{IDLE} \cdot \overline{\mathrm{KEY}}$$



Der PULSE-Zustand wird eingeleitet, wenn im IDLE-Zustand die Taste betätigt wird. Er wird mit dem nächsten Takt wieder verlassen.

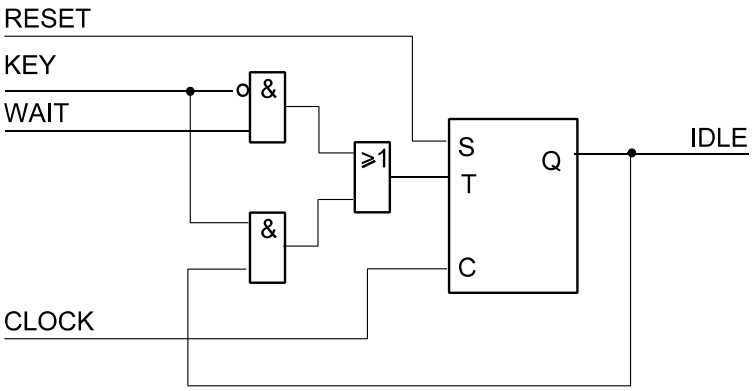
$$PULSE_D = IDLE \cdot KEY$$



Der WAIT-Zustand wird eingeleitet, wenn der PULSE-Zustand aktiv ist. Er wird gehalten, solange die Taste betätigt ist. Bei losgelassener Taste wird er verlassen.

$$WAIT_D = PULSE \vee WAIT \cdot KEY$$

Implementierung mit T-Flipflops:

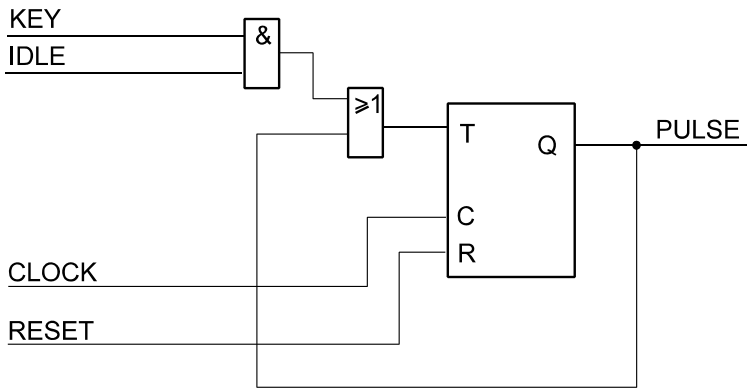


Der IDLE-Zustand wird eingeleitet, wenn im WAIT-Zustand die Taste losgelassen wird.

Er wird verlassen, wenn er aktiv ist und die Taste betätigt wird.

In beiden Fällen ist eine Änderung (T-Bedingung) zu veranlassen.

$$IDLE_T = WAIT \cdot \overline{KEY} \vee IDLE \cdot KEY$$

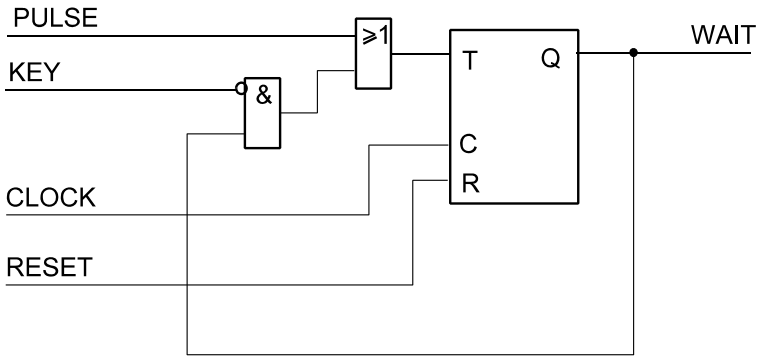


Der PULSE-Zustand wird eingeleitet, wenn im IDLE-Zustand die Taste betätigt wird.

Er wird verlassen, wenn er aktiv ist (= mit dem nächsten Takt).

In beiden Fällen ist eine Änderung (T-Bedingung) zu veranlassen.

$$PULSE_T = IDLE \cdot KEY \vee PULSE$$



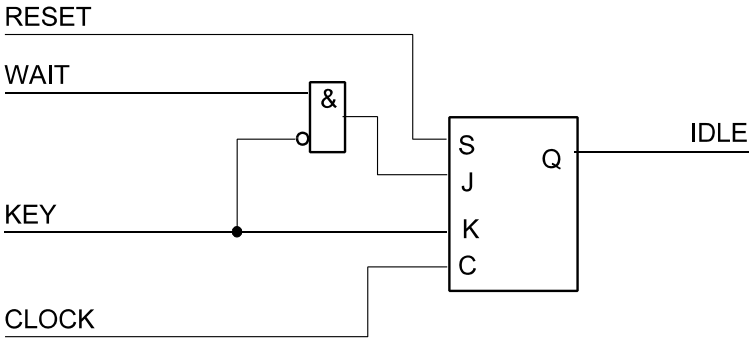
Der WAIT-Zustand wird eingeleitet, wenn der PULSE-Zustand aktiv ist.

Er wird verlassen, wenn die Taste nicht mehr betätigt (= losgelassen) ist.

In beiden Fällen ist eine Änderung (T-Bedingung) zu veranlassen.

$$WAIT_T = PULSE \vee WAIT \cdot \overline{KEY}$$

Implementierung mit JK-Flipflops (RS-Funktion):

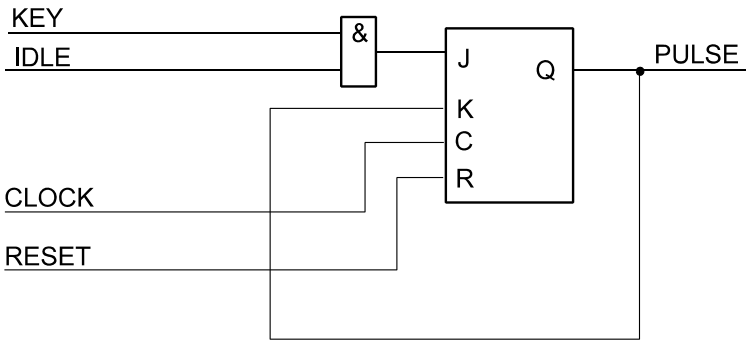


Der IDLE-Zustand wird eingeleitet, wenn im WAIT-Zustand die Taste losgelassen wird.

Er wird verlassen, wenn die Taste betätigt wird.

$$IDLE_J = WAIT \cdot \overline{KEY}$$

$$IDLE_K = KEY$$

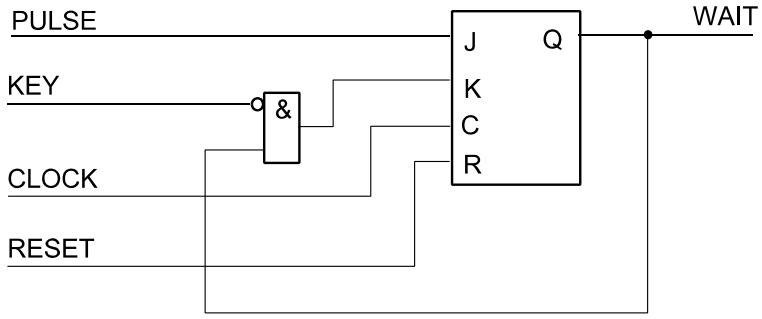


Der PULSE-Zustand wird eingeleitet, wenn im IDLE-Zustand die Taste betätigt wird.

Er wird verlassen, wenn er aktiv ist (= mit dem nächsten Takt).

$$PULSE_J = IDLE \cdot KEY$$

$$PULSE_K = PULSE$$



Der WAIT-Zustand wird eingeleitet, wenn der PULSE-Zustand aktiv ist.

Er wird verlassen, wenn die Taste nicht mehr betätigt (= losgelassen) ist.

$$WAIT_J = PULSE$$

$$WAIT_K = WAIT \cdot \overline{KEY}$$

Aufwandsvergleich durch Auszählen der Gatter:

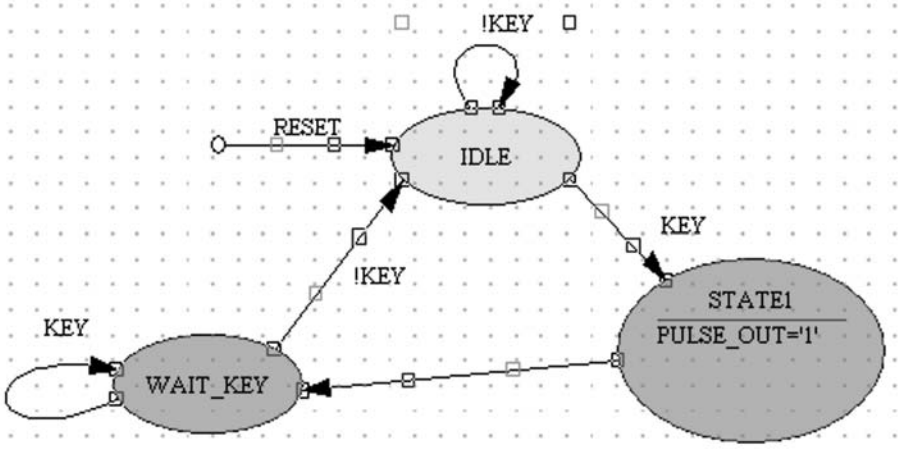
- mit D-Flipflops: 6 Gatter,
- mit T-Flipflops: 7 Gatter,
- mit JK-Flipflops: 3 Gatter.

Ersichtlicherweise sind bei den T- und JK-Flipflops die UND-Verknüpfungen der Übergangsbedingungen des jeweils vorhergehenden Zustandes gleich denen der Einleitungsbedingungen des jeweiligen Folgezustandes. Sie müssen also in der Schaltung nur einmal vorgesehen werden (Aufwandsersparnis bei den T-Flipflops 2 Gatter, bei den JK-Flipflops 1 Gatter). Andererseits sind die T- und JK-Flipflops intern aufwendiger als das D-Flipflop (sie bestehen heutzutage meist aus D-Flipflops mit vorgeschalteten Rückführungsnetzwerken).

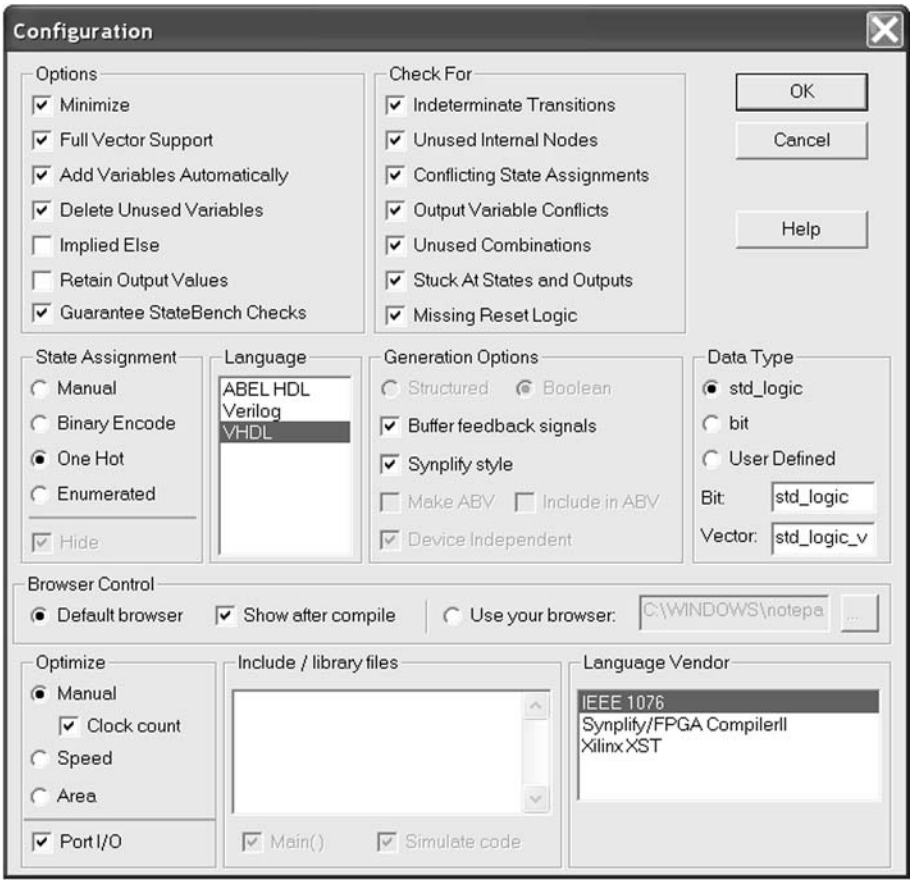
Hinweis:
Die hier dargestellten State Machines setzen vollsynchrone Signale voraus (alle Signale auf den gemeinsamen Takt bezogen). Ggf. sind Schaltmittel zur Eintaktierung (Synchronisation), Entprellung usw. vorzuordnen.

Single Shot Generator mit StateCAD

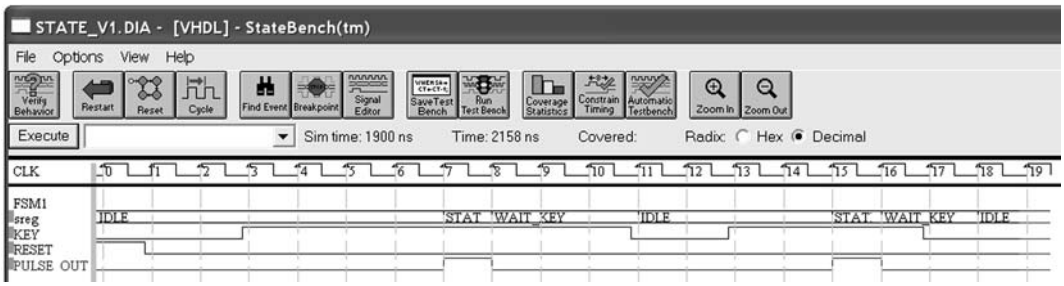
Eingabe des Zustandsdiagramms:



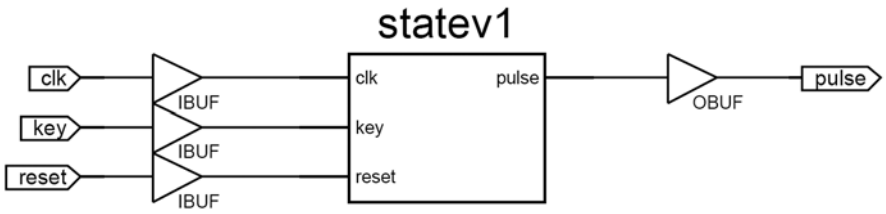
Konfigurationsmenü. Es wurde One Hot Encoding ausgewählt.



Simulation (StateBench):



Das Schaltsymbol:



Die Schaltgleichungen (vgl. weiter oben):

```
FDCPE_IDLE: FDCPE port map (IDLE,IDLE_D,CLK,'0',RESET);
    IDLE_D <= ((NOT KEY AND IDLE.LFBK)
     OR (NOT KEY AND WAIT_KEY.LFBK));
FDCPE_PULSE_OUT: FDCPE port map (PULSE_OUT,PULSE_OUT_D,CLK,RESET,'0');
    PULSE_OUT_D <= (KEY AND IDLE.LFBK);
FDCPE_WAIT_KEY: FDCPE port map (WAIT_KEY,WAIT_KEY_D,CLK,RESET,'0');
    WAIT_KEY_D <= ((STATE1.LFBK)
     OR (KEY AND WAIT_KEY.LFBK));
```