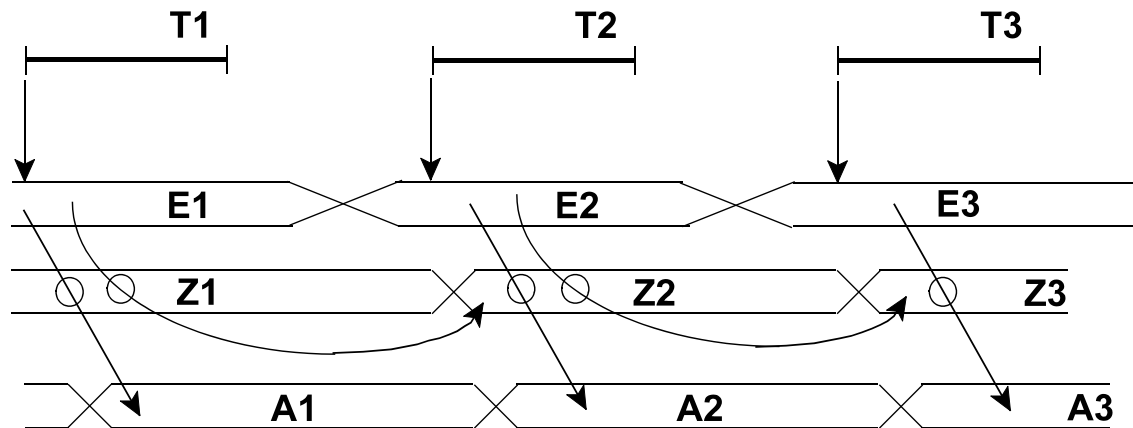


State Machines

Ergänzende Übersicht

12. 1. 09

Der Mealy-Automat



Die Ausgabe wird an der auswärtsführenden Kante des Zustandsgraphen vermerkt.

Zustand $Z(n)$ und Eingabe $E(n)$ ergeben Ausgabe $A(n)$ und Zustand $Z(n+1)$.

T1:

- $Z1$ ist bekannt (Anfangszustand). $E1$ wird abgetastet.
- Aus $Z1$ und dem abgetasteten $E1$ ergibt sich sofort $A1$ (im Idealfall unendlich schnell).
- Aus $Z1$ und dem abgetasteten $E1$ wird $Z2$ berechnet, der zu $T2$ wirksam wird.
- Während der Taktperiode $T1$ hat die Umwelt die Ausgangsbelegung $A1$ zu sehen bekommen. So kann sich vor $T2$ eine neue Eingangsbelegung $E2$ einstellen.
- Die in der nächsten Taktperiode abgetastete Eingangsbelegung $E(n+1)$ ergibt sich unter Einfluß der aktuellen Ausgangsbelegung $A(n)$.

T2:

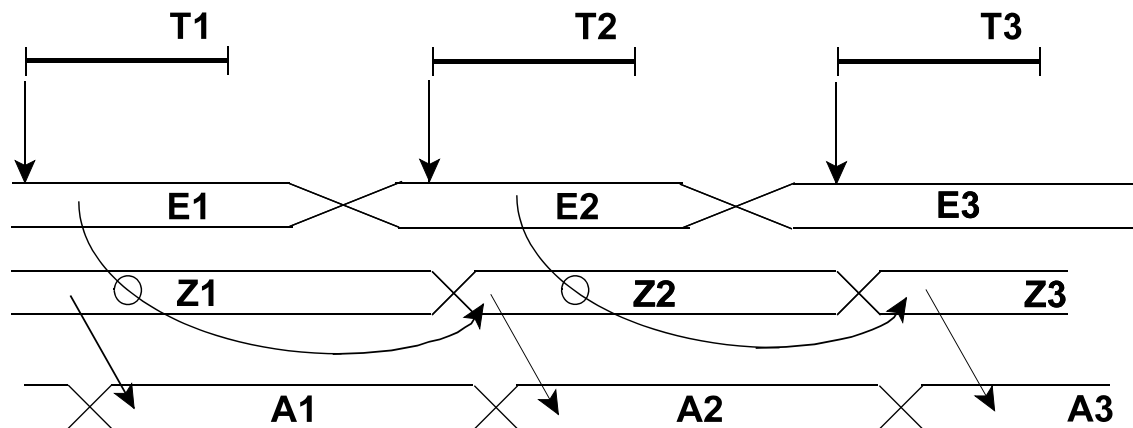
- Der zu $T1$ berechnete Zustand $Z2$ wird wirksam. $E2$ wird abgetastet.
- Aus $Z2$ und dem abgetasteten $E2$ ergibt sich sofort $A2$ (im Idealfall unendlich schnell).
- Aus $Z2$ und dem abgetasteten $E2$ wird $Z3$ berechnet, der zu $T3$ wirksam wird.

Die im Takt n abgetastete Eingangsbelegung $E(n)$ kann noch in der gleichen Taktperiode die Ausgangsbelegung $A(n)$ beeinflussen.

Im Sinne der Automatentheorie wird der Umwelt in jedem Takt nur einmal Information entnommen.

Implementierung: Mit jedem Takt den neuen Zustand einstellen und die Eingangsbelegung abtasten. Dann sofort Ausgangsbelegung bestimmen. Die sinnvolle Mindestdauer der Taktperiode wird davon bestimmt, daß die Umwelt in der Lage sein muß, auf die Ausgabe mit einer Änderung der Eingangsbelegung zu reagieren.

Der Moore-Automat



Die Ausgabe wird im Knoten des Zustandsgraphen vermerkt.

Zustand $Z(n)$ ergibt Ausgabe $A(n)$ (nur Umcodierung).

T1:

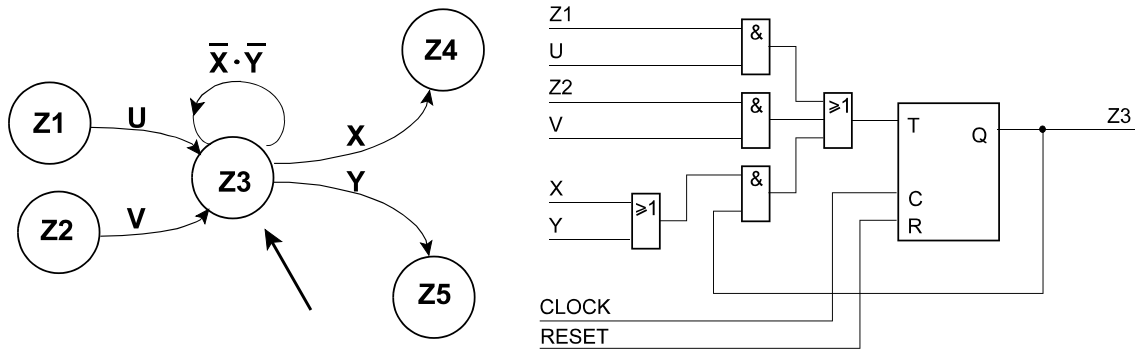
- $Z1$ ist bekannt (Anfangszustand). $E1$ wird abgetastet.
- Aus $Z1$ ergibt sich direkt $A1$ (Anfangswert).
- Aus $Z1$ und dem abgetasteten $E1$ wird $Z2$ berechnet, der zu $T2$ wirksam wird.
- Während der Taktperiode $T1$ hat die Umwelt die Ausgangsbelegung $A1$ zu sehen bekommen. So kann sich vor $T2$ eine neue Eingangsbelegung $E2$ einstellen.
- Die in der nächsten Taktperiode abgetastete Eingangsbelegung $E(n+1)$ ergibt sich unter Einfluß der aktuellen Ausgangsbelegung $A(n)$.

T2:

- Der zu $T1$ berechnete Zustand $Z2$ wird wirksam. $E2$ wird abgetastet.
- Aus $Z2$ ergibt sich direkt $A2$ (nur Umcodierung).
- Da $Z2$ von $E1$ abhängt, hängt auch $A2$ von $E1$ ab; $E2$ kann sich nicht direkt auf $A2$ auswirken.
- Aus $Z2$ und dem abgetasteten $E2$ wird $Z3$ berechnet, der zu $T3$ wirksam wird.
- $E2$ wirkt über $Z3$ auf $A3$ ein.

Die im Takt n abgetastete Eingangsbelegung $E(n)$ kann nur auf den Zustand $Z(n+1)$ der nächsten Taktperiode einwirken. Gegenüber dem Mealy-Automaten dauert es somit einen Takt länger, bis sie die Ausgangsbelegung beeinflussen kann ($A(n+1)$).

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.

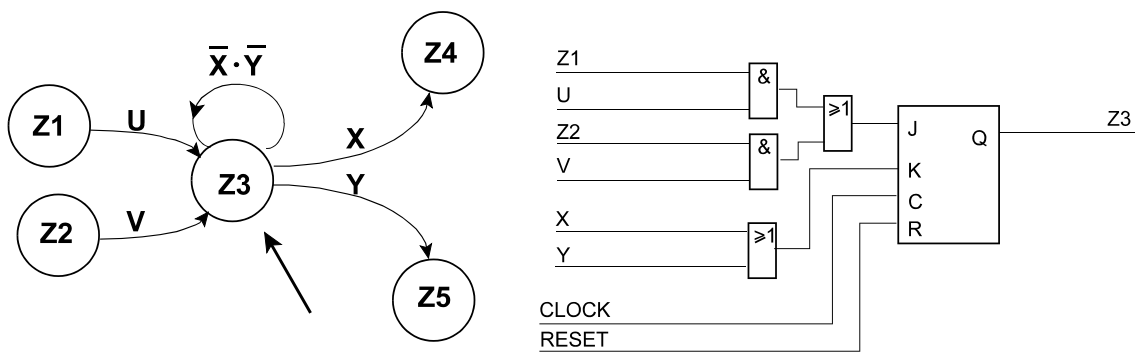
Beispiel: $Z1 \cdot U \vee Z2 \cdot V$

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.

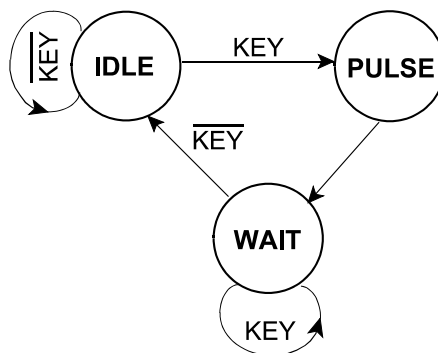
Beispiel: $K = X \vee Y$

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

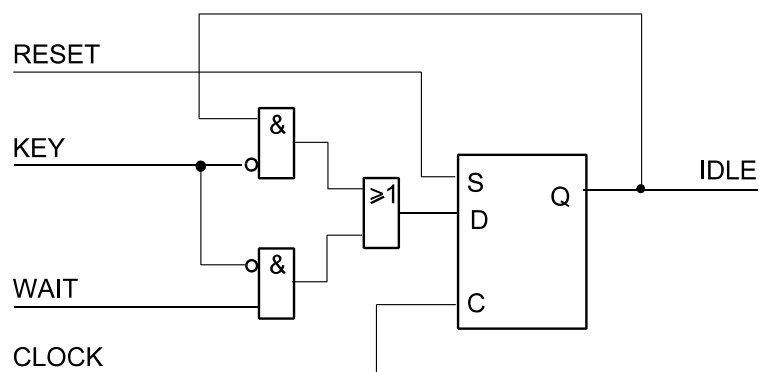
Die UND-Verknüpfung (vgl. T-Flipflop) soll verhindern, daß das Flipflop schaltet, wenn die Übergangsbedingungen aktiv werden, während sich die State Machine in anderen Zuständen befindet. In einem solchen Fall ist das hier betrachtete Flipflop ohnehin inaktiv, so daß ein Rücksetzen nicht schadet. Tritt 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-Flipflops. Hier ist $R = S = 1$ verboten. Dann müßte also tatsächlich $K = X \vee Y$ ersetzt werden durch $R = Z3 (X \vee Y)$.

Beispiel: der Single-Shot-Generator

Diese State Machine 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.



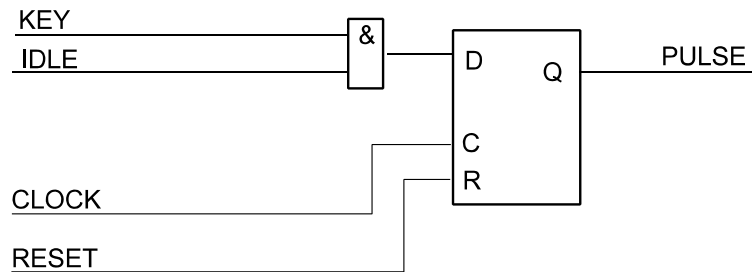
Implementierung mit D-Flipflops:



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

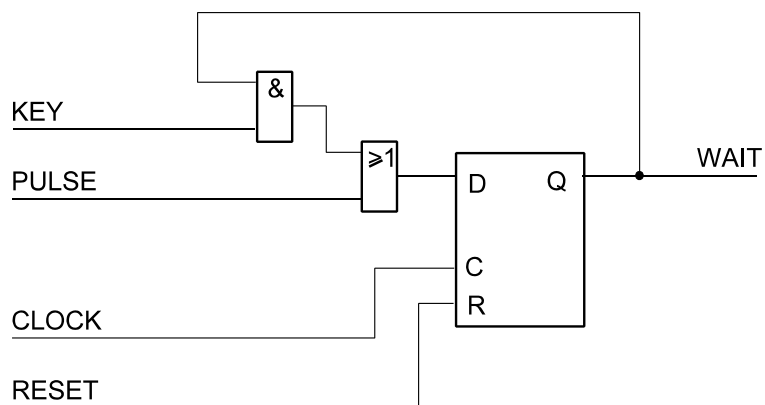
Er wird gehalten, solange die Taste nicht betätigt (= losgelassen) ist. Bei betätigter Taste wird er verlassen.

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



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

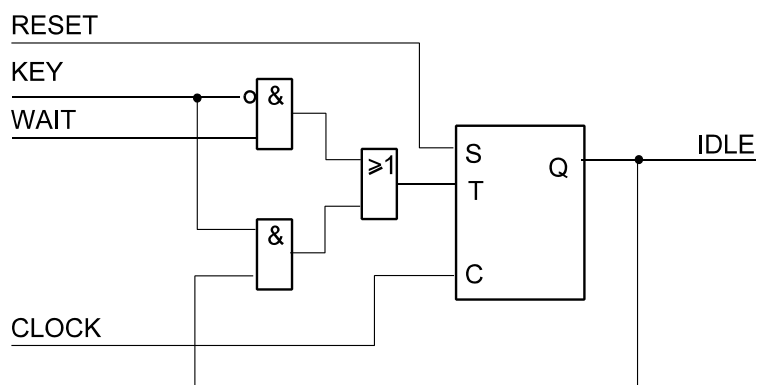
$$\text{PULSE}_D = \text{IDLE} \cdot \text{KEY}$$



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.

$$\text{WAIT}_D = \text{PULSE} \vee \text{WAIT} \cdot \text{KEY}$$

Implementierung mit T-Flipflops:

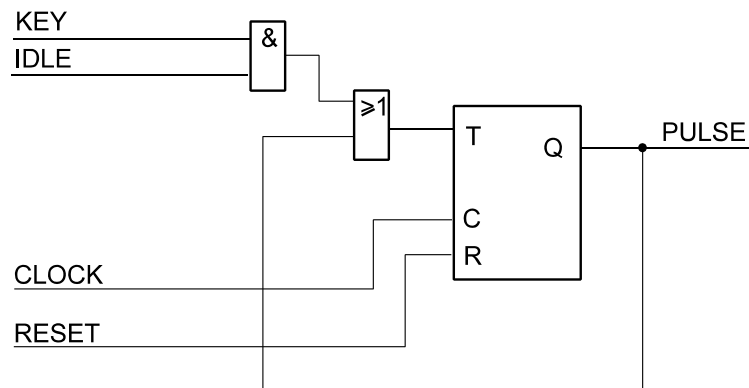


Der IDLE-Zustand wird eingeleitet, wenn im WAIT-Zustand die Taste losgelassen wird.

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

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

$$\text{IDLE}_T = \text{WAIT} \cdot \overline{\text{KEY}} \vee \text{IDLE} \cdot \text{KEY}$$

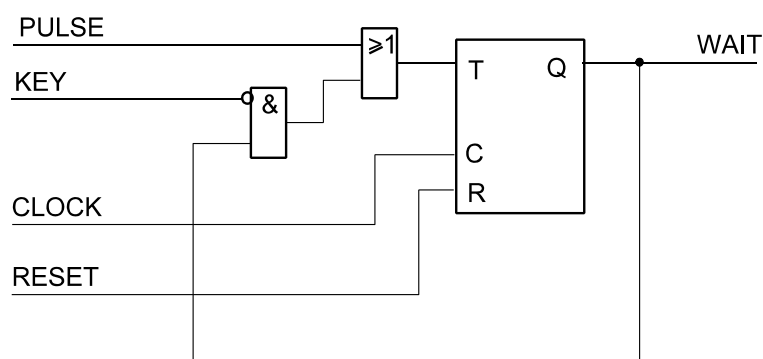


Der PULSE-Zustand wird eingeleitet, wenn im IDLE-Zustand die Taste betätigt wird.

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

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

$$\text{PULSE}_T = \text{IDLE} \cdot \text{KEY} \vee \text{PULSE}$$

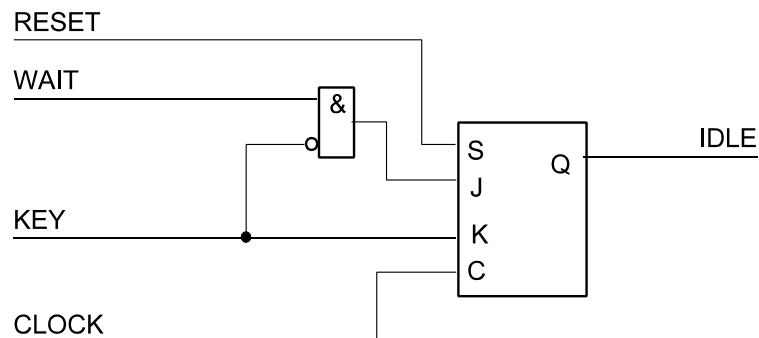


Der WAIT-Zustand wird eingeleitet, wenn der PULSE-Zustand aktiv ist.

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

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

$$\text{WAIT}_T = \text{PULSE} \vee \text{WAIT} \cdot \overline{\text{KEY}}$$

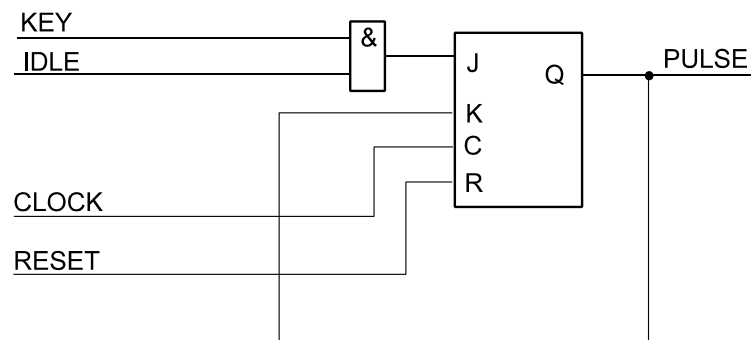
Implementierung mit JK-Flipflops (RS-Funktion):

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.

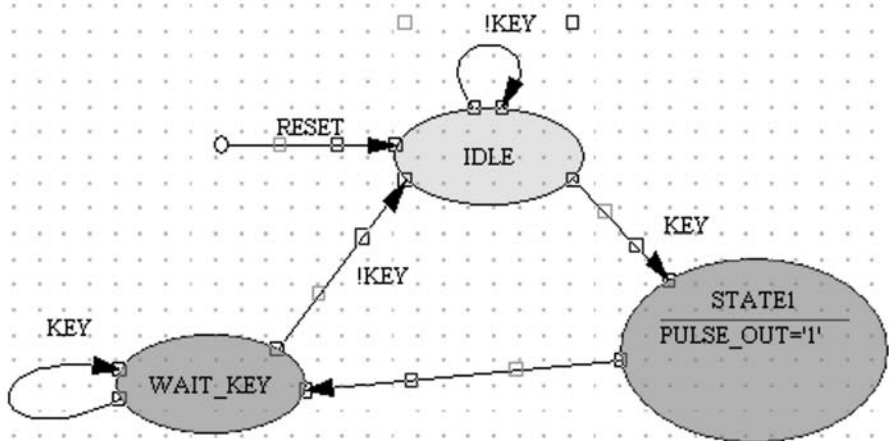
Er wird verlassen, wenn er aktiv ist (= mit dem nächsten Takt).

$$PULSE_J = IDLE \cdot KEY$$

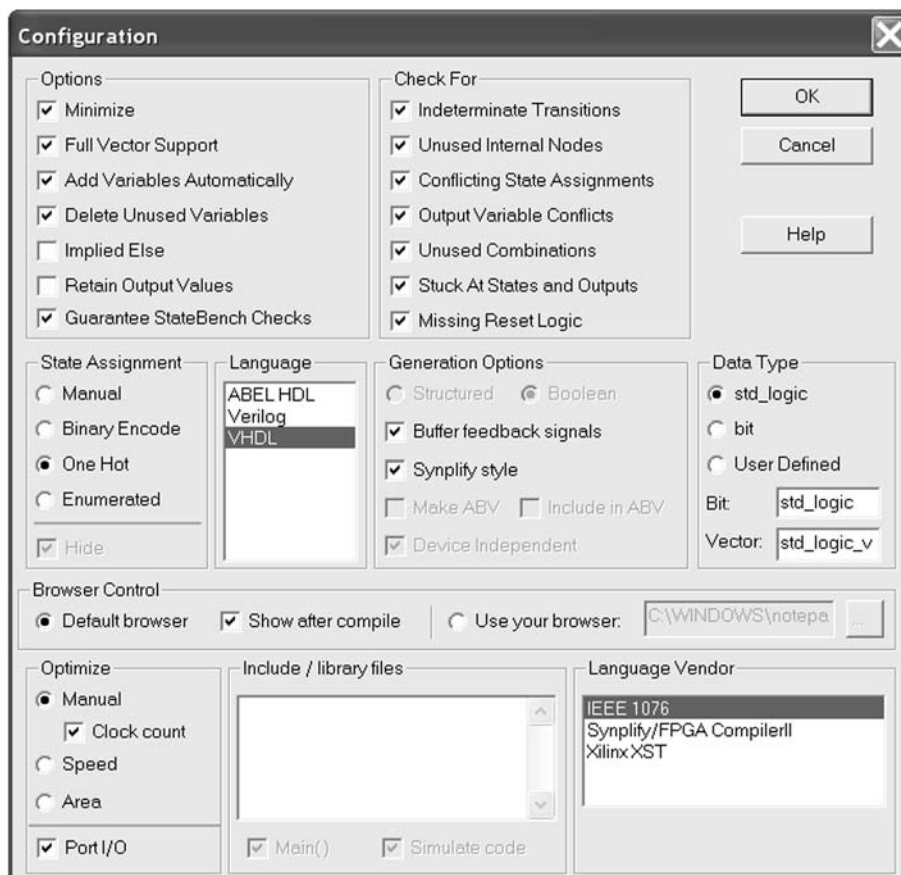
$$PULSE_K = PULSE$$

Single Shot Generator mit StateCAD

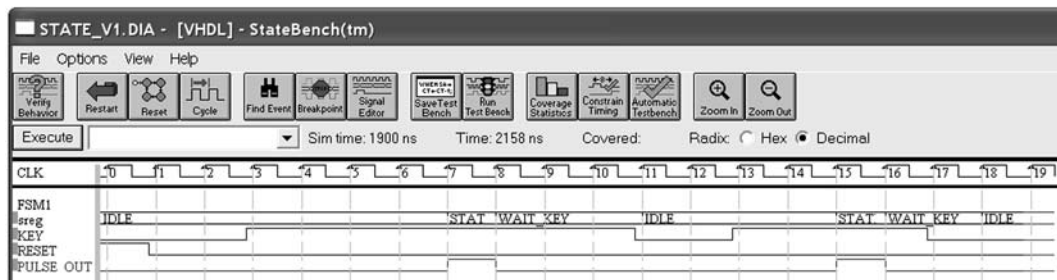
Eingabe des Zustandsdiagramms:



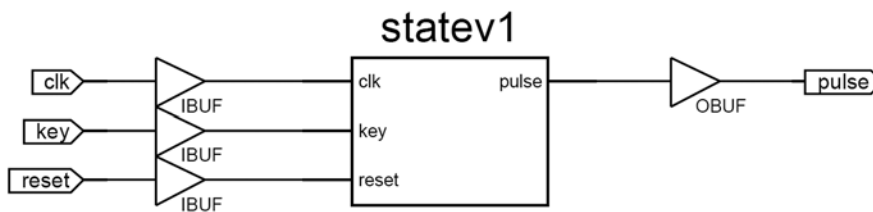
Konfigurationsmenü. Es wurde One Hot Encoding ausgewählt.



Simulation (StateBench):



Das Schaltsymbol:



Die Schaltgleichungen (vgl. S. 5 und 6):

```

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));

```