Übung 8

Bevor wir das Steuerwerk des Mikroprozessors entwickeln, wird anhand eines Modulo-6-Zählers sowie einer Glückspiels(BlackJack) die Modellierung einer FSM (Finite State Machine) mit ActiveHdl besprochen. Erstellen Sie dafür ein neues Design, um Ihre bisherige Arbeit am Mikroprozessor nicht zu stören.

Zustandsdiagramme

Ein Schaltwerk kann durch seine Zustände (States) und die Möglichkeiten der Zustandsübergänge (Transitions) dargestellt werden. Diese Darstellung wird auch als Zustandsdiagramm bezeichnet. Folgende Notation wird dafür verwendet.

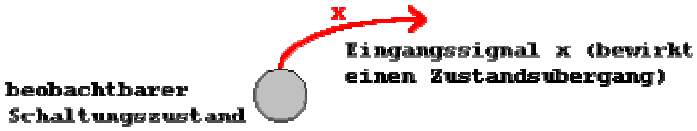

Abbildung 17: Notation

Am Beispiel eines Modulo-4-Zählers soll diese graphische Darstellung des Zustandsdiagrammes gezeigt werden. Hierbei **x** sei hier der Takt, die beiden Ziffern in den jeweiligen Zuständen spiegeln den Zählerzustand wider.

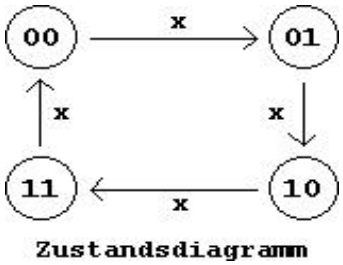

Abbildung 18: Modulo-4 Zähler

Zustandsdiagramme können in ActiveHDL einfach erstellt werden. Mit einem Rechtsklick im Design-Browser und nach Auswahl von *New->State-Diagram*, können im Source-Wizard dieselben Eingaben wie bei einem Bock-Diagram vorgenommen werden.
Anders als beim Blockdiagramm sollte man bei einer State-Machine einen Eingabeport als *clock* definieren, damit man ein synchrones Zustandsdiagramm erhält. Diese haben die Eigenschaft, dass Zustandsübergänge nur bei steigenden Taktflanken ausgelöst werden.

Modulo-6-Zähler

Implementieren Sie einen Modulo-6-Zähler mit Hilfe einer FSM. Der Zähler besitzt folgende Eingänge:

Port	***Richtung***	***Bedeutung***
CLK	IN	Taksignal
RES	IN	Asynchroner Reset
COUNT(…:…)	OUT	Zählerausgang

- Erzeugen Sie ein neues Zustandsdiagramm in ActiveHDL und überlegen Sie sich die benötigten Ein- und Ausgänge.

- Erstellen Sie zuerst die benötigten Zustände.
- Fügen Sie die Zustandsübergänge hinzu.
- Legen Sie den Startzustand fest, indem Sie einen entsprechenden Indikator setzen. Dieser Zustand soll immer dann erreicht werden, wenn RES='1' ist. Zustandsübergänge finden bei jeder steigenden Taktflanke des Taktsignals statt, es können jedoch auch weitere Bedingungen gestellt werden (im Menü mit *FSB->Condition*).
- Legen Sie die Aktionen fest, die in den jeweiligen Zuständen ausgeführt werden sollen. Diese können Sie in den Eigenschaften der jeweiligen Zustände festlegen (*Properties->Actions*, Eingabe als VHDL-Code).
- Übersetzen Sie Ihre FSM. Versuchen Sie den generierten VHDL-Code zu verstehen und Simulieren Sie diesen anschließend.

BlackJack

Mit Hilfe einer FSM modellieren wir nun das Glücksspiel BlackJack. Laden Sie sich dazu von der Seite mit den Übungsmaterialien die Dateien rng.vhd und blackjack.vhd herunter und fügen Sie diese Ihrem Design hinzu.
Im Folgenden sei noch mal der (vereinfachte) Spielablauf erklärt:
Der Spieler zieht nacheinander Karten und summiert die Werte auf. Ein As kann dabei wahlweise als 11 oder als 1 gezählt werden. Ziel ist es, möglichst viele Punkte aber höchstens 21 (= Black Jack) zu erreichen. Werden mehr als 21 Punkte erreicht, ist das Spiel bereits verloren. Ansonsten darf der Gegenspieler (Computer) so lange Karten ziehen, bis er den Spieler überboten hat (Spieler verliert) oder mehr als 21 Punkte hat und somit verliert (Spieler gewinnt).Die Procedure *ziehen* aus dem Package *blackjack* liefert dafür bei jedem Aufruf einen Zufallswert für die gezogene Karte.
Modellieren Sie das Spiel als FSM. Die Entity game hat dabei folgende Schnittstelle:

Port	***Richtung***	***Bedeutung***
Clk	IN	Taktsignal
Reset	IN	Asynchroner Reset
Weiter	IN	Weiter gibt an, ob noch eine weitere Karte gezogen werden soll bzw. ob ein weiteres Spiel begonnen werden soll.
GameOver	OUT	GameOver gibt an, ob das Spiel zu Ende ist
Sieg	OUT	Gibt an, ob das Spiel gewonnen wurde

Simulieren Sie das Spiel in einem Wavefenster.