Erstellen eines getakteten-Flip-Flops

von Jörn Schneider

1. Schritt Zuerst wird mit dem Projekt-Wizard ein neues Projekt erstellt. Vor dem Starten sollte unbedingt ein leeren Ordner angelegt werden. Dieser wird dann als Verzeichnis angegeben. Der Projektname ist beliebig, er lautet hier **T_*Flipflop***. Wichtig ist der ***Top-Level-Enity*** Name, dieser ist ***T_flipflop_top***.
Nun kann mit der Schaltfäche ***Use Existing Project Settings...*** das erste Projekt eingelesen werden. Beide Warnhinweise werden jeweils mit ***NO*** quittiert und danach mit ***Finish*** der Wizard beendet.

2. Schritt Mit ***File New*** öffnet sich ein neues Fenster.
Ausgewählt wird das ***Block Diagram***. Mit ***OK*** bestätigen.

3. Schritt Es öffnet sich ein leeres Fenster, in dem man die Schaltfläche ***Block Tool*** auswählt. Man zieht einen neuen Block in das Fenster und klickt ein mal auf die Pfeil-Schaltfläche. Mit der rechten Maustaste klickt man in den Block, es öffnet sich ein neues Fenster. Dort wählt man den Punkt ***Properties*** aus.
In dem neuen Fenster wird der Name des Blockes eingetragen. Dieser muss unbedingt mit dem ***Top-Level-Entity*** übereinstimmen, damit man später nichts mehr ändern muss. In unserem Fall lautet er ***T_flipflop_top***.
Durch klicken auf den Reiter I/O gelangt man zu den nächsten Einstellungen. Unser RS-Flip-Flop soll zwei Eingänge (***clk*** und ***reset***) und einen Ausgang (***Q***) haben. Durch klicken in das Feld ***<NEW>*** kann man dort den Namen des Ein- bzw. Ausganges eingeben. Mit ***ENTER*** bestätigen. Durch einen weiteren Klick in das Feld ***INPUT*** kann dort auch ***OUTPUT*** ausgewählt werden. Wenn alles richtig gemacht wurde, sieht unser Block nun so aus. Nun mit ***File Save...as*** unter ***T_flipflop_top.bdf*** speichern.
Mit der rechten Maustaste noch mal in den Block klicken, es öffnet sich wieder das im Schritt 5 beschriebene Fenster. Diesmal wird allerdings der unten gezeigte Punkt ausgewählt. Danach öffnet sich ein neues Fenster, in dem ***VHDL*** ausgewählt wird und der Dateiname unbedingt ***T_flipflop_top.vhd*** lauten muss, da er unsere ***Top-Level-Entity*** enthält. Das Fenster mit ***OK*** schließen.

4. Schritt Nun wurde automatisch ein VHDL-File erzeugt. Aus diesem kann man alle grünen Kommentare (egal was da steht!) löschen. Unser VHDL-File sieht nun so aus.

```
1
2   LIBRARY ieee;
3   USE ieee.std_logic_1164.all;
4
5   ENTITY T_flipflop_top IS
6
7   PORT
8   (
9       clk   : IN STD_LOGIC;
10      reset : IN STD_LOGIC;
11      Q     : OUT STD_LOGIC
12  );
13
14  END T_flipflop_top;
15
16  ARCHITECTURE T_flipflop_top_architecture OF T_flipflop_top IS
17
18
19  BEGIN
20
21  END T_flipflop_top_architecture;
```

5. Schritt Bevor wir nun dieses verändern, können wir die Pins noch zuordnen. Dazu die Schaltfläche  anklicken und nach Fertigstellung den Pin-Planer  aufrufen.

Node Name	Direction	Location
clk	Input	PIN_M23
Q	Output	PIN_G19
reset	Input	PIN_M21
<<new node>>		

Dabei liegt ***clk*** auf dem ***Key0*** und ***reset*** auf dem ***Key1***. ***Q*** ist der ***LEDR0*** zugeordnet. Den Pin-Planer einfach schließen, es muss nichts gespeichert werden.

6. Schritt Nun müssen wir nur noch das Flip-Flops mit VHDL programieren. Der ***ENTITY***-Block ist komplett fertig und darf nicht mehr verändert werden.

```
16  ARCHITECTURE T_flipflop_top_architecture OF T_flipflop_top IS
17
18  signal intern :std_logic;
19
20  BEGIN
21
22  process(clk, reset)
23
24  begin
25
26  if rising_edge(clk) then
27
28     if intern = '0' then
29     intern <= '1';
30     else
31     intern <= '0';
32     end if;
33
34     if reset = '0' then
35     intern <= '0';
36     end if;
37
38  end if;
39
40  Q <= intern;
41
42  end process;
43
44  END T_flipflop_top_architecture;
```

Kurze Programmerklärung

Zuerst wird mit ***SIGNAL intern*** ein internes Signal definiert, dass vom Prinzip her einem Eingabe- oder Ausgabesignal entspricht, aber nicht nach außen verbunden ist.

Achtung: Bei den Tastern bedeutet eine logisch '1' dass der Taster nicht gedrückt wurde. Daher wird ein gedrückter Taster mit einer logischen '0' abgefragt!

Mit ***BEGIN*** starten wir die Definition unserer Beschreibung

PROCESS(clk) bedeutet, dass dieser Block ausgeführt wird, wenn das ***clk***-Signal eine steigende Flanke (***rising_edge***) hat. Er wird ebenfalls mit einem BEGIN gestartet.

Die erste IF....THEN Bedingung fragt ob das interne Signal den logischen Wert '0' hat. Wenn ja, dann wird das Signal in logisch '1' geändert, wenn nein in logisch '0'. Das entspricht einer Umschaltung.

Die zweite IF....THEN Bedingung fragt den RESET-Taster ab. Das interne Signal wird auf logisch '0' gezogen. Logisch '0' bedeutet gedrückter Taster (siehe oben!).

Mit Q <= intern wird Q mit dem internen Signal intern verbunden.

7. Schritt Das fertige Projekt wird nun mit ***Compilation*** übersetzt, was einige Minuten dauern kann und dann mit dem ***Programmer*** auf das Board übertragen. Die Bedienung des ***Programmer***s ist in der ersten Anleitung ausführlich beschrieben.