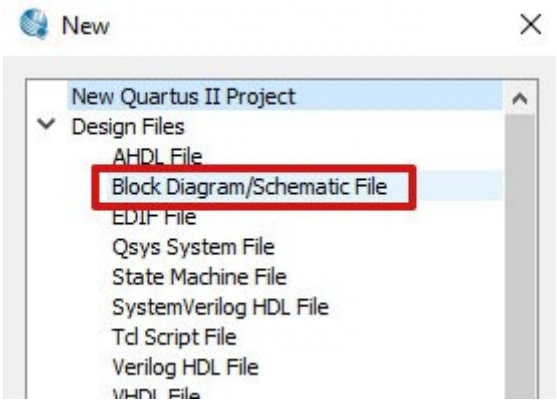
Erstellen eines RS-Flip-Flops

mit Hilfe eines Blockdiagramms

von Jörn Schneider

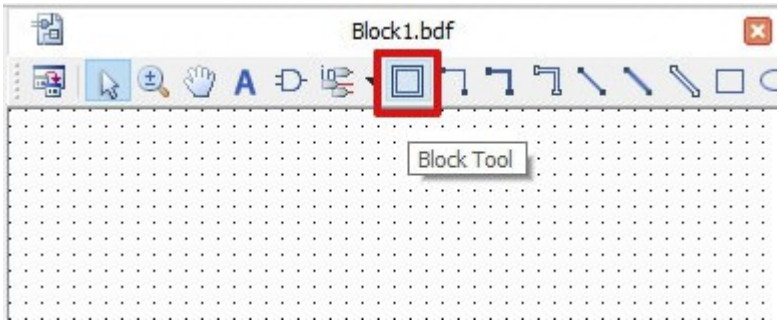
1. Schritt Zuerst wird mit dem Projekt-Wizard ein neues Projekt erstellt. Vor dem Starten sollte unbedingt ein leeren Ordner angelegt werden. Dieser wird dann als Verzeichnis angegeben. Der Projektname ist beliebig, er lautet hier ***RS_Flipflop***. Wichtig ist der ***Top-Level-Enity*** Name, dieser ist ***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.

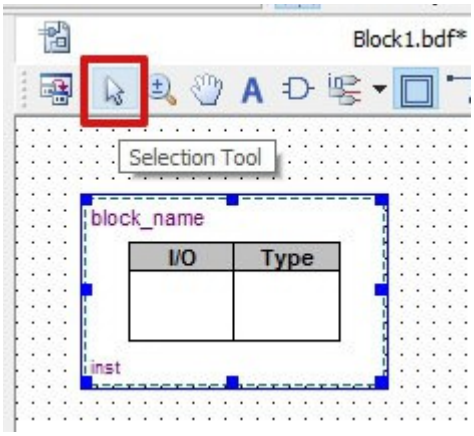


Ausgewählt wird das ***Block Diagram***. Mit ***OK*** bestätigen.

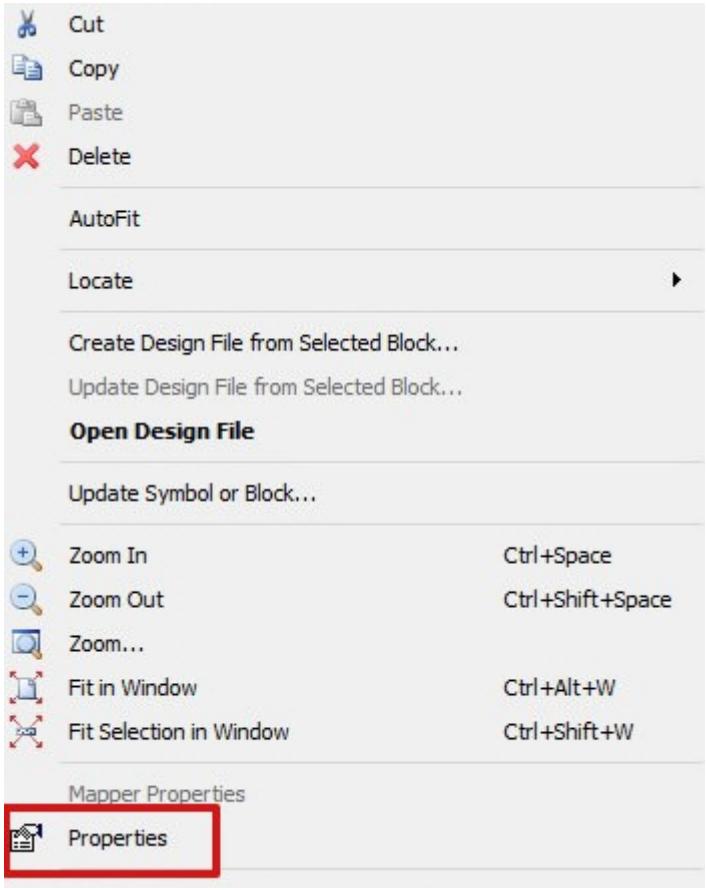
3. Schritt Es öffnet sich ein leeres Fenster, in dem man die Schaltfläche ***Block Tool*** auswählt.



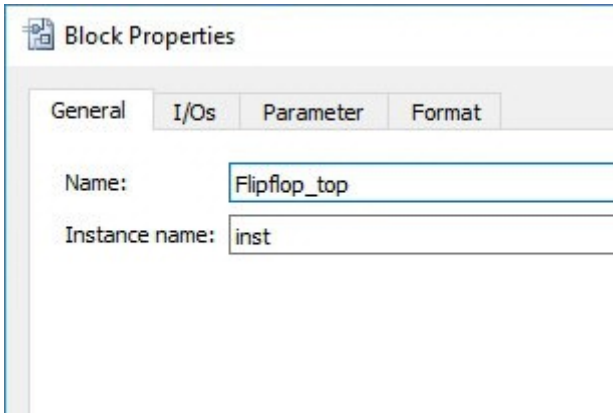
4. Schritt Man zieht einen neuen Block in das Fenster und klickt ein mal auf die Pfeil-Schaltfläche.



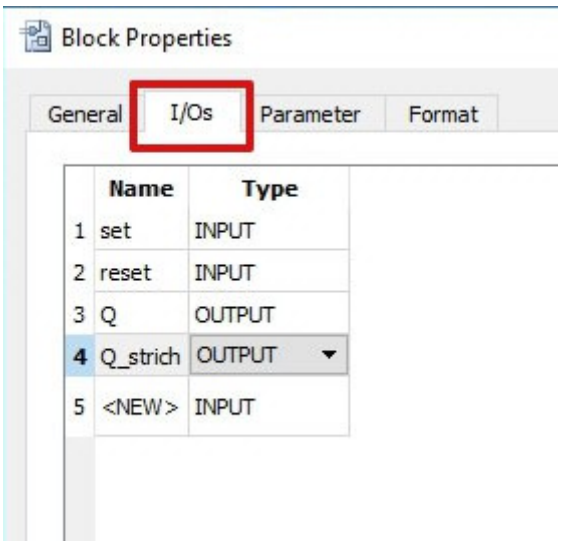
5. Schritt Mit der rechten Maustaste klickt man in den Block, es öffnet sich ein neues Fenster. Dort wählt man den Punkt ***Properties*** aus.



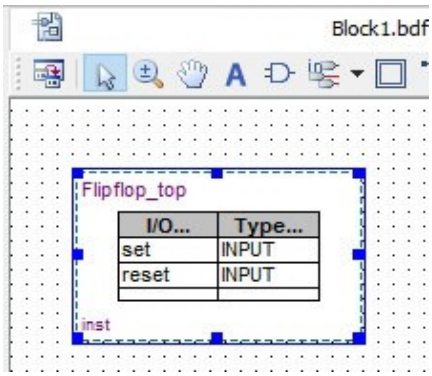
6. Schritt In dem neuen Fenster wird der Name des Blockes eingetragen. Dieser muss unbedingt mit dem ***Top-Level-Entity*** übereinstimmen, damit man später nichts mehr ändern muss. In unserem Fall lautet er ***Flipflop_top***.



7. Schritt Durch klicken auf den Reiter I/O gelangt man zu den nächsten Einstellungen. Unser RS-Flip-Flop soll zwei Eingänge (***set*** und ***reset***) und zwei Ausgänge (***Q*** und ***Q_strich***) haben.

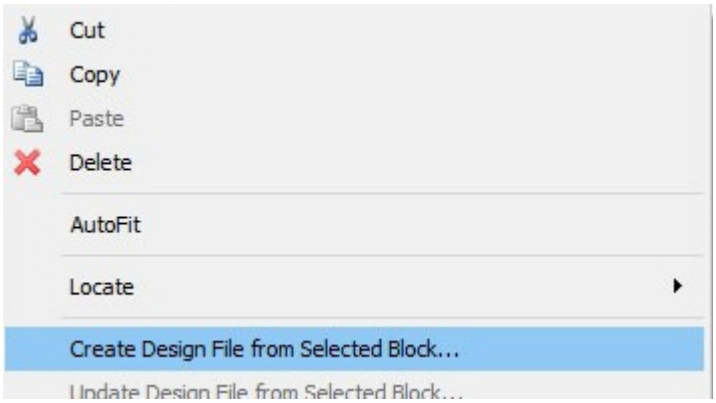


Durch klicken in das Feld ***<NEW>*** kann man dort den Namen des Ein- bzw. Ausganges eingeben. Mit ***ENTER*** bestätigen. Durch einen weiteren Klick in das Feld ***INPUT*** kann dort auch ***OUTPUT*** ausgewählt werden. Wenn alles richtig gemacht wurde, sieht unser Block nun so aus.

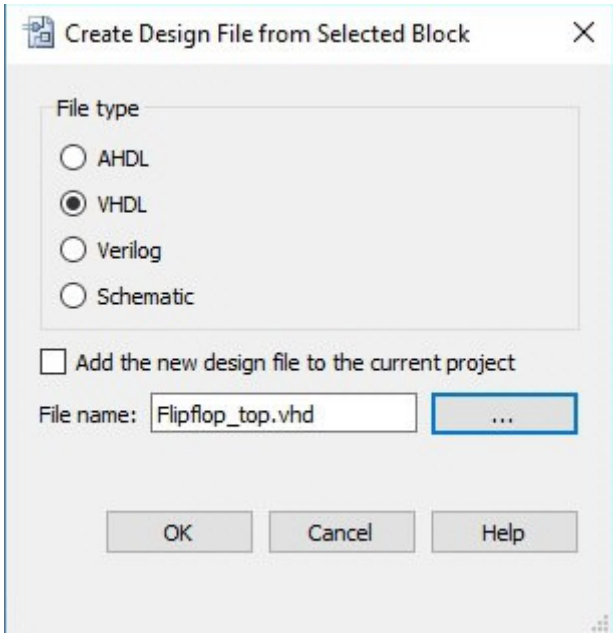


Nun mit ***File Save...as*** unter ***Flipflop_top.bdf*** speichern.

8. Schritt 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 ***Flipflop_top.vhd*** lauten muss, da er unsere ***Top-Level-Entity*** enthält. Das Fenster mit ***OK*** schließen.



9. 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   LIBRARY ieee;
2   USE ieee.std_logic_1164.all;
3
4   ENTITY Flipflop_top IS
5
6   PORT
7   (
8   set : IN STD_LOGIC;
9   reset : IN STD_LOGIC;
10  Q : OUT STD_LOGIC;
11  Q_strich : OUT STD_LOGIC
12  );
13
14
15  END Flipflop_top;
16
17  ARCHITECTURE Flipflop_top_architecture OF Flipflop_top IS
18
19
20  BEGIN
21
22  END Flipflop_top_architecture;
23
```

10. 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
out Q	Output	PIN_G19
out Q_strich	Output	PIN_F19
in reset	Input	PIN_M21
in set	Input	PIN_M23
<<new node>>		

Dabei liegt ***set*** auf dem ***Key0*** und ***reset*** auf dem ***Key1***. ***Q*** ist der ***LEDR0*** zugeordnet, ***Q_strich*** der ***LEDR1***.

11. Schritt Nun müssen wir nur noch das Flip-Flops mit VHDL programieren. Der ***ENTITY***-Block ist komplett fertig und darf nicht mehr verändert werden. Das unten gezeigte Program ist eine Möglichkeit der Programmierung.

```
17  ARCHITECTURE Flipflop_top_architecture OF Flipflop_top IS
18
19    SIGNAL intern : std_LOGIC;
20
21  BEGIN
22
23  PROCESS (set,reset)
24
25    BEGIN
26
27  IF SET = '0' THEN
28
29      intern <= '1';
30    END IF;
31
32  IF reset = '0' THEN
33
34      intern <= '0';
35    END IF;
36
37    Q <= intern;
38    Q_Strich <= not intern;
39
40    END PROCESS;
41
42
43  END 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(set,reset) bedeutet, dass dieser Block ausgeführt wird, wenn sich das set oder reset-Signal ändert. Er wird ebenfalls mit einem BEGIN gestartet.

Die erste IF....THEN Bedingung fragt das Drücken des SET-Tasters ab. Ist dieser gedrückt, wird das interne Signal auf logisch '1' gezogen.

Die zweite IF....THEN Bedingung fragt den RESET-Taster ab. Das interne Signal wird auf logisch '0' gezogen.

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

Mit Q_strich <= not intern wird Q_strich mit dem invertierten Signal intern verbunden.

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