Einführung in Computer Microsystems

Prof. Dr. Andreas Koch
Thorsten Wink

TECHNISCHE UNIVERSITÄT DARMSTADT

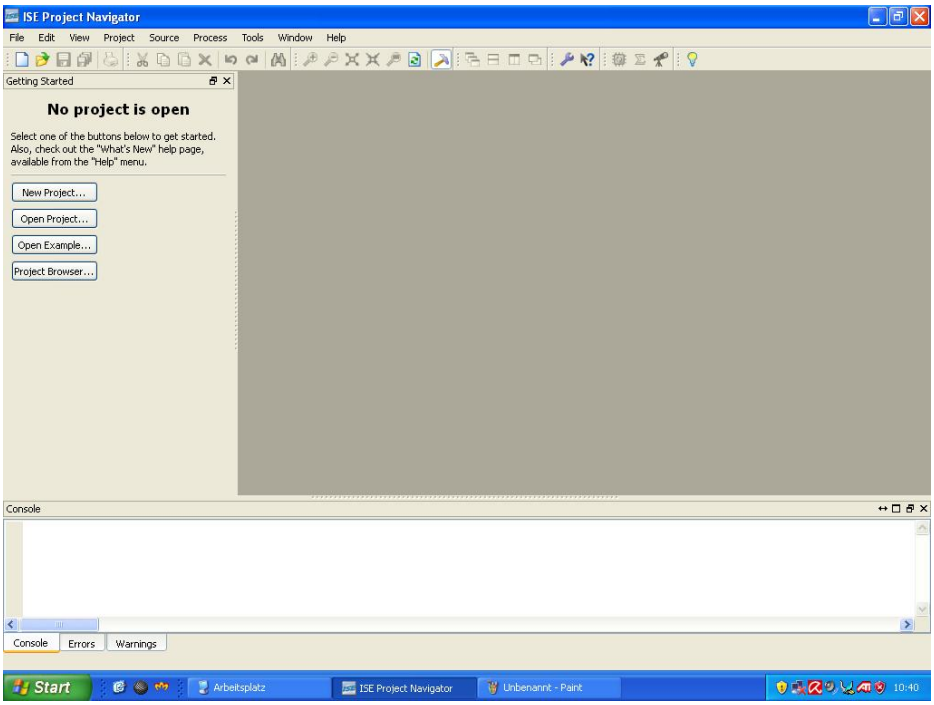
Sommersemester 11
Einrichten eines Projekts und Simulation in XILINX ISE 11.3

Abschnitt 1 Einleitung

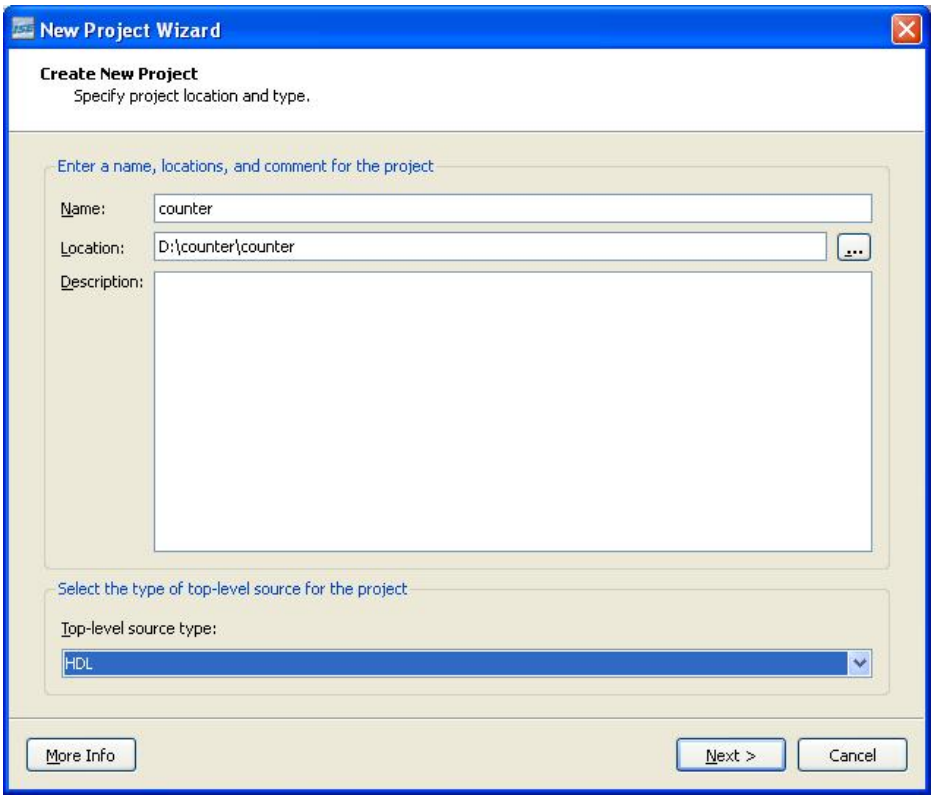
Dieses Tutorial gibt eine kurze Einführung in das Synthese- und Simulationstool XILINX ISE 11.3. Mit Hilfe des frei verfügbaren Webpacks ist eine für die Vorlesung ausreichende Version frei verfügbar. Auf den Poolrechner der RBG ist ISE ebenfalls installiert, es kann über das Kommando *ise* gestartet werden.
In diesem Tutorial wird ein einfacher 4-Bit Zähler beschrieben und simuliert. Ziel ist eine Einführung in die Erstellung eines neuen Projekts und in die automatische Testbenchgenerierung.

Abschnitt 2 Erstellung eines neuen Projekts

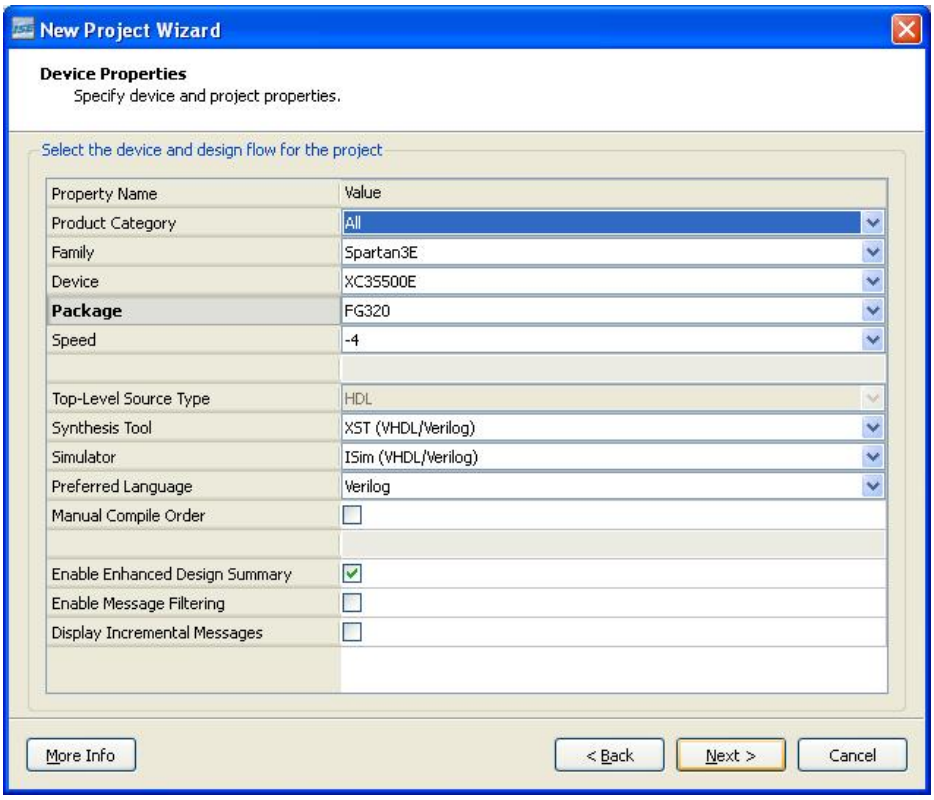
Nach dem Start muss zunächst ein Projekt angelegt werden, damit ISE einige wichtige Parameter kennt. Dies wird über *New Project* gestartet.



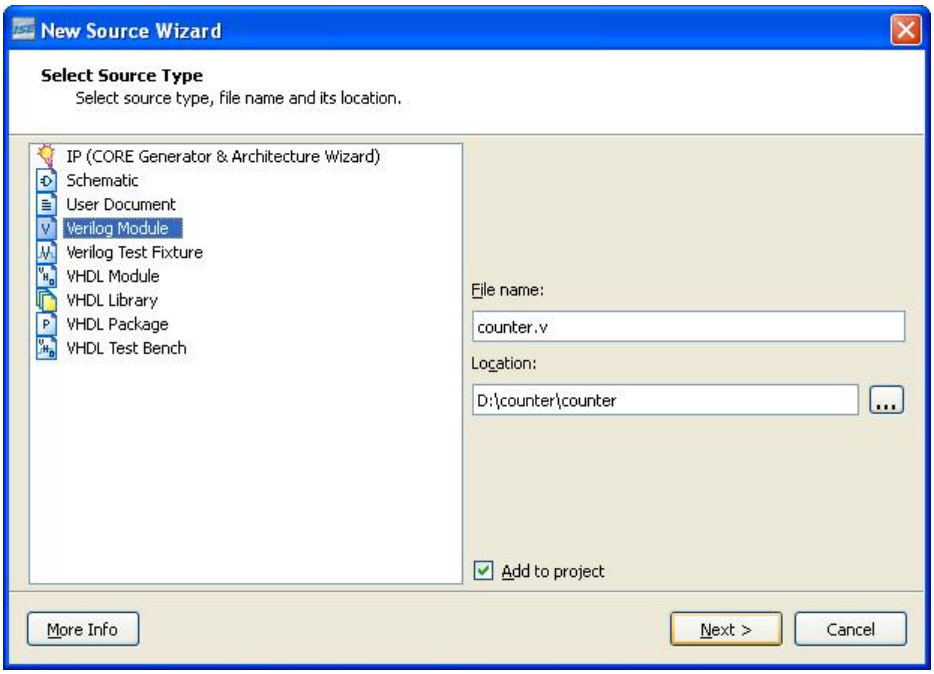
Es muss ein Name für das Projekt und ein Speicherort auf der Festplatte angegeben werden. Als *Top-level-source-type* muss *HDL* angegeben werden.



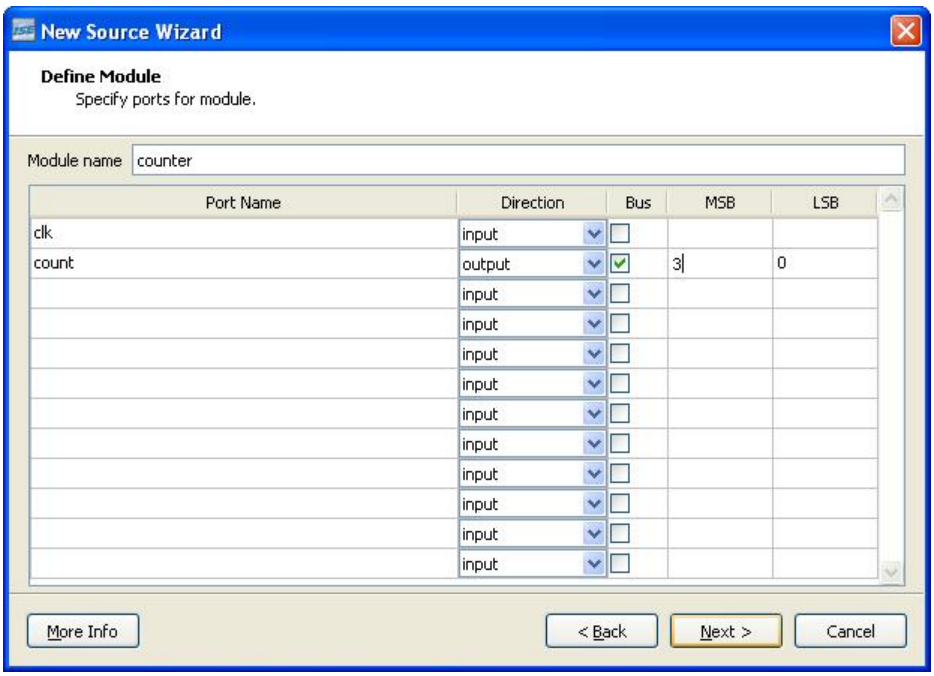
Im nächsten Dialog folgen die Einstellung zu der verwendeten Zielhardware. Hier muss folgendes ausgewählt werden: *Family: Spartan3E, Device: XC3S500E, Package FG320, Speed -4*. Als *Synthesis Tool* ist *XST* einzustellen, als *Simulator ISim*. *Preferred Language* ist *Verilog*.



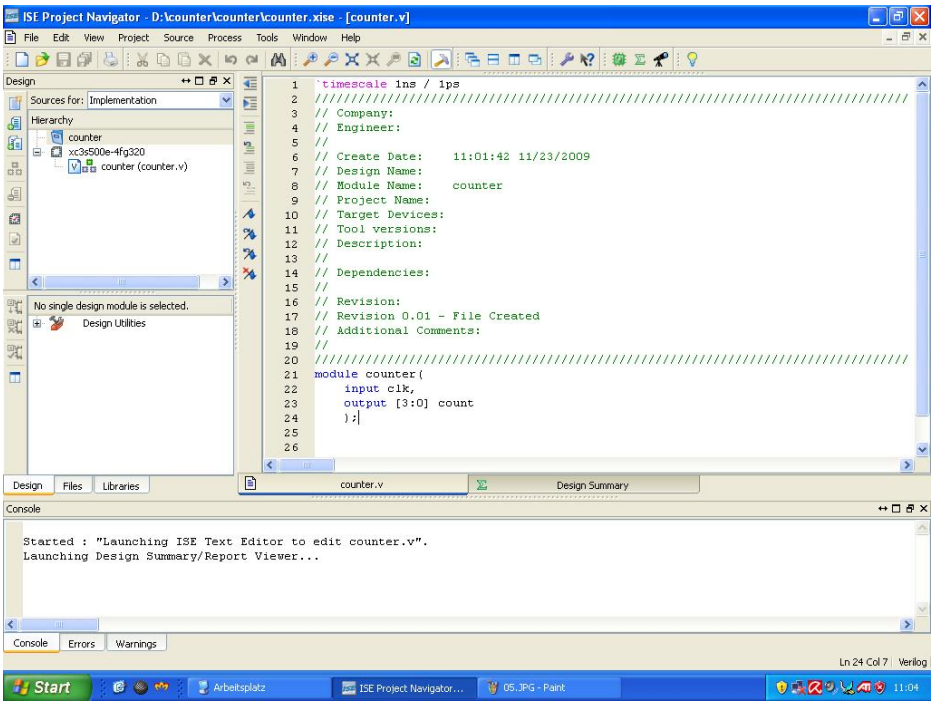
Danach erstellen wir ein neues Modul, welches den Zähler aus dem Beispiel enthält. Dazu wählen Sie *Verilog Module* und vergeben einen Dateinamen.



Im nächsten Schritt können Sie die Ports des Moduls angeben. Unser Beispiel hat den Takteingang *clk* und den Ausgang *count*, der 4 Bit breit ist [3:0].



Da wir keine schon existierenden Dateien zum Projekt hinzufügen wollen, überspringen Sie den folgenden Dialog mit *Next*.
Nun wird nochmal eine Übersicht des Projekts angezeigt. Klicken Sie auf *Finish*, um die Dateien zu erzeugen. Nun wird das Projekt angelegt und der generierte Coderahmen *counter.v* wird angezeigt.

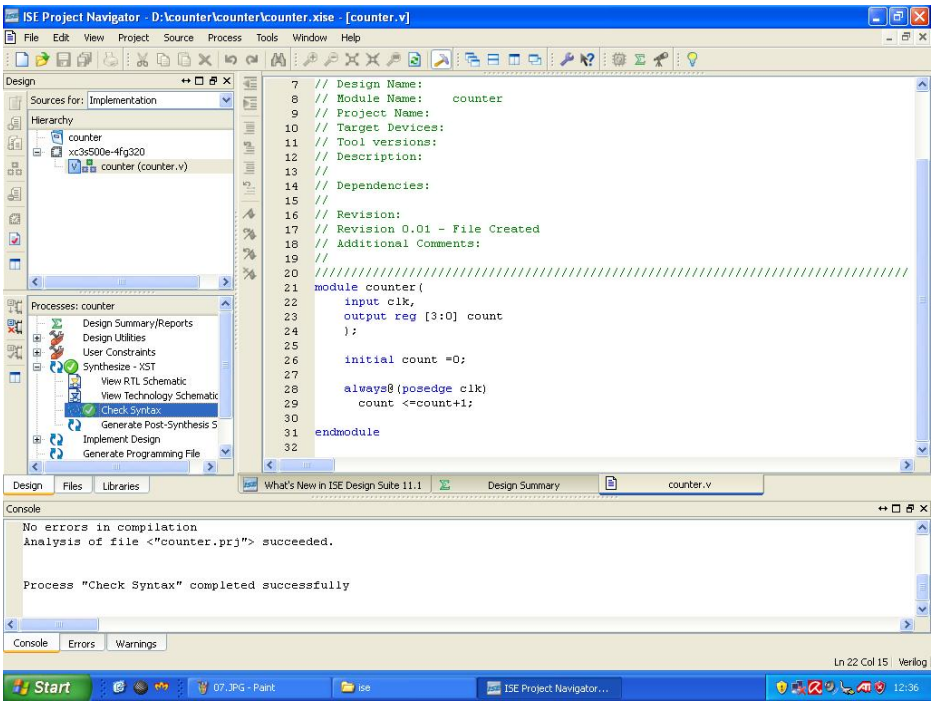


Abschnitt 3 Vervollständigung des Codes

Bei der Projekterstellung wurde bereits der Modulkopf mit den Ein- und Ausgängen generiert. Nun muss noch der eigentliche Zähler in Verilog beschrieben werden. Der Zähler soll synchron zum Takt arbeiten, deshalb wird das Hochzählen in einem Prozess, der auf die steigende Taktflanke sensitiv ist, implementiert. Der Ausgang *count* muss aus diesem Grund als *reg* deklariert werden. Der vollständige Code sieht wie folgt aus:

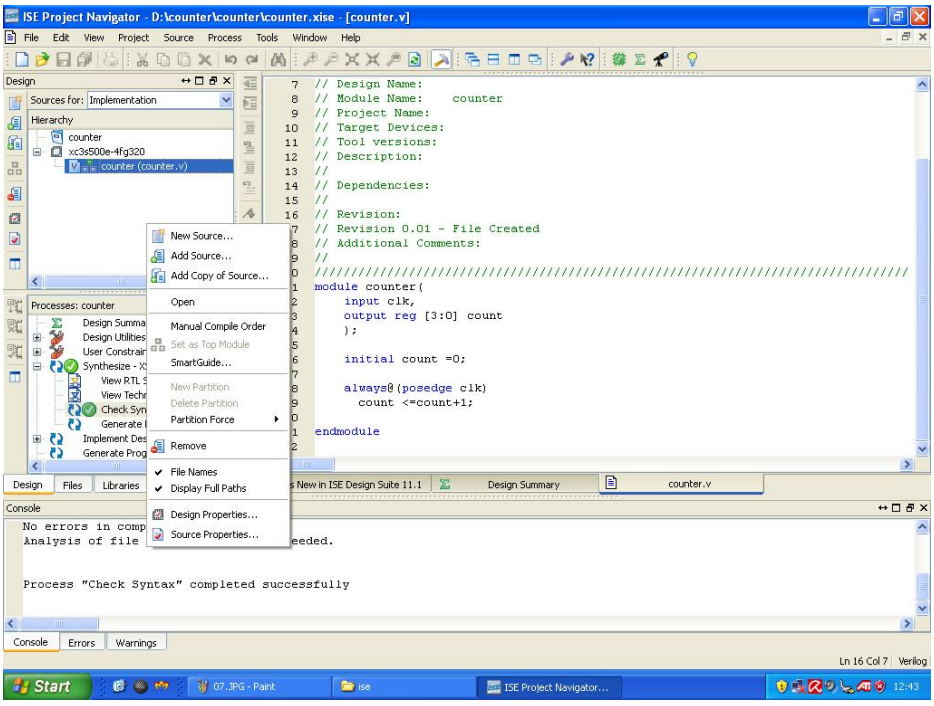
```
module counter(
  input clk,
  output reg[3:0] count
  );

  initial count = 0;

  always @(posedge clk)
    count <= count + 1
endmodule
```

Nun kann ein Syntax-Check durchgeführt werden. Dazu klicken Sie auf *Check Syntax*, falls Fehler gefunden werden werden diese unten angezeigt. Falls nicht, ist der Code syntaktisch korrekt (was nicht unbedingt heißt, dass er auch semantisch korrekt ist).

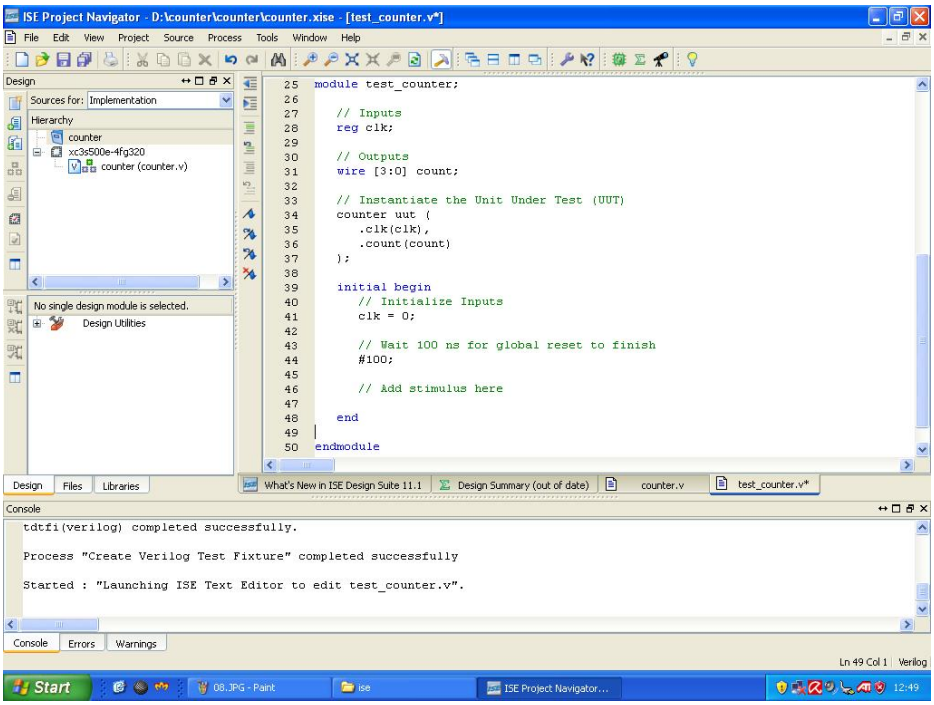


Abschnitt 4 Simulation

Um zu überprüfen, ob der Verilog-Code semantisch korrekt ist (also dass er das tut, was Sie sich beim Schreiben gedacht haben), ist eine Simulation des Moduls nötig. Tests in echter Hwardware sind sehr aufwändig, deshalb wird als erster Schritt immer eine Simulation mit einem Hardware-Simulator durchgeführt. Hierbei werden die Eingangsvariablen auf feste Werte gelegt und am Ausgang beobachtet, ob die Schaltung die gewünschte Funktion ausführt. Auch alle Zwischensignale sind im Simulator im Gegensatz zur echten Hardware leicht zu beobachten. In dieser Vorlesung wird der in ISE enthaltene Simulator ISim verwendet.
Bevor die Simulation ausgeführt werden kann, muss eine Testumgebung (Testbench) erzeugt werden (mehr dazu in der Vorlesung). In ISE gibt es einen automatischen Testbenchgenerator, der ein Grundgerüst für die Testumgebung erzeugen kann. Hierzu muss eine neue Datei erzeugt werden. Wählen Sie nach einem Rechtsklick im Hierarchy-Bereich aus dem Kontextmenü *New Source*.



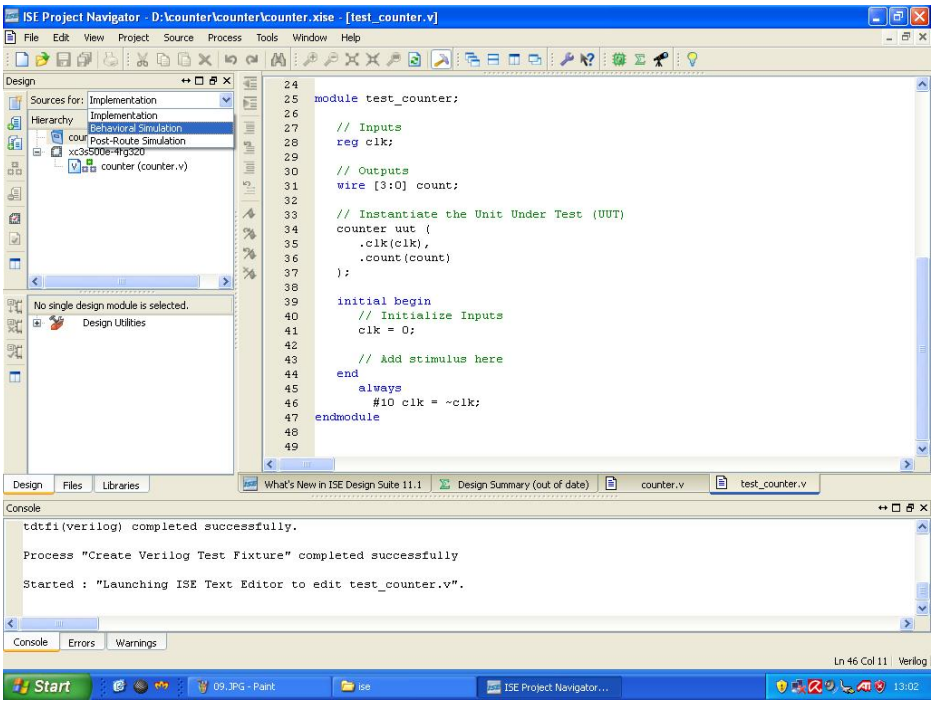
Wählen Sie als *Source Type Verilog Text Fixture* und vergeben Sie einen Dateinamen. Nach einem Klick auf *Next* wählen Sie das Modul aus, welches simuliert werden soll, in diesem Beispiel ist nur unser einziges Modul *counter* in der Auswahlliste. Nachdem die Erzeugung mit *Finish* abgeschlossen wurde, wird der erzeugte Code der Datei *test_counter.v* angezeigt.



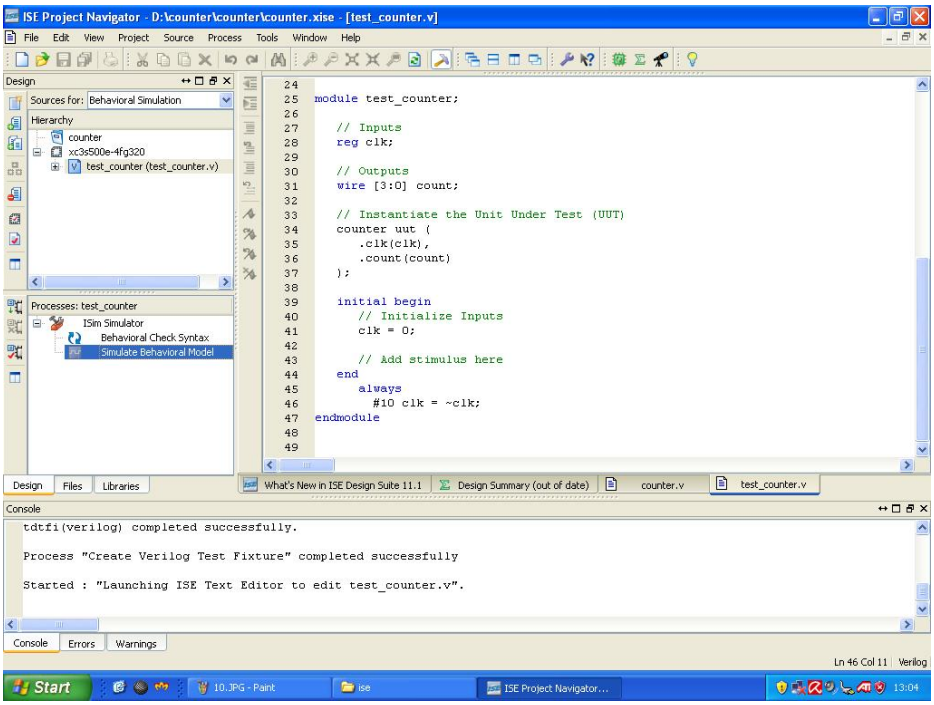
Unser Beispiel hat nur den Takteingang clk. Dieser wird durch die beiden folgenden Zeilen beschrieben:

```
alway
  #10 clk = ~clk;
```

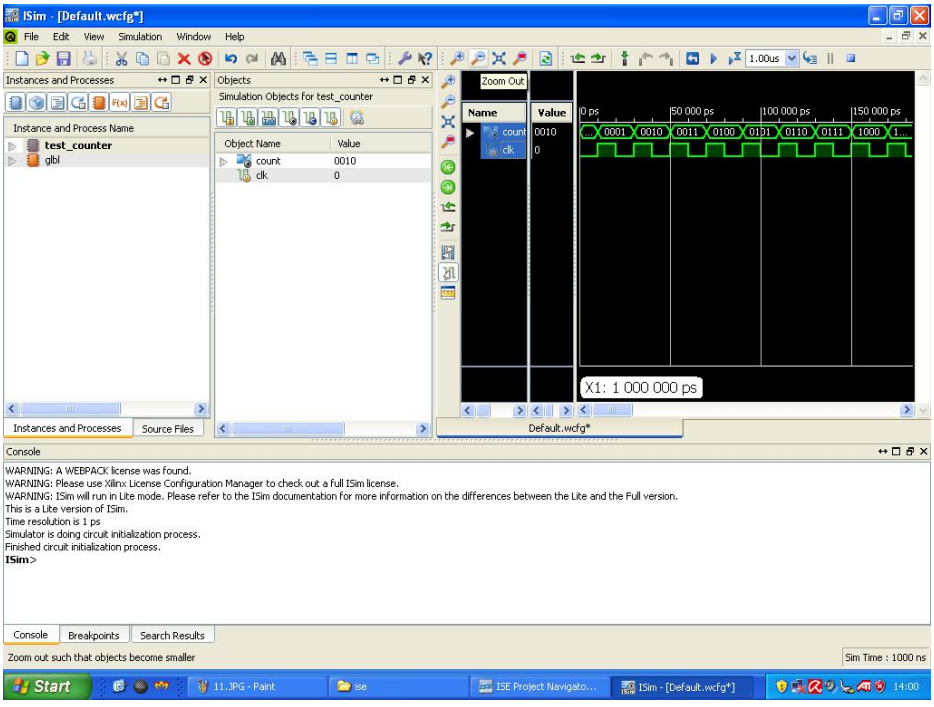
Zur Simulation muss das System auf *Behavioral Simulation* umgestellt werden:



Danach kann der Simulator durch Doppelklick auf *Simulate Behavioral Model* gestartet werden.



Es öffnet sich ein neues Fenster, in dem der Simulator die Ergebnisse darstellt. Unser Beispiel hat nur 2 Signale, sie werden automatisch in das Waveform-Fenster rechts übernommen und ihr Werteverlauf wird augezeigt. Wahrscheinlich muss der Zoom verkleinert werden, um eine gute Darstellung zu erhalten. Dann sollten Sie sowohl das Taktignal als auch den Wechsel des Zählers bei jeder steigenden Taktflanke gut beobachten können.



Da Simulatorfenster ist in mehrere Bereiche aufgeteilt. Im linken Bereich sehen Sie die Modulhierachie (in diesem Beispiel nicht vohanden, das Projekt besteht nur aus einem Modul). In der Mitte sind die Signale dargestellt, die zum entsprechenden Modul gehören. Sie können von dort zum Waveform-Fenster hinzugefügt werden. In der Konsole werden eventuelle Fehler ausgegeben und eventuell in der Testbench vorhandene *$display*-Aufrufe ausgeführt.

Abschnitt 5 Fazit

Mit diesem Leitfaden sollten Sie in der Lage sein, eigene Verilog-Projekte in XILINX ISE anzulegen und zu simulieren.