Aufgabe 1: Entwurf, Simulation und Test einer kombinatorischen Schaltung

G. Kemnitz, C. Giesemann, TU Clausthal, Institut für Informatik

21. Mai 2015

Zusammenfassung

Es ist eine kombinatorische Schaltung aus wenigen Gattern zu simulieren, synthesefähig zu beschreiben, in den programmierbaren Logikschaltkreis der Testbaugruppe zu laden und die Schaltung zu testen. Lernziel ist das Kennenlernen der unterschiedlichen Modellsichten und Bearbeitungswerkzeuge für die Simulation und den Entwurf einer digitalen Schaltung.

1 Vorbereitung

Zur Abbarbeitung der nachfolgenden Aufgaben ist das Archiv »PraktVHDL1_1.zip« in das Arbeitsverzeichnis für das Praktikum zu laden. Unter Linux erfolgt das Entpacken mit dem Kommando:

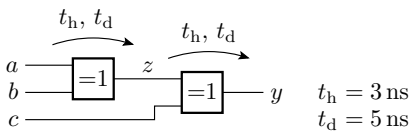
```
unzip PraktVHDL1_1.zip
```

Unter Windows ist in C:\users\<login_name>\Dokuments ein Praktikumsverzeichnis anzulegen, dort das Archiv hinladen, mit der rechten Maustaste im Dateimanager auf dem Archiv mit »Extract All...« der Entpackvorgang zu starten und das Praktikumsverzeichnisses als Zielverzeichnis anzugeben. Es werden folgende Dateien entpackt[1]:

```
Aufg1/Sim_GHDL/Exor2_Sim.vhdl   # Simulationsbeschreibung Testobjekt
Aufg1/Sim_GHDL/Exor2_Synth.vhdl # Synthesebeschreibung Testobjekt
Aufg1/Sim_GHDL/Test_Exor2.vhdl  # Testrahmen
Aufg1/Sim_GHDL/Test_Exor2.sav   # Visualisierungseinstellungen GTKWAVE
Aufg1/Sim_GHDL/Exor2.sh         # Kommandodatei Simulation unter Linux
Aufg1/Sim_GHDL/Exor2.bat         # Kommandodatei Simulation unter Windows
Aufg1/Exor2/Exor2.xise          # Projektdatei für ISE
Aufg1/Exor2/Exor2.ucf           # UCF-Datei für ISE
```

2 Simulation mit GHDL und GTKWAVE

Beispiel ist die nachfolgende Schaltung aus zwei EXOR-Gattern mit Halte- und Verzögerungszeiten:



[1] Unter Windows wird ein Teil der Dateiendung standardmäßig nicht angezeigt.

Die Beschreibung der Entwurfseinheit besteht aus zwei nebenläufigen Signalzuweisungen, die bei jeder Änderung der Signale auf der rechten Seite dem Signal auf der linken Seite des Zuweisungsoperators »<=« nach der Haltezeit den Wert »ungültig« und nach der Verzögerungszeit den neuen gültigen Signalwert zuweisen:

```
z <= 'X' after 3 ns, a xor b after 5 ns;
y <= 'X' after 3 ns, z xor c after 5 ns;
```

Die komplette Beschreibung steht in der Datei »Aufg1/Sim_GHDL/Exor2_Sim.vhdl«. Zur Simulation der Schaltung wird zusätzlich der Testrahmen »Aufg1/Sim_GHDL/Test_Exor2.vhdl« benötigt, der die Eingabesignale erzeugt und die zu testende Schaltung als Instanz enthält:

```
Testobjekt: entity work.Exor2 port map(a=>x(0), b=>x(1), c=>x(2));
Testprozess: process
begin
 wait for 20 ns; x <= "000"; wait for 20 ns; x <= "010";
 wait for 20 ns; x <= "110"; wait for 20 ns; x <= "111";
 wait for 20 ns; x <= "100"; wait for 20 ns; x <= "101";
 wait for 20 ns; x <= "001"; wait;
end process;
```

Das Testobjekt ist ohne Architekturnamen instanziiert. Das ist Voraussetzung, damit der Testrahmen später auch für die Post-Route-Simulation in »ISE« verwendet werden kann[2]. Der Testprozess weist dem Eingabesignal jeweils nach einer Verzögerung von 20 ns einen neuen Wert zu und beendet sich mit der letzten Warteanweisung nach insgesamt 140 ns. Simulation unter Linux:

- Über das Menü »Anwendungen« ▷ »Zubehör« ▷ »Terminal« ein Terminal starten,
- in das Verzeichnis »Aufg1/Sim_GHDL« wechseln

  ```
  cd Aufg1/Sim_GHDL
  ```

- und die Kommandos aus der »Exor2.sh« ausführen:

  ```
  ghdl -a EXOR2_Sim.vhdl                   # Analyse der Schaltungsbeschreibung
  ghdl -a Test_Exor2.vhdl                  # Analyse des Testrahmen
  ghdl -m Test_Exor2                       # Make (ausführbares Programm erzeugen)
  ghdl -r Test_Exor2 --wave=Test_Exor2.ghw # Simulationsstart
  gtkwave Test_Exor2.ghw Test_Exor2.sav    # Visualisierung der Signalverläufe
  ```

Unter Windows: Start. Im Suchfenster »cmd« eintragen. Suchen. Im sich öffnenden Kommandofenster auf Laufwerk »H« wechslen (»H« eintippen), mit »cd« ins Praktikumsverzeichnis und weiter in das Verzeichnis »...\Aufg1\Sim_GHDL« wechseln. Kommandos eingeben. Nach dem Start von GTKWAVE sind über die entsprechenden Menüs und Icons die darzustellenden Signale und das darzustellende Zeitfenster auszuwählen. Abbildung 1 zeigt die mit den Einstellungen in der Datei »Aufg1/Sim_GHDL/Test_Exor2.sav« dargestellten Signalverläufe.
Statt die Kommandos abzutippen können unter Linux diese auch aus der Datei in die Konsole kopiert oder mit dem Skript-Aufruf

```
./Exor2.sh
```

ausgeführt werden. Unter Windows ist die Datei

```
Exor2.bat
```

zu starten.

[2]Bei einer Instanziierung ohne Beschreibungsnamen wird immer die zuletzt analysierte Beschreibung der Entwurfseinheit eingebunden.

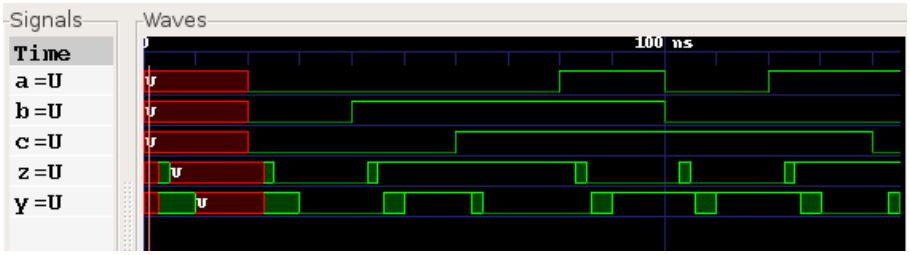


Abbildung 1: Simulationsergebnis

2.1 Schaltungsentwurf mit ISE

Für die Synthese ist die Zielfunktion synthesefähig, d.h. ohne Zuweisungen ungültiger Signalwerte, ohne Textausgaben und ohne Verzögerungszeiten, zu beschreiben. Die beiden Signalzuweisungen aus der Simulationsbeschreibung vereinfachen sich wie folgt (siehe Synthesebeschreibung »Aufg1/Sim_GHDL/Exor2_Synth.vhdl«):

```
z <= a xor b;
y <= z xor c;
```

Zur Simulation der vereinfachten Verhaltensbeschreibung ist in der Kommandodatei »Exor2.sh« bzw. »Exor2.bat[3]« der Dateinamen »Exor2_Sim.vhdl« gegen »Exor2_Synth.vhdl« auszutauschen.

Für den Entwurf mit »ISE« ist das Entwurfssystem zu starten. Unter Linux:

- Menü: »Anwendungen« ▷ »Umgebung« ▷ »Xilinx ISE 13«

und unter Windows:

- »Start« ▷ »All Programs« ▷ »Xilinx Design Tools« ▷ »ISE Design Suite 14.6« ▷ »ISE Design Tools« ▷ »64-bit Project Navigator«

In dem sich öffnenden Projektnavigator ist über das Menü

- »File« ▷ »Open Project«
- Wechsel zum Verzeichnis: »Aufg1/Exor2«

die Datei »Exor2.xise« auszuwählen und zu öffnen. Im Fenster »Hierachy« sollten bei der Auswahl von »Implementation« hinter »View« jetzt der zu programmierende Schaltkreises »xc6slx16-3csg324«, die Datei mit der Synthesebeschreibung »Exor2_Synth.vhdl« und die ucf-Datei »Exor2.ucf« in der in Abb. 2 a dargestellten Form stehen. Im Prozessfenster darunter werden für das oben ausgewählte Entwurfsobjekt alle anwendbaren Operationen angezeigt. Für die in der Abbildung ausgewählte Synthesebeschreibung sind das u.a. Synthese, Implementierung und die Generierung der Progammierdatei. Ein Doppelklick auf den zu programmierenden Schaltkreis darüber öffnet das Menü in Abb. 2 b. In diesem Menü kann nachträglich der Schaltkreistyp geändert, ein anderer Simulator ausgewählt werden etc.. In dieser und in allen folgenden Praktikumsaufgaben müssen genau die Werte wie in der Abbildung eingestellt sein[4].

Im ersten Entwurfsschritt soll die Register-Transfer-Synthese ausgeführt werden:

- Auswahl der Synthesebeschreibung »Exor2_Synth.vhdl« und
- Anklicken »Synthesize - XST« ▷ »View RTL Schematic«

Im sich öffnenden Fenster »Start with a schematic ...« auswählen und »ok«. Nach Doppelclick in das dargestellte XOR und zurecht-zoomen wird die synthetisierte Schaltung in Abb. 3 angezeigt, ein Exor mit drei Eingängen[5].

[3] »Rechte Maustaste« ▷ »Edit with Notepad++«.
[4] Die Einstellungen stammen aus der Projektdatei Exor2.xise und Änderungen werden dort gespeichert.
[5] Die graphische Schaltungsdarstellung der Syntheseergebnisse sind gewöhnungsbedürftig.

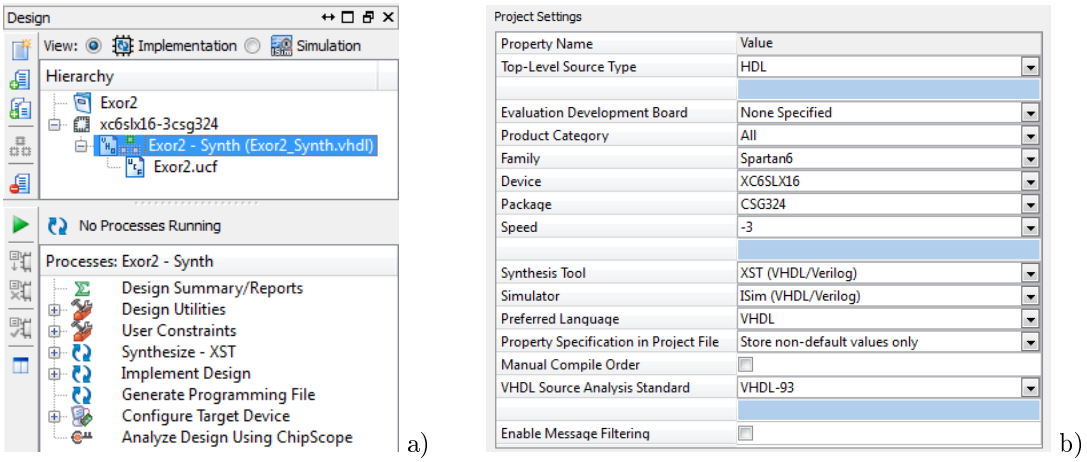


Abbildung 2: a) Fenster mit den Entwurfsquellen für die Implementierung und den anwendbaren Operationen b) Projekteinstellungen

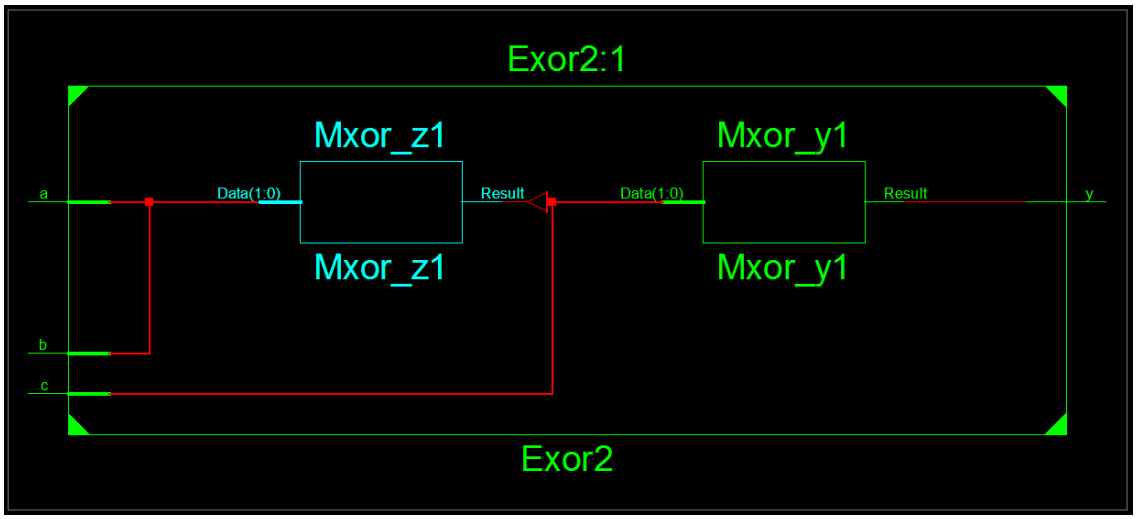


Abbildung 3: Ergebnis der Register-Transfer-Synthese

Nach der Synthese folgt die Technologieabbildung. Dazu ist der Prozess

- »Synthesize - XST« ▷ »View Technology Schematic«

zu starten. Wieder »Start with a schematic ...« auswählen und »ok«. Für die Nachbildung der Beispielschaltung genügt eine einzelne Tabellenfunktion[6]. An den Anschlüssen werden Buffer zur Signalwandlung zwischen den schaltkreisinternen und den Anschlusspegeln eingefügt. Mit einem (Doppel-) Klick auf den Funktionsbaustein lässt sich zusätzlich der zu programmierende Tabelleninhalt im Bild rechts unten anzeigen (Abb. 4).

Zur Platzierung und Verdrahtung benötigt das Projekt zusätzlich eine ucf-Datei (ucf – **u**ser **c**onstraints **f**ile). Darin stehen die Zuordnungen der Schaltungsanschlüsse zu den Schaltkreisanschlüssen und optional weitere Randbedingungen für die Synthese, die Platzierung und die Verdrahtung. Zur Bearbeitung der ucf-Datei ist diese im Fenster »Sources for Implementation«

[6] In unserem programmierbaren Logikschaltkreis werden kombinatorische Schaltungen hauptsächlich aus Tabellenfunktionen mit vier Eingängen und einem Ausgang nachgebildet. Für eine Tabellenfunktion mit drei Eingängen wird ein Eingang weniger genutzt.

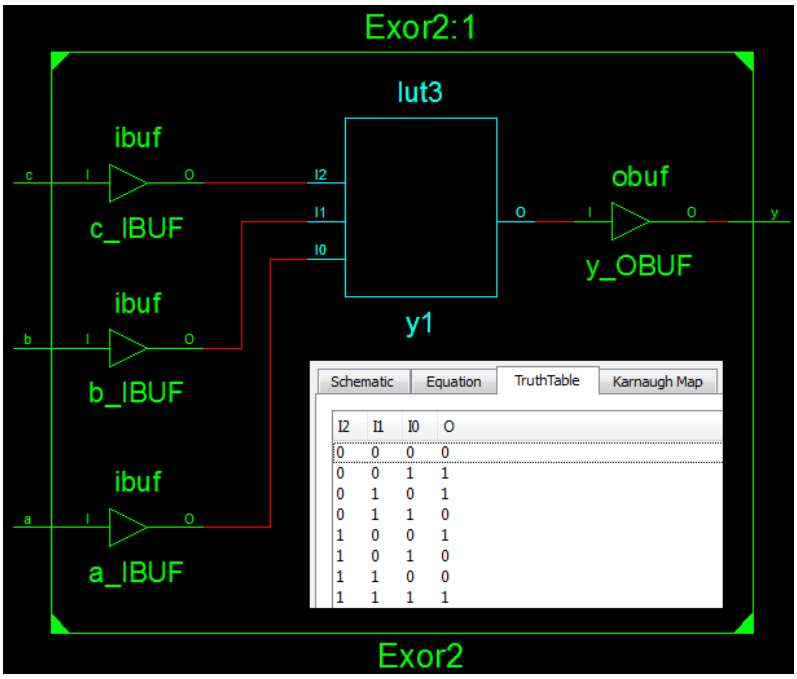


Abbildung 4: Schaltung nach der Technologieabbildung und Inhalt der Tabellenfunktion

auszuwählen und im Prozessfenster »User Constraints« ▷ »Edit Constraints (Text)« zu starten. Die in der vorgegebenen ucf-Datei enthaltenen Einträge

```
NET "a" loc="T10"; # Schalter SW0
NET "b" loc="T9";  # Schalter SW1
NET "c" loc="V9";  # Schalter SW2
NET "y" loc="U16"; # Leuchtdiode LD0
```

bedeuten, dass das Anschlusssignal »a« mit den Gehäuseanschluss »T10« etc. zu verbinden ist. Die Gehäuseanschlüsse stehen auf der Baugruppe neben den Schaltern und Leuchtdioden. Zur Generierung der Programmierdatei ist wieder die Synthesebeschreibung auszuwählen und

- »Generate Programming File« zu starten.

Zur Programmierung ist

- »Configure Target Device« ▷ »Manage Configuration Project (iMPACT)«

zustarten. iMPACT ist ein eigenes Programm. Die Programmierung erfordert folgende Schritte:

- Spannungsversorgung und Programmierkabel an die Baugruppe anschließen, anschalten,
- in iMPACT »File« ▷ »New Project« starten,
- die Frage »Do you want the system to automatically ...« mit »Yes« beantworten,
- im nächsten Fenster »Configure device using Boundary Scan« und »Automatically connect ...« auswählen,
- im nächsten Fenster »... assign configuration file(s)« mit »Yes« bestätigen,
- dem programmierbaren Logikschaltkreis die Datei »Aufg1/exor2.bit« zuordnen und »Open«
- im Fenster »...SPI or BPI PROM to this device?« mit »No« beantworten und

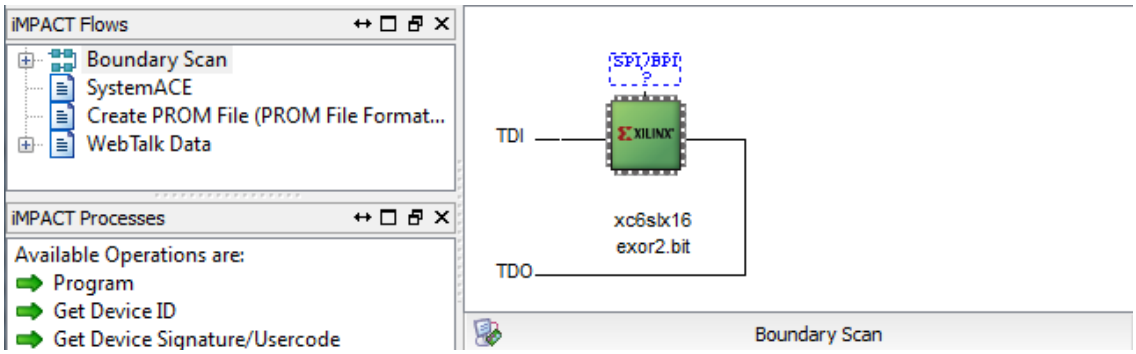


Abbildung 5: Programmieroberfläche nach der Konfiguration

- das nächste Fenster mit »OK« schließen.

Nach der Configuration sollte das Fenster wie in Abb. 5 aussehen. Boundary-Scan ist ein serieller Test- und Konfigurationsbus mit den Signalen TDI (**t**est **d**ata **i**n), TDO (**t**est **d**ata **o**ut), TCK (Testtakt) und TMS (**t**est **m**ode **s**elect, siehe Testbusstecker auf der Baugruppe). Mehrere Schaltkreise mit Testbus auf der Baugruppe bilden eine Kette. Jeder Schaltkreis hat eine auslesbare Hersteller- und Produkt-Identifikationsnummer, die mit

- »Get Device ID«

im Fenster »iMPACT Processes« gelesen werden kann. Mit

- »Program«

im selben Fenster erfolgt die Programmierung. Nach der Programmierung sollte die Erfolgsmeldung »Program Succeeded« angezeigt werden. Nach der erfolgreichen Programmierung ist die Schaltung bereit für den Test.

2.2 Test

Zum Testen der Beispielschaltung aus zwei EXOR-Gattern sind nacheinander alle acht möglichen Kombinationen der Schalterstellungen von SW0 bis SW2 einzustellen und zu kontrollieren, dass die Leuchtdiode nur dann leuchtet, wenn die Anzahl der eingeschalteten Schalter ungerade ist.

2.3 Manuelle Platzierung

In der Regel erfolgt die Platzierung und Verdrahtung automatisch. Falls erforderlich kann die Platzierung vor der Erzeugung der Programmierdatei mit »Implement Design« → »Place & Route« → »Analyze Timinig / Floorplan Design (PlanAhead)« manuell nachgebessert werden. Nach dem Start zwei aufgehende Fenster wegklicken. Die Programmoberfläche von »Plan Ahead« kann u.a. alle programmierbaren Hardware-Bausteine und deren geometrische Anordnung auf dem Silizium anzeigen. Unser Schaltkreis besitzt 2278 Slices, 32 konfigurierbare Blockspeicher, 32 Multiplizierer und 232 programmierbare Ein-/Ausgabeschaltungen. Ein Slice umfasst 4 LUTs mit 6 Eingängen, 8 Flipflops und zum Teil einige zusätzliche Gatter und Multiplexer für spezielle Aufgaben[7]. Abbildung 6 zeigt einen kleinen Ausschnitt mit der Tabellenfunktion so angeordnet, dass der Weg zu allen Anschlüssen minimal ist.

[7] z.B. die schnelle Übertragsweiterleitung in Addierern

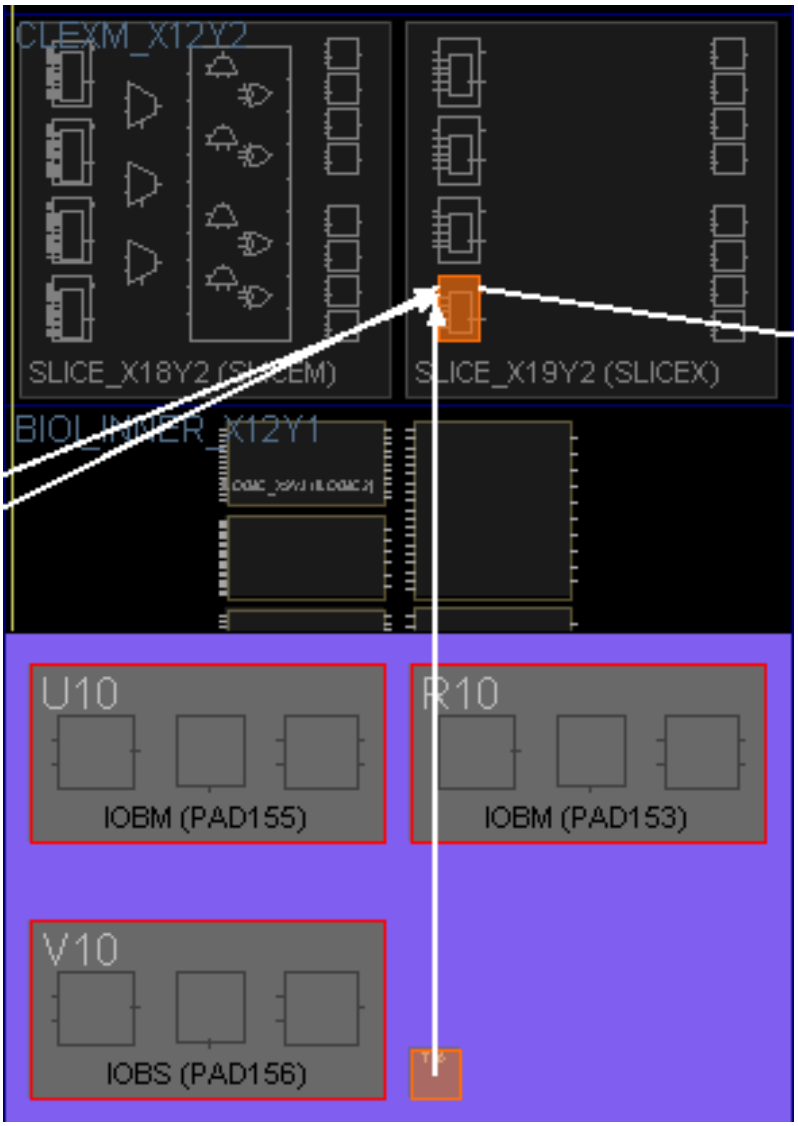


Abbildung 6: Platzierung mit PlanAhead

2.4 Laufzeitanalyse

Mit »Implement Design« → »Place & Route« → »Generate Post Place and Route Static Timing« → »Analyze Post-Place & Route Static Timing« lassen sich die mit der Laufzeitanalyse errechneten Pfadverzögerungen anzeigen, für die Beispielschaltung sind die Verzögerungen zwischen den Anschlüssen ähnlich wie in der nachfolgenden Tabelle:

```
All values displayed in nanoseconds (ns)
Pad to Pad
---------------+---------------+---------+
Source Pad     |Destination Pad|  Delay  |
---------------+---------------+---------+
a              |y              |    7.287|
b              |y              |    7.527|
c              |y              |    7.457|
---------------+---------------+---------+
```

2.5 Simulation unter ISE

Über dem linken Fenster mit den Entwurfsobjekten kann bei »View:« gewählt werden zwischen »Implementation« und »Simulation«. Bei Auswahl von »Simulation« ist in dem Auswahlfeld darunter die Simulationsart auszuwählen. Zuerst soll wie in Abbildung 7 links Behavioral Simulation« gewählt werden. In dieser Simulationsart wird das Verhalten ohne Verzögerungen, simulieren. Nach Auswahl »Behavioral« ist ein Testrahmen hinzuzufügen. Im vorbereiteten Projekt ist das die bereits eingefügte Datei »Test_Exor2.vhdl« im Verzeichnis »../Sim_GHDL«. Sonst

- Rechtsklick im Fenster »Hierarchy« bei den Dateien ▷ »Add Source«
- zum Verzeichnis »../Sim_GHDL« wechseln und
- die Datei »Test_Exor2.vhdl«.

auswählen und »Open«. Das Quellen- und Prozess-Fenster muss danach wie in Abb. 7 links aussehen. Im Quellenfenster ist »Test_Exor2« auszuwählen und im Prozess-Fenster »Simulate Behavioral Model« zu starten. Nach der richtigen Zoom-Einstellung wird das Simulationsergebnis in Abb. 7 rechts angezeigt.

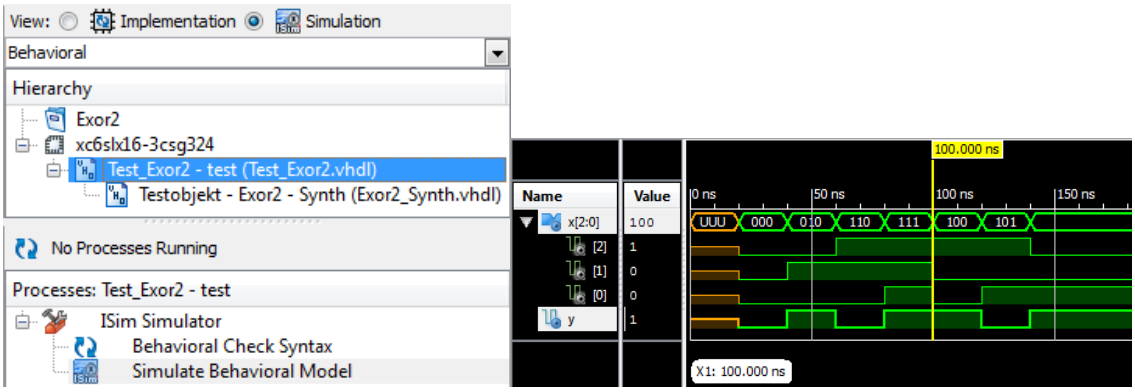


Abbildung 7: Simulation der Synthesebeschreibung

Zur Simulation der fertig synthetisierten, platzierten und verdrahteten Schaltung mit den echten Verzögerungszeiten ist zuerst das Simulationsmodell zu erzeugen. Dazu ist

- in die Darstellungsebene »View« auf »Implementation« zurückzuwechseln
- über »Implement Design« → »Place & Route« → »Generate Post-Place and Route Simulation Modell« mit der rechten Maustaste »Process Properties« für »Simulation Model Target« den Wert »VHDL« auswählen
- mit der linken Maustaste auf »Implement Design« → »Place & Route« → Doppelklick auf »Generate Post-Place and Route Simulation Modell« das Simulationsmodell erzeugen.

Um das Simulationsmodell sichtbar zu machen, ist unter »View« »Simulation« und darunter »Post-Route« auszuwählen. Abbildung 8 links zeigt die dann angezeigte Entwurfshierarchie. Die erzeugte VHDL-Datei hat den Architekturnahmen »Structure« und steht in der automatisch generierten Datei »Exor2_timesim.vhd«. Der Testrahmen ist derselbe wie für »Behavioral Simulation«. Damit die generierte Beschreibung mit diesem Testrahmen simuliert werden kann, muss das Testobjekt ohne Architekturnahmen instanziiert sein. Zur Simulation wird im Fenster »Hierarchy« der Testrahmen und im Fenster darunter der Prozess »Simulate Post-Place ...« gestartet. Abbildung 8 rechts zeigt das Simulationsergebnis nach dem richtigen Einstellen des Zooms. Wie die Abbildung zeigt, berechnet die Simulation keine Gültigkeitsfenster, sondern nur Richtwerte für den Ausgabesignalverlauf.

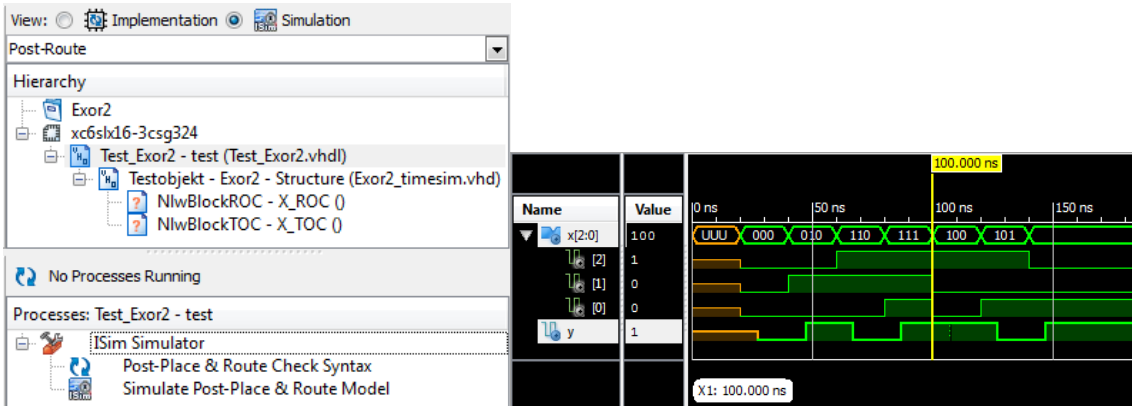


Abbildung 8: Simulation der fertig synthetisierten, platzierten und verdrahteten Schaltung mit den echten Verzögerungszeiten

3 Aufgaben

1. Entwerfen Sie eine Beispielschaltung für die Übung. Sie soll mindestens drei Eingänge haben und aus mindestens drei Gattern bestehen. Zeichnen Sie die Schaltung auf das Abgabeblatt und füllen Sie die Wertetabelle (Spalte »berechnet«) aus.

2. Schreiben Sie ein Simulationsmodell mit Halte- und Verzögerungszeiten sowie einen Testrahmen mit mindestens fünf verschiedenen Eingabewerten, simulieren Sie die Schaltung und füllen Sie in der Wertetabelle die zweite Spalte (»Simulation«) aus[8].

3. Schreiben Sie ein synthesefähiges Modell für die Beispielschaltung und eine ucf-Datei. Wählen Sie für die Anschlusszuordnung der Eingabesignale die Schalter und für die Ausgabesignale die Leuchtdioden. Die Anschlussnahmen stehen an den Schaltern und Leuchtdioden auf der Baugruppe (Leuchtdioden: U16, V16 etc., Schalter: T10, T9 etc.).

4. Programmieren und testen Sie die Schaltung mit Hilfe der Schalter und Leuchtdioden und füllen Sie die dritte Spalte in der Wertetabelle des Abgabeblatts aus.

Empfehlung (keine Pflichtaufgabe): Pro_Prakt1_VHDLbieren Sie auch die anderen beschrieben Entwurfsschritte (Visualisierung des Syntheseergebnisses, des Ergebnisses nach der Technologieabbildung etc. bis zur Simulation der fertigen Schaltung mit ihrer Beispielschaltung aus.

4 Fragen zur Selbstkontrolle

- Wozu dienen die Bibliotheken ieee und work?
- Was steht in dem Package ieee.std_logic_1164?
- Warum sollte am Ende der Anweisungsfolge im Testprozess ein »Wait« ohne Weckbedingung stehen?
- Warum lässt sich eine Schaltung mit exakt vorgegebenen Halte- und Verzögerungszeiten zwar simulieren, aber nicht synthetisieren? Hilfestellung: Welche Schaltungsmaßnahmen können Sie sich zur Anpassung der tatsächlichen Halte- und Verzögerungswerte an exakte Vorgabewerte vorstellen und erscheinen ihnen diese praktikabel und sinnvoll?
- Was ist zu erwarten, wenn die ucf-Datei fehlt oder falsche Anschlusszuordnungen enthält.

[8] Tragen Sie jeweils die stationären Werte, nachdem alle Signaländerungen abgeklungen sind, ein.

5 Abnahmekriterien

- Zeichnung und Wertetabelle auf dem Abgabeblatt
- Vorführung der Simulationsergebnisse
- Stichprobenweise Vorführung der Tests
- Stichprobenweise Beantwortung der Fragen zur Selbstkontrolle.

6 Aufräumen

Damit beim Ein- und Ausloggen nicht mehrere Minuten Daten transferiert werden, neu generierbaren Entwurfsdateien löschen:

`»Project« ▷ »Cleanup Project Files«`