# Einführung in Computer Microsystems



TECHNISCHE UNIVERSITÄT DARMSTADT

Prof. Dr. Andreas Koch Thorsten Wink

## 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.

🔤 New Project	t Wizard	
<b>Create New P</b> Specify pro	roject oject location and type.	
Enter a name	e, locations, and comment for the project	
<u>N</u> ame:	counter	
Location:	D:\counter\counter	<u></u>
<u>D</u> escription:		
- Select the typ Top-level sou HDL	pe of top-level source for the project urce type:	<b>→</b>
More Info		Next > Cancel

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*.

elect the device and design flow for t	the project	
Property Name	Value	
Product Category	All	~
Family	Spartan3E	~
Device	XC35500E	V
Package	FG320	~
Speed	-4	~
Top-Level Source Type	HDL	~
Synthesis Tool	XST (VHDL/Verilog)	Y
Simulator	ISim (VHDL/Verilog)	~
Preferred Language	Verilog	v
Manual Compile Order		
Enable Enhanced Design Summary		
Enable Message Filtering		
Display Incremental Messages		

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

🚾 New Source Wizard	
Select Source Type Select source type, file name and its location.	
<ul> <li>IP (CORE Generator &amp; Architecture Wizard)</li> <li>Schematic</li> <li>User Document</li> <li>Verilog Module</li> <li>Verilog Test Fixture</li> <li>VHDL Module</li> <li>VHDL Library</li> <li>VHDL Package</li> <li>VHDL Test Bench</li> </ul>	Eile name: counter.v Logation: D:\counter\counter 
More Info	Next > Cancel

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].

Define Mod Specify	<b>lule</b> ports for module.						
Module name	counter						
	Port Name	Direction	n	Bus	MSB	LSB	13
clk		input	~				
count		output	~		3	0	
		input	~				
		input	~				
		input	*				
		input	~				
		input	~				
		input	~				
		input	*				
		input	~				
		input	~				
		input	~				1

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.

ISE Project Navigator - D:\counter\cou	Inter\counter.xise - [counter.v]		
File Edit View Project Source Proces	ss Tools Window Help		- 8 ×
i 🗋 ờ 🖬 🕼 😓 i 🐰 🗅 🖨 🗙 🕪	a 🗛 🥬 🖉 🗶 🗶 🖉 🔊	🔁 🗄 🖽 🖬 🥕 🖋 🛛 🖗 🖾 🗶 🕴 🤇	7
Design     ↔ I Ø X       Sources for:     Implementation       Herardiv     ↓       Implementation     ♥       Implementation     ♥	<pre>1</pre>	(1:01:42 11/23/2009 )unter (le Created :s:	
		l ma	
Design Files Libraries	Counter.v	🕹 Design Summary	
Console Started : "Launching ISE Text Launching Design Summary/Repor	Editor to edit counter.v". rt Viewer		↔ □ & ×
Console Errors Warnings			In 24 Col 7. Verilos
🐮 Start 🔰 🧭 👄 🚧 💈 Arbei	itsplatz 🛛 🔤 ISE Project Navigator	👹 05.JPG - Paint	● 義 <b>孫</b> ● 兄/ (4) * (1) • (1) • (1)

## 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</pre>
```

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).

🚾 ISE Project Navigator - D:\counter\co	ounter\counter.xise - [counter.v]							
File Edit View Project Source Proc	ess Tools Window Help		20 No.	_ 8 ×				
: 🗋 🤌 🗐 🕼 😓 i 🗶 🗅 🖨 🗙 🕨	O & A PPXXB	💫 i 🗟 🗄 🖬 🖻 i 🌶 K?	i 🌐 🗵 🛠 i 💡					
Design ↔ □ ♂ ×	🛛 🐨 7 // Design Name:			~				
Sources for: Implementation	8 // Module Name:	counter						
(a) Hierarchy	9 // Project Name:							
Counter	- I // Target Device	18:						
💷 🚊 🔂 xc3s500e-4fg320	12 // Description:							
B Counter (counter.v)	· 三 13 //							
a	14 // Dependencies:							
	15 //							
63	17 // Revision 0.01	- File Created						
	18 // Additional Co	omments:						
-	74 19 //							
	20 ////////////////////////////////////		( , , , , , , , , , , , , , , , , , , ,	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,				
(Thu -	21 module counter(							
Processes: counter	23 output reg [	[3:0] count						
Design Summary/Reports	24 );							
B User Constraints	25							
👘 🖥 🚺 Synthesize - XST	26 initial cour	nt =0;						
View RTL Schematic	27 29 alwaye@/noes	dae alki						
View Technology Schematic	29 count <=co	ount+1;						
Check Syntax	30							
(a) Generate Post-Synthesis 5	31 endmodule							
Generate Programming File	32			~				
<	<	1	_					
Design Files Libraries	What's New in ISE Design Suite 11.1	🗵 Design Summary	counter.v					
Console				⇔⊡∄×				
No errors in compilation				^				
Analysis of file <"counter.pr	rj"> succeeded.							
Drocess "Chark Suntay" completed successfully								
				×				
Constant of the second								
console errors Warnings				In 22 Col 15 United				
				Lin 22 Col 15 Verilog				
🥵 🌑 🕫 👔 🕫	JPG - Paint 📁 🛅 ise	ISE Project Navigator		🖲 🐺 🐼 🔍 😓 🔊 🕲 12:36				

#### **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*.

🔤 ISE Pr	roject Navigato	- D:\cour	nter\coun	iter\co	ounter	r.xise	e - [counter.v]							F	×
🖹 File 🛛	Edit View Proj	ect Source	Process	Tool	s Wi	ndow	Help							- 8	×
: 🗋 🆻		X D B	×	(1)	M) : .	Æ F	XXPD	20	38091	• <b>k?</b>   @	🗵 📌 🗄 💡				
Design Source G Hiera G C C C C C C C C C C C C C C C C C C C	ces for: Implementa archy Counter Cou	tion ation (counter.v) Add: Add: Coper Add: Coper Add: Set at Smart X	Source Source Copy of Sou Hal Compile I is Top Modu tGuide	urce	7 8 9 10 11 12 13 14 15 16 7 8 9 0 1 1 2 3 4 4 5 6 7	// // // // // // // // // // // // //	Design Name: Module Name: Module Name: Target Device Target Device Description: Dependencies: Revision 0.0: Additional Cc Additional Cc (input clk, output reg ); initial courter	co es: s: L - Fi comment (3:0] nt =0;	le Created s: ///////////////////////////////////						<ul> <li>In the second sec</li></ul>
	View Tec Color Check St Color Check St Check St	hr Delei /n Delei ei Parti es 🗿 Remi	Partition te Partition tion Force		8 9 0 1 2	end	always@(pose count <=co dmodule	edge c ount+1	1k) ;						>
<	Car denerate Ph	V File N	lamec				)							>	
Design	Files Libraries	✓ Displ	ay Full Path	5	s N	ew in IS	SE Design Suite 11.1	E	Design Summary		counter.	v			
Console		Desir	n Propertie	s										+□6	×
No e: Anal Proc	rrors in com ysis of file ess "Check S	yntax" (	ce Propertie	ed si	lee	ded. sful	.ly								
<														>	Ê
Console	Errors Warr	ings													
_		_			_	_		_				_	Ln 16 Co	7 Ver	llog
🔧 Sta	art 🖉 🙆 🤅	9 en	👸 07.JPG	5 - Paint	ŝ.		ise 🔁		ISE Project Na	vigator			1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	12:4	3

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.

🔤 ISE Project Navigator - D: \counter\cou	Inter\counter.xis	e - [test_counter.v	<b>^</b> ]	
File Edit View Project Source Proces	ss Tools Window	Help		_ 8 ×
i 🗋 🤌 🗟 🕼 😓 i 🔏 🗅 🖄 🛏	A 10	XXPD	🚬 🖻 🗄 🖬 🖬 🖉 🥬 🖉 🦧 📔	♥
Design     → □ @ X       Sources for:     Implementation       Implementation     Implementation	Image: Constraint of the second sec	<pre>dule test_coun // Inputs reg clk; // Outputs wire [3:0] co // Instantiat counter uut ( .clk(clk), .count(cou ); initial begin</pre>	unt; e the Unit Under Test (UUT) nt)	**** •
약값 No single design module is selected. ♥값 ಈ 🎾 Design Utilities ♥값	40 41 42 43 44 45 46 47 48 49 50 en	<pre>// Initial clk = 0; // Wait 10 #100; // Add stip end dmodule</pre>	ize Inputs O ns for global reset to finish mulus here	
Design Files Libraries	🗾 What's New in :	ISE Design Suite 11.1	📓 Design Summary (out of date) 🖹 counter.v	test_counter.v*
Console				↔□♂×
tdtfi(verilog) completed succ Process "Create Verilog Test : Started : "Launching ISE Text	essfully. Fixture" comp. Editor to ed:	leted successfi it test_counter	ully r.v".	
				*
<				
Console Errors Warnings				Lo 49 Col 1 Veriloa
Chart Chart	oc point	[ <b>P</b>		
		<b>153</b>	ISE Project Navigator	V 20 0 5 00 0 12:19

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.

ISE Project Navigator - D:\counter\counter	Ncounter.xise - [t	test counter.vl		
File Edit View Project Source Process	Tools Window He	k i		_ = = ×
∞ × □ □ X   0   0   0   0   0	M P P >	(x / 2 💫 🗟 🖬 🖬 🗣 🖊 🕴	🖡 🗵 🛠 🗄 💡	
Design ↔ □ ♂ × 📲	24	***************************************		~
Sources for: Behavioral Simulation	25 module	test_counter;		
(i) Hierarchy	26	The second se		
🚰 🕂 🔄 counter	27 // 28 rec	inputs		
🛄 🖻 🛄 xc3s500e-4fg320	29			
	30 //	Outputs		
	31 W11	re [3:0] count;		
an i A	33 //	Instantiate the Unit Under Test (UUT	1	
cy.	34 001	unter uut (	M.	
2 · · · · · · · · · · · · · · · · · · ·	3.5	.clk(clk),		
	36	.count (count)		
< >>> >>	38			
Processes: test_counter	39 in:	itial begin		
are in Simulator	40	// Initialize Inputs		
Behavioral Check Syntax	41	clk = 0;		
💥 🔤 🔤 Simulate Behavioral Model	42	// Add stimulus here		=
-	44 end	4		
	45	always		
	46	#10 clk = ~clk;		
	47 endmod	aule		
	40			~
	<			>
Design Files Libraries	What's New in ISE De	esign Suite 11.1 🛛 🗕 Design Summary (out of date) 🗍 🗎	counter.v 🖹 test_c	counter.v
Console				↔ ⊡ ♂ ×
tdtfi(verilog) completed success	fully.			^
Process "Creace Verilog lest Fix"	cure. compiece	a successfully		
Started : "Launching ISE Text Ed:	itor to edit t	est counter.v".		1231
		_		
<				>
Console Errors Warnings				
				Ln 46 Col 11 Verilog
Start @ @ m 2 N In PS-P	2aint 📁	Ist		

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.

🏭 lSim - [Default.wcfg*]					🗖 🗗 💟
🔕 File Edit View Simulation Windo	w Help				_ 8 ×
: 🗋 🦻 🖬 😓 : 🐰 🗈 🗅 🗙 🔇	) 🗠 🖂 🔚	8 0 9 P N? .	P P X P 2	***	1.00us 💌 🔙 🝴 💷
Instances and Processes ↔ □ 문 ×	Objects	↔□륜× 🦽	Zoom Out		1
	Simulation Objects for te	st_counter			New York Control of the State o
	- 14 14 14 14 14 14 14	16 😧 🙀	Name Value	0 ps 50 000 ps	100 000 ps 150 000 ps
Instance and Process Name	Object Name	Value	► Count 0010	(X 0001 X 0010 X 0011 X 0100 X	<u>111 X 0110 X 0111 X 1000 X 1</u>
▷ albi	⊳ 🏹 count	0010	i cik U		
	🚺 🗓 dk	0			
	-				
		14			
		3			
		য়া			
	1				
				X1: 1 000 000 ps	
< >			< > < >	<	> ~
Instances and Processes Source Files	<	<b>&gt;</b>	Default.w	cfg*	
Console			******		+□8>
WARNING: A WEBPACK license was found.					
WARNING: Please use Xilinx License Configura	tion Manager to check out a	i full ISim license. on for more information on th	a differences between the	Lite and the Full version	
This is a Lite version of ISim.	aren to the taini documentati	on for more information on th	e un crences beciveen che	Lice and the Full version.	
Time resolution is 1 ps					
Finished circuit initialization process.					
ISim>					
Console Breakpoints Search Results					
Zoom out such that objects become smaller					Sim Time : 1000 ns
📲 Start 🔰 🙆 🍩 🕫 👔	11.JPG - Paint	😂 ise	ISE Project Navig	ato 🌆 ISim - [Default.wcfg*]	14:00

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.