Basispraktikum Hardwarenaher Systementwurf
WS 11/12
Arbeitsbericht 2

Sarah Lutteropp, Parto Karwat

22. November 2011

1 Vorwort

In diesem Aufgabenblatt ging es darum, Kenntnisse über den Umgang mit FPGAs zu erlangen und sich mit Xlinx vertraut zu machen sowie einen Überblick über verschiedene Hardwarebeschreibungssprachen zu erlangen.

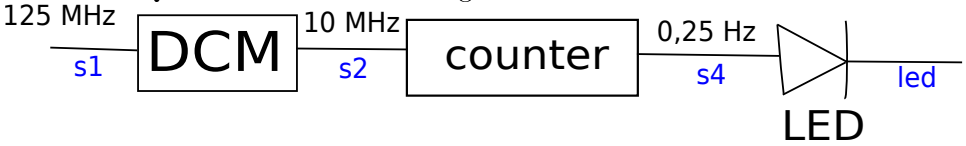
2 Xilinx ISE

Wie haben in Xilinx unser aus dem letzten Aufgabenblatt erstelltes my-nand.vhd importiert, die Pins zugeordnet und den Bitstream generiert. Anschließend haben wir erfolgreich den Bitstream auf die FPGA-Karte gespielt und unsere NAND-Schaltung mithilfe der LED und 2 Schaltern getestet. Siehe Source.

3 LED-Test

3.1 Gesamtschaltung

Hier ein Überblick über unsere Gesamtschaltung. Die blauen Werte stehen für die im Quelltext verwendeten Signale.



3.2 Digital Clock Manager (DCM)

Wir haben die DCM so eingestellt, dass wir eine Ausgangsfrequenz von 10 MHz erhalten. (CLKFX ausgewählt, $M = 2$, $D = 25$)

3.3 Zähler

Wurde eingefügt. (Siehe angefügte Dateien 'counter.vhd' 'ledTest.vhd' und 'ledTest_tb.vhd') Den Wert $ANZAHL$, den der Zähler hochzählt, haben wir folgendermaßen bestimmt:

$$f = \frac{1}{T}, \quad T = 4s, f_{LED} = 0.25Hz, f_{DCM} = 10Mhz = 10 \cdot 10^6 Hz$$

$$f_{LED} = \frac{f_{DCM}}{2 \cdot ANZAHL}$$
$$\Rightarrow ANZAHL = 2 \cdot 10^7 = 20000000$$

4 Synthese

Siehe angefügte Datei 'in1_out1_myLEDtest.ucf'. Anschließend haben wir den Bitstream generiert, auf die FPGA-Karte übertragen. Die Blinkzeit der LED betrug, wie erwartet, zwei Sekunden.

5 Hardwarebeschreibungssprachen

	Verilog	**VHDL**	**SystemC**	**ParC**	**JHDL**	**Lola**
Einsatz	Spezifikation und Entwurf kompletter Systeme	Spezifikation und Entwurf kompletter Systeme	noch nicht in großen wirtschaftlichen Projekten eingesetzt	Multicore und verteilte Systeme, RF/Wireless, Neuronale Netze	selbstkonfigurierende Systeme	Synchrone, digitale Schaltkreise
basierend auf	C	ADA	C++	C++	Java	PASCAL, Niklaus Wirth
Vorteile	leichter erlernbar als VHDL, schnelle, simple Umsetzung, Einflussmöglichkeiten auf der Gatterebene (UDP), keine strenge Typprüfung	großer Sprachschatz, höher angesiedelt (bis zur Systemebene), starke Typisierung, Wiederverwendbarkeit, Vielseitigkeit,	schnelle Simulationen, nahtloser Top-Down-Entwurf	kann als Basis für KI verwendet werden (Neuronale Netze), kann dynamisch erzeugt werden	kostenlos, leicht konfigurierbar, leicht erweiterbar	einfach, leicht erlernbar
Nachteile	geringerer Sprachschatz als VHDL	schwerer erlernbar als Verilog, enthält Konstrukte, die sich nicht auf Hardwareebene realisieren lassen	syntaktischer Overhead, mangelnde Angebot an Syntheswerkzeugen, kein Powermanagement	geringe Verbreitung	JVM benötigt, auf Linux braucht man Wine zur Bitstreamgenerierung	keine Anwendung in der Industrie, geringer Sprachschatz

	Verilog	**VHDL**	**SystemC**	**ParC**	**JHDL**	**Lola**
Verbreitungsgrad	sehr hoch, weltweit, insbesondere USA	sehr hoch, bedeutendste HDL in Europa	De-facto-Standard im Bereich IP-Nutzung und System-Level-Spezifikation	gering	gering	nur in der Lehre (ETH Zürich), sehr gering
Synthese-programme	Xilinx, Icarus Verilog, Quartus II ...	Xilinx, Synplify Pro, ...	SystemC-Compiler von Synopsys	-	Xilinx	-
Sonstiges	ursprünglich Simulationssprache, seit 1995 IEEE-Standard, lange Zeit herstellerspezifisch	seit 1987 IEEE-Standard, von Beginn an als offener Standard entwickelt	Firmenunterstützung bei Weiterentwicklung, Spracherweiterung	Teil des V2000 open simulator project, 1:1 - Mapping, Spracherweiterung	Open-Source, Spracherweiterung	-
Entwurfsjahr	1983/84	1985 (erste kommerzielle Version)	1999	-	1997	1994