Praktikum Softprocessor SP1: Einführung

7. Juli 2014

Zusammenfassung

Es ist ein minimales Rechnersystem aus einem Softprozessor, einem kleinen Daten- und Befehlsspeicher, einer Schaltereingabe, einer Leuchtdiodenausgabe, einer seriellen Schittstelle für die Kommunikation mit dem Arbeitsplatzrechner und einer Debug-Schnittstelle aus IP-Cores zu konfigurieren und in C zu programmieren.

1 Hardware-Entwurf

Ein eingebettetes Rechnersystem besteht aus dem Prozessor, dem Speicher und peripheren Schnittstellen zur Kommunikation mit der Schaltungsumgebung. Abbildung 1 zeigt das Blockschaltbild des zu entwerfenden Rechnersystems. Der Kern ist ein 32-Bit-RISC-Prozessor vom Typ MicroBlaze. Das ist ein Prozessor, der speziell für die Implementierung in programmierbaren Logikschaltkreisen optimiert ist. An den Prozessor sind der Datenbus (DLMB), der Befehlsbus (ILMB) und der Peripheriebus (PLB) angeschlossen. Der Adress- und der Datenbus sind jeweils über einen Speicher-Controller an einen der beiden Ports des 2-Port-Speichers angeschlossen, der somit gleichzeitig als Daten- und Befehlsspeicher dient. An den Peripheriebus sind über entsprechend konfigurierte Schnittstellenschaltungen

- eine parallele Ausgabeschnittstelle zur Ansteuerung von acht Leuchtdioden,
- eine parallele Eingabeschnittstelle zum Einlesen von acht Schalterwerten,
- eine serielle Schnittstelle für die Kommunikation mit dem Arbeitsplatzrechner und

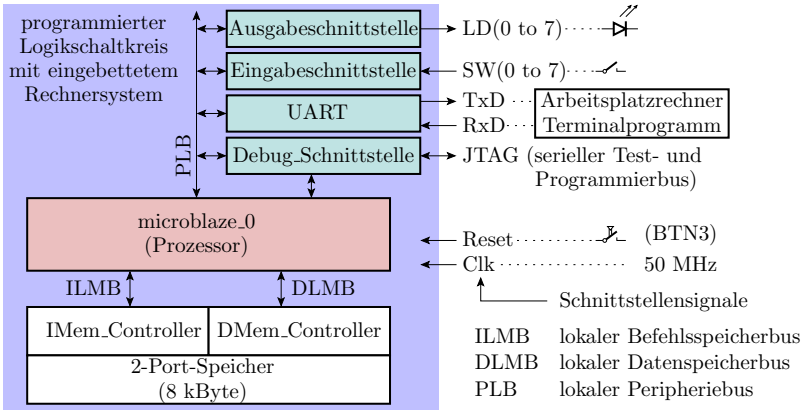


Abbildung 1: Das zu entwerfende Rechnersystem

- eine spezielle Debug-Schnittstelle für den Schrittbetrieb, zum Setzen von Haltepunkten etc..

angeschlossen. Ein eingebettetes Rechnersystem besitzt weder eine Tastatur, noch eine Festplatte noch einen Bildschirm. Das Programm wird wie die Hardware-Konfiguration über den seriellen Test- und Programmierbus in den Schaltkreis geladen. Die Benutzereingabe und -ausgabe erfolgt über die serielle Schnittstelle und die Kommunikation mit dem Debugger über den Test- und Programmierbus. Die Taste »BTN3« dient als Reset-Taste. Über die acht Schalter kann ein Byte eingegeben und über die acht Leuchtdioden ein Byte ausgegeben werden. Der Systemtakt beträgt 50 MHz und wird von IC4 auf der Unterseite der Versuchsbaugruppe bereitgestellt.

1.1 Anlegen eines neuen Projektes

- Programmstart unter Linux über das Menü:
  - Anwendungen ▷ Umgebung ▷ Xilinx Platformstudio (EDK)

  und unter Windows über das Menü:
  - Start ▷ All Programs ▷ Xilinx Design Tools ▷ Xilinx ISE Design Suite 14.6« ▷ »EDK« ▷ »Xilinx Platform Studio«
- Auswahl »Create New Blank Project« (Abb. 2 links). Weiter mit »OK«.
- Festlegung des Arbeitsverzeichnisses für das Projekt über »Project file« ▷ »Browse« und Auswahl des programmierbaren Logikschaltkreistyps, für die Praktikumsbaugruppen: »spartan3«, »xc3s1000«, »ft256« und »-4« (Abb. 2 rechts[1]).
- Weiter mit »OK«.

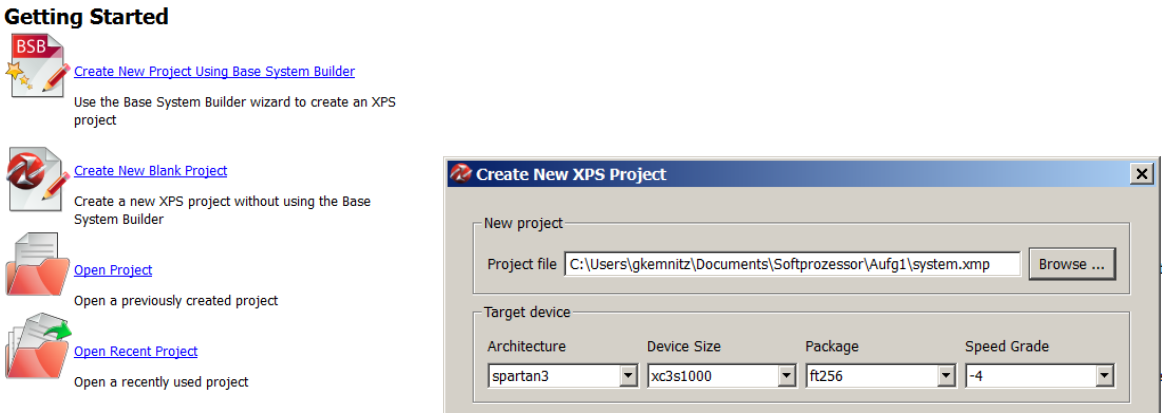


Abbildung 2: Anlegen eines neuen Projekts

[1] Achtung Eingabe-Bug: Alle Menü-Einträge anklicken und Werte korrigieren, auch wenn vor der Eingabe richtige Werte angezeigt werden.

1.2 Auswahl, Konfiguration und Verbindung der IP-Cores[2]

Auf der Oberfläche unter dem linken Fenster ist der Reiter »IP Catalog« auszuwählen. Die benötigten IP-Cores sind mit der Maus auf die rechte Seite zu ziehen. Die sich dabei öffnenden Menüs zur Konfiguration der Cores mit »ok« wegklicken. In Abb. 3 ist das bereits erfolgt. Die nachfolgende Tabelle zeigt für alle zehn Funktionsblöcke, unter welcher Kategorie sie zu finden sind, wie sie heißen und wie sie anschließend entsprechend Abb. 1 umzubenennen sind. Für den Prozessor ist der automatisch vergebene Name beizubehalten[3].

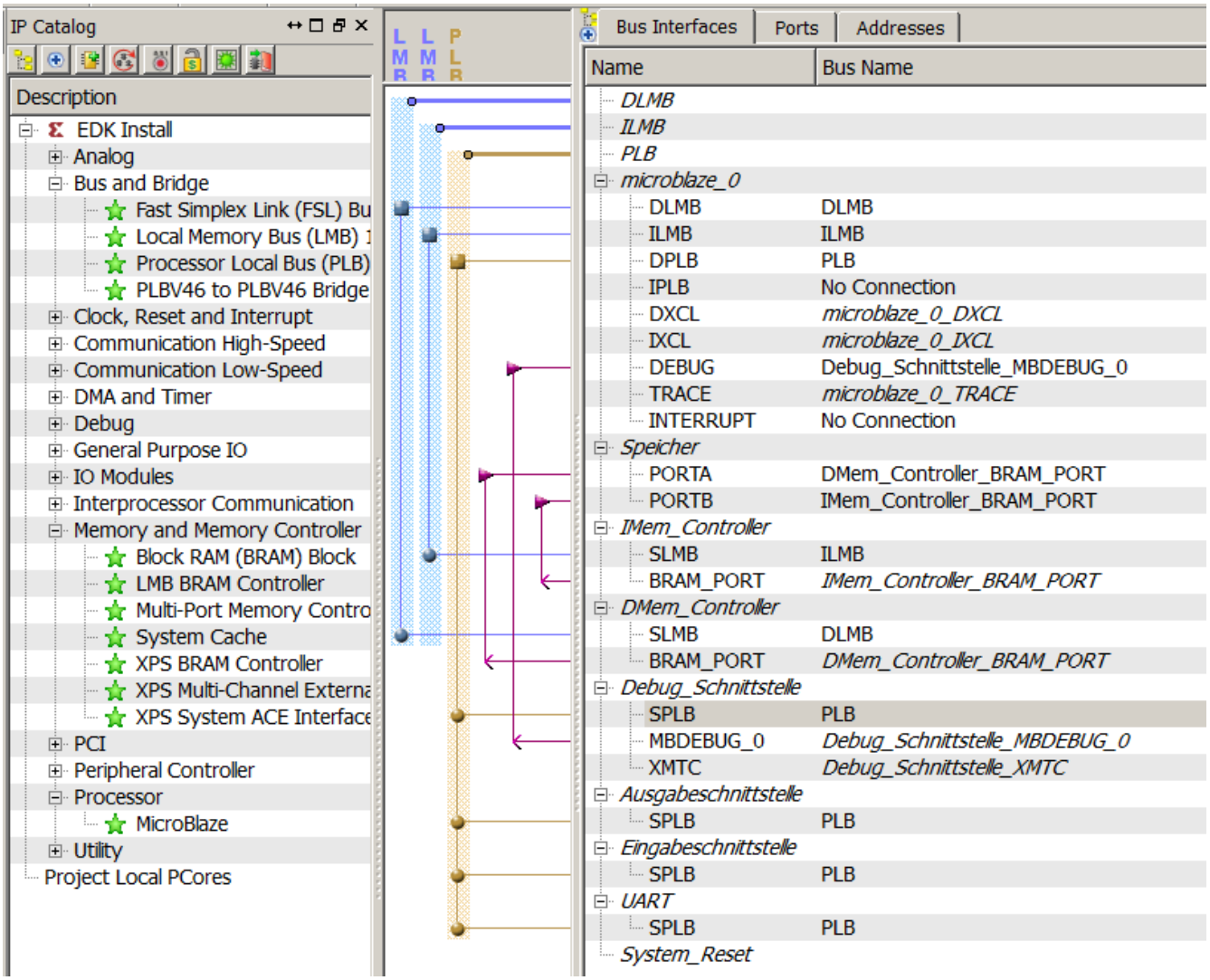


Abbildung 3: Auswahl der Funktionsblöcke aus dem IP-Core-Katalog und Umbenennung

[2]IP-Cores sind vorentworfene, konfigurierbare Schaltungsbeschreibungen. Die Abkürzung IP steht für »intellectual property«.

[3]Der Prozessorname wird zur Generierung von Bezeichner in den Header-Dateien verwendet. Bei Umbenennung gab es zumindest in älteren EDK/SDK-Versionen Fehler, dass Bezeichner beim compilieren und linken nicht gefunden wurden.

Kategorie	Bausteintyp	umbenennen in
Processor	MicroBlaze	microblaze_0
Bus and Bridge	Local Memory Bus (LMB)	DLMB
Bus and Bridge	Local Memory Bus (LMB)	ILMB
Bus and Bridge	Processor Local Bus (plb_v46)	PLB
Memory and Memory Contr.	LMB BRAM Controller	IMem_Controller
Memory and Memory Contr.	LMB BRAM Controller	DMem_Controller
Memory and Memory Contr.	Block RAM (BRAM)	Speicher
Debug	MicroBlaze Debug Module (MDM)	Debug_Schnittstelle
General Purpose IO	XPS General Purpose IO	Ausgabeschnittstelle
General Purpose IO	XPS General Purpose IO	Eingabeschnittstelle
Communication Low Speed	XPS UART (Lite)	UART
Clock, Reset and Interrupt	Processor System Reset Module	System_Reset

Für den Prozessor soll »Current Settings« beibehalten werden. Zur Umbenennung sind die automatisch vergebenen Namen mit der linken Maustaste auszuwählen und zu überschreiben. Als nächstes werden die Busverbindungen festgelegt. Vom Prozessor werden der Datenbus (DLMB), der Befehlsbus (ILMB) und der Peripherie-Bus (DPLB) mit den Systembussen wie in Abb. 4 durch Anklicken der Kreuzungspunkte in der Graphik verbunden. Die Speicher-Controller werden jeweils mit ihrem Speicherbus verbunden, jeder Port des 2-Port-Speichers mit den roten Busanschlüssen der Speicher-Controller und die drei Schnittstellenschaltungen mit dem Peripherie-Bus (PLB). Die Debug-Schnittstelle wird mit dem Peripherie-Bus und einem speziellen Debug-Port des Prozessors verbunden.

Die peripheren Einheiten müssen konfiguriert werden. Für die Ausgabeschnittstelle ist unter »Channel 1«, Reiter »User« die Datenbitbreite auf acht zu reduzieren (Abb. 5 a). Zusätzlich soll der »Channel 1 Data Default Value« so verändert werden, dass nach der Prozessorinitialisierung vier Leuchtdioden ein und vier Leuchtdioden aus sind. Für die Eingabeschnittstelle ist wie in Abb. 5 b die Datenbreite auf acht zu verringern und »Channel 1 is Input Only« auf »TRUE« zu setzen.

Für die UART (universal asyncronous receiver transmitter) können die Einstellungen im Menü »User« in Abb. 6 a beibehalten oder verändert werden. Wichtig ist, dass die Einstellungen mit denen des Terminalprogramms, über das die Kommunikation später mit dem PC erfolgt, übereinstimmen. Im Menü »System, PLB« Abb. 6 b, muss die Taktfrequenz auf »50000000Hz« (50 MHz) verringert werden[4].

An der Konfiguration des Debug-Ports ist nichts zu ändern, aber für den Prozessor ist über »Config IP« ▷ »Advanced« im Reiter »Debug« ein Haken bei »Enable Microblaze Debug Module Interface« zu setzen. Anderenfalls gibt es beim Programmstart unter SDK einen Fehler.

1.3 Externe Verdrahtung

Es ist zur Ansicht »Ports« zu wechseln (Abb. 7). Die gesamte Schaltung hat nach Abb. 1 einen Takteingang (Clk), eine Initialisierungseingang (Reset), einen seriellen Eingang (RxD), einen seriellen Ausgang (TxD), einen 8-Bit-Ausgabevektor »LD« für die Leuchtdioden und einen 8-Bit-Eingabevektor »SW« für die Schalter. Zuerst ist mit dem Button »Add External Port« (rechts oben in Abbildung 7) für jedes Anschlusssignal hinzu zu fügen und dann anzupassen. Die Beze-

[4] Zuerst das Handsymbol anklicken. Danach lässt sich auch der Frequenzwert editieren.

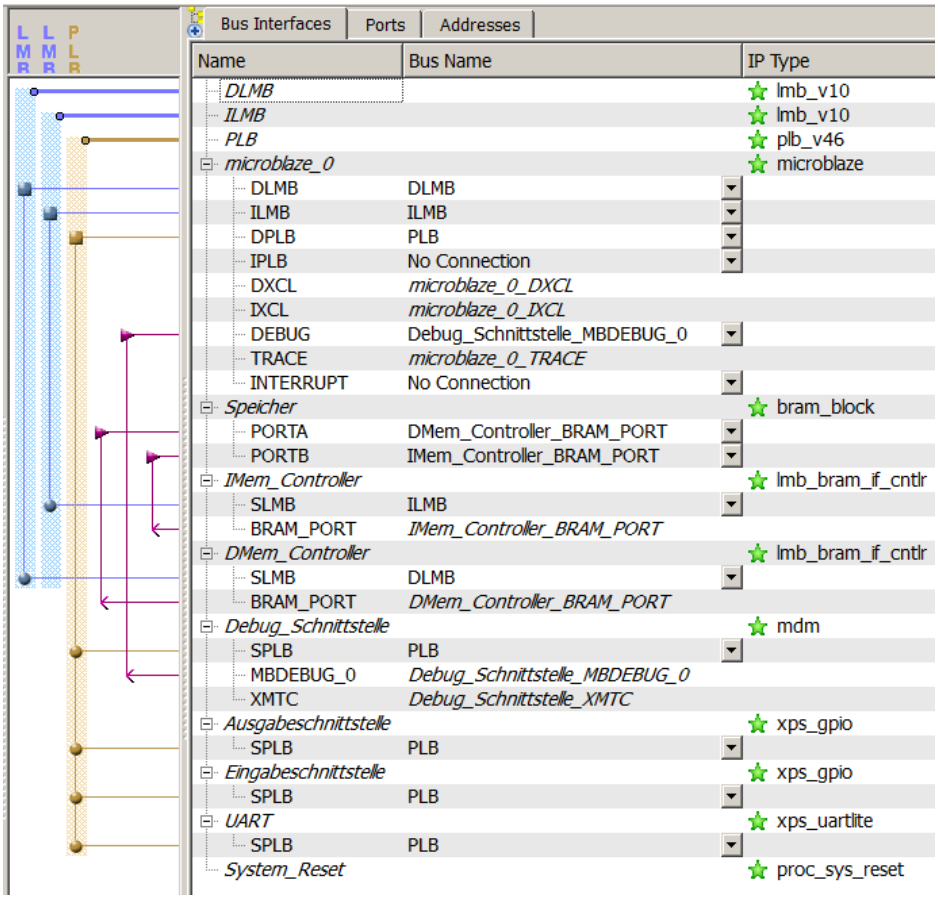
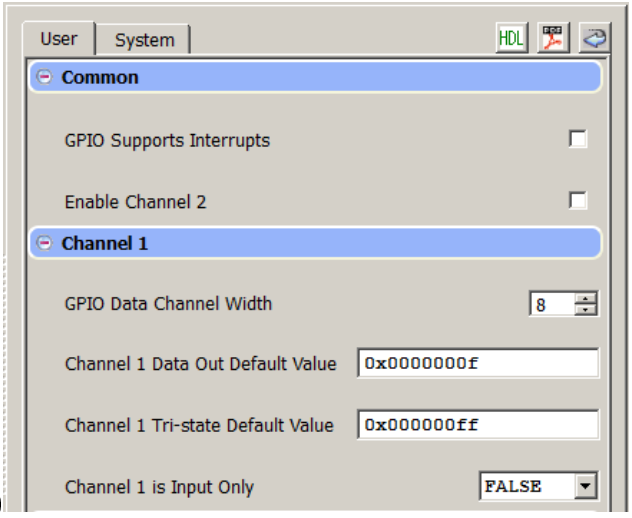


Abbildung 4: Busverbindungen

ichner in der Spalte »Name« müssen mit denen in der ucf-Datei übereinstimmen. In der Spalte »Net« ist der interne Bezeichner anzugeben, in der Regel derselbe wie für den Port. In der Spalte »Direction« bedeutet »I« Eingabe und »O« Ausgabe. Für die beiden Bitvektoren ist in der Spalte »Range« der Indexbereich anzugeben. Der Indexbereich der Daten ist absteigend sortiert. Takte

a)



b)

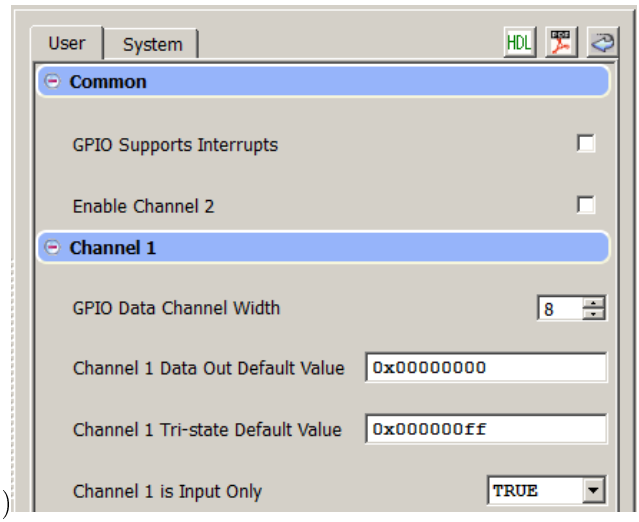
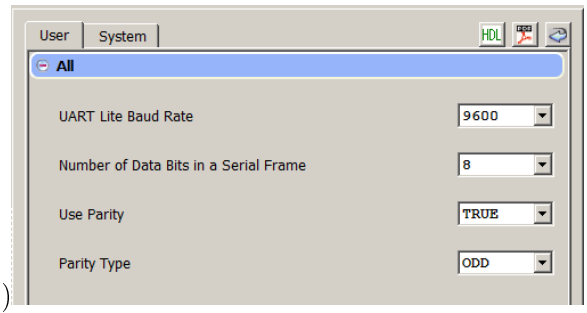


Abbildung 5: Konfiguration a) Ausgabeschnittstelle b) Eingabeschnittstelle

a)



b)

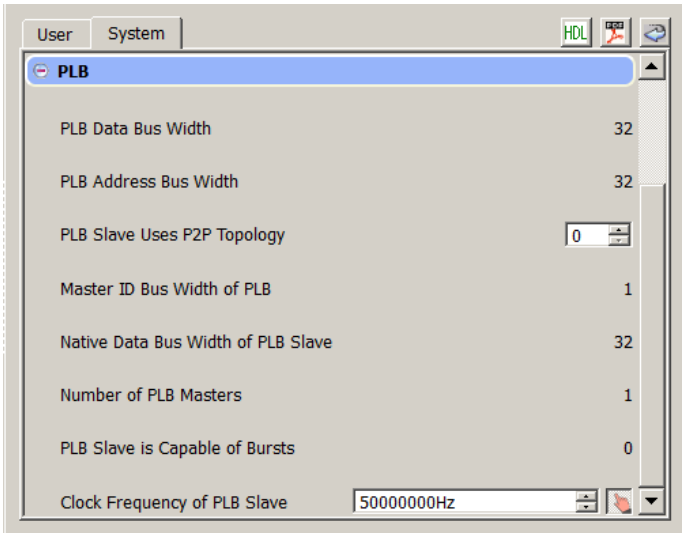


Abbildung 6: UART-Einstellungen a) Übertragungsrate und -format b) Taktfrequenz

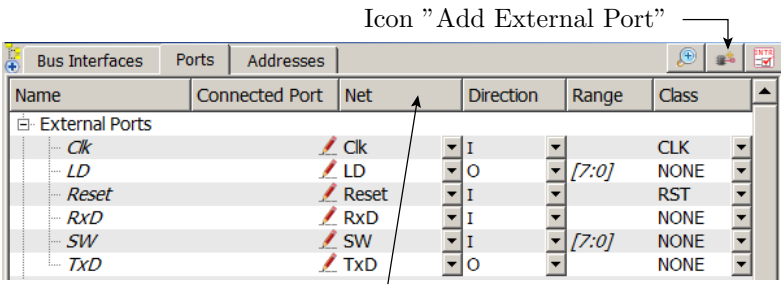
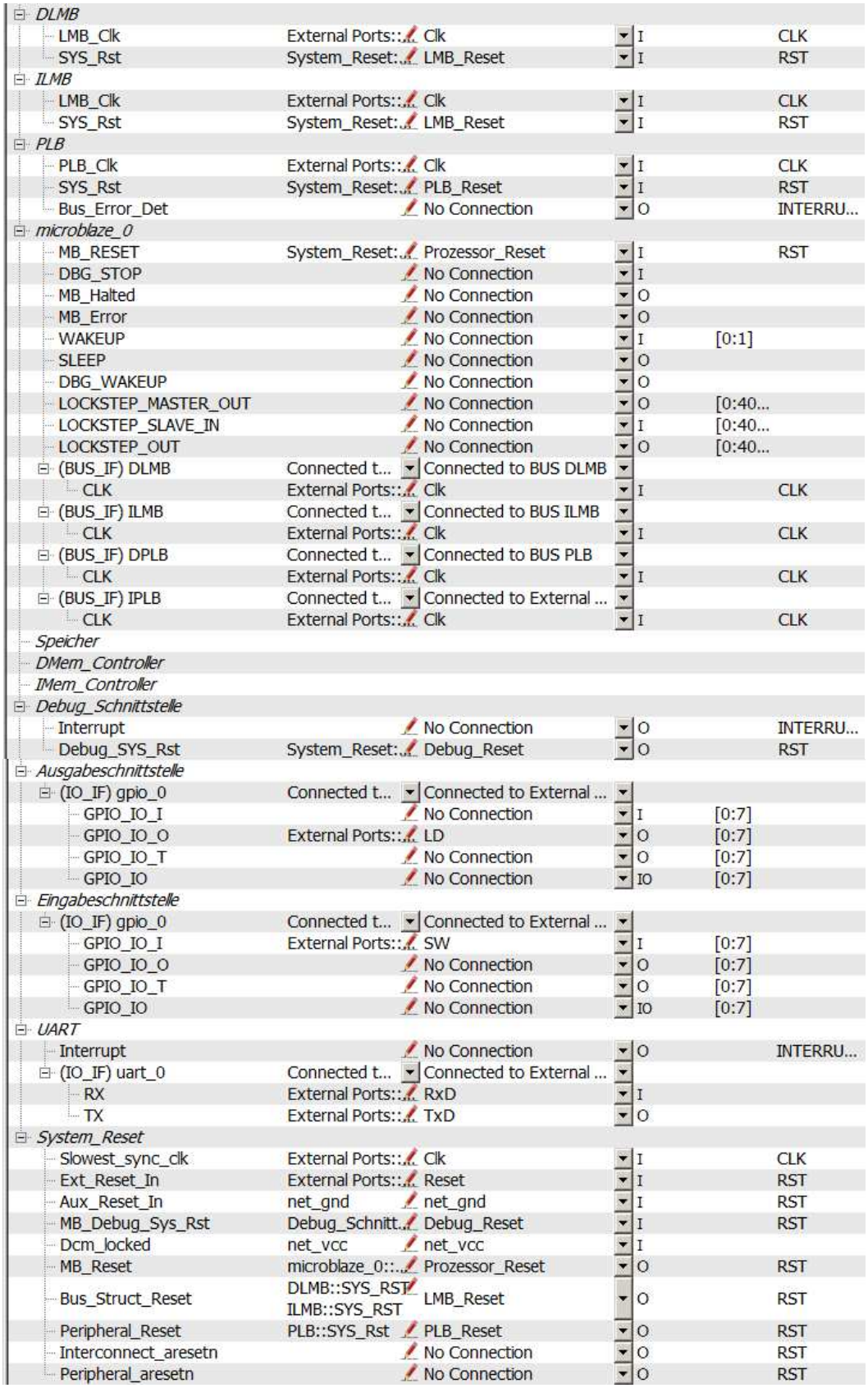


Abbildung 7: Externe Anschlüsse

und Initialisierungssignale werden in der Spalte »Class« speziell gekennzeichnet und haben spezielle Attribute.

Als nächstes werden die externen Anschlüsse mit den entsprechenden Anschlüssen der Rechnerkomponenten durch Eintrag ihrer internen Bezeichner in die entsprechenden Menüs verbunden. An den Daten-, den Befehls- und den Peripheriebus wird jeweils der Takt (Clk) und ein Initialisierungssignal, das von der System-Reset-Einheit bereitgestellt wird, angeschlossen. Die Ausgabeschnittstelle wird mit den Leuchtdioden (LD), die Eingabeschnittstelle mit den Schaltern (SW) und die UART mit den beiden seriellen Signale »RxD« und »TxD« verbunden. Das Rücksetzsignal, das die Debug-Schnittstelle ausgibt, ist mit dem entsprechenden Eingang der System-Reset-Schaltung zu verbinden. Der »Slowest_sync_clk« für die Abtastung und zeitliche Ausrichtung der Initialisierung ist der Takt »Clk« und der externe Reset-Eingang ist »Reset«. Der ungenutzte Zusatz-Reset-Eingang »Aux_Reset_In« ist mit Masse und der hier nicht benötigte Eingang »Dcm_locked«[5] mit »net_vcc« zu verbinden.

[5]Bei Verwendung einer DCM verzögert dieses Signal die Deaktivierung des internen Rücksetzsignals bis zum Einrasten des Phasenregelkreises.



DLMB					
LMB_Clk	External Ports::...	Clk	I		CLK
SYS_Rst	System_Reset:...	LMB_Reset	I		RST
ILMB					
LMB_Clk	External Ports::...	Clk	I		CLK
SYS_Rst	System_Reset:...	LMB_Reset	I		RST
PLB					
PLB_Clk	External Ports::...	Clk	I		CLK
SYS_Rst	System_Reset:...	PLB_Reset	I		RST
Bus_Error_Det		No Connection	O		INTERRU...
microblaze_0					
MB_RESET	System_Reset:...	Prozessor_Reset	I		RST
DBG_STOP		No Connection	I		
MB_Halted		No Connection	O		
MB_Error		No Connection	O		
WAKEUP		No Connection	I	[0:1]	
SLEEP		No Connection	O		
DBG_WAKEUP		No Connection	O		
LOCKSTEP_MASTER_OUT		No Connection	O	[0:40...	
LOCKSTEP_SLAVE_IN		No Connection	I	[0:40...	
LOCKSTEP_OUT		No Connection	O	[0:40...	
(BUS_IF) DLMB	Connected t...	Connected to BUS DLMB			
CLK	External Ports::...	Clk	I		CLK
(BUS_IF) ILMB	Connected t...	Connected to BUS ILMB			
CLK	External Ports::...	Clk	I		CLK
(BUS_IF) DPLB	Connected t...	Connected to BUS PLB			
CLK	External Ports::...	Clk	I		CLK
(BUS_IF) IPLB	Connected t...	Connected to External ...			
CLK	External Ports::...	Clk	I		CLK
Speicher					
DMem_Controller					
IMem_Controller					
Debug_Schnittstelle					
Interrupt		No Connection	O		INTERRU...
Debug_SYS_Rst	System_Reset:...	Debug_Reset	O		RST
Ausgabeschnittstelle					
(IO_IF) gpio_0	Connected t...	Connected to External ...			
GPIO_IO_I		No Connection	I	[0:7]	
GPIO_IO_O	External Ports::...	LD	O	[0:7]	
GPIO_IO_T		No Connection	O	[0:7]	
GPIO_IO		No Connection	IO	[0:7]	
Eingabeschnittstelle					
(IO_IF) gpio_0	Connected t...	Connected to External ...			
GPIO_IO_I	External Ports::...	SW	I	[0:7]	
GPIO_IO_O		No Connection	O	[0:7]	
GPIO_IO_T		No Connection	O	[0:7]	
GPIO_IO		No Connection	IO	[0:7]	
UART					
Interrupt		No Connection	O		INTERRU...
(IO_IF) uart_0	Connected t...	Connected to External ...			
RX	External Ports::...	RxD	I		
TX	External Ports::...	TxD	O		
System_Reset					
Slowest_sync_clk	External Ports::...	Clk	I		CLK
Ext_Reset_In	External Ports::...	Reset	I		RST
Aux_Reset_In	net_gnd	net_gnd	I		RST
MB_Debug_Sys_Rst	Debug_Schnitt...	Debug_Reset	I		RST
Dcm_locked	net_vcc	net_vcc	I		
MB_Reset	microblaze_0::...	Prozessor_Reset	O		RST
Bus_Struct_Reset	DLMB::SYS_RST ILMB::SYS_RST	LMB_Reset	O		RST
Peripheral_Reset	PLB::SYS_Rst	PLB_Reset	O		RST
Interconnect_aresetn		No Connection	O		RST
Peripheral_aresetn		No Connection	O		RST

Abbildung 8: Verbindung der externen Anschlüsse mit Systemkomponenten

1.4 Adressbereiche festlegen

In der Ansicht »Addresses« sind für den Datenspeicher-Controller, den Befehlsspeicher-Controller, die parallelen, die serielle und die Debug-Schnittstelle die Adressbereiche festzulegen (Abb. 9). Mit dem Button »Generate Addresses« (in Abb. 9 oben rechts) lassen sich Voreinstellungen treffen. Die kleinste zulässige Blockspeichergröße ist 8 kByte. Der Adressbereich für Befehle und Daten sollte übereinstimmen. Die peripheren Einheiten müssen überlappungsfreie Adressbereiche haben. Für die parallelen Schnittstellen genügt der kleinste unter »Size« einstellbare Wert. Bei der UART verursachen reservierte Adresssbereiche kleiner 2 kByte Fehlfunktionen. Die mit »Generate Adresses« (Button rechts oben in Abbildung 9) automatisch vergebene Adressbereiche von je 64 kByte können auch beibehalten werden. Denn von dem 2^{32} Byte großen Adressraum wird auch dann nur ein winziger Ausschnitt genutzt.

Bus Interfaces | Ports | Addresses — Generate Adresses

Instance	Base Name	Base Address	High Address	Size	Bus Interface(s)	Bus Name
microblaze_0's Address Map						
DMem_Controller	C_BASEADDR	0x00000000	0x00001FFF	8K	SLMB	DLMB
IMem_Controller	C_BASEADDR	0x00000000	0x00001FFF	8K	SLMB	ILMB
Eingabeschnittstelle	C_BASEADDR	0x81400000	0x8140FFFF	64K	SPLB	PLB
Ausgabeschnittstelle	C_BASEADDR	0x81420000	0x8142FFFF	64K	SPLB	PLB
UART	C_BASEADDR	0x84000000	0x8400FFFF	64K	SPLB	PLB
Debug_Schnittstelle	C_BASEADDR	0x84400000	0x8440FFFF	64K	SPLB	PLB

Abbildung 9: Festlegen der Adressbereiche

1.5 Erzeugung der UCF-Datei

In der UCF-Datei (UCF – **u**ser **c**onstraint **f**ile) werden den symbolischen Schaltkreisnahmen Gehäuseanschlüsse zugeordnet und genau wie beim normalen Entwurf mit »Xilinx-ISE« weitere Anschluss- und Zeit-Constraints übergeben. Wichtig ist die Vorgabe der Taktperiode, damit die Synthese prüfen kann, ob die Schaltung ausreichend schnell ist. Die UCF-Datei wird über den Reiter »Project« unter dem linken Auswahlfeld, »Project Files« und »UCF File data/system.ucf« ausgewählt. Abbildung 10 zeigt die erforderlichen Einträge für das Beispielprojekt.

1.6 Abschluss des Hardware-Entwurfes

Der Software-Entwurf soll mit SDK, einer auf »eclipse« aufsetzenden Entwicklungsumgebung erfolgen. Dazu sind die Daten zu exportieren:

- »Project« ▷ »Export Hardware Design to SDK ...«
- »Export and Launch SDK«

Die Synthese, Verdrahtung und Platzierung dauert mehrere Minuten. Nach erfolgreichem Abschluss wird ein Unterverzeichnis ».../SDK/SDK_Export/hw/« erstellt, in das alle hardware-spezifischen Daten incl. der Bitdatei für die Programmierung, Treiber etc. hinterlegt werden.

Falls in SDK nicht automatisch das Fenster »New Board Support Package Project« aufgeht:

- File > New > Board Support Package

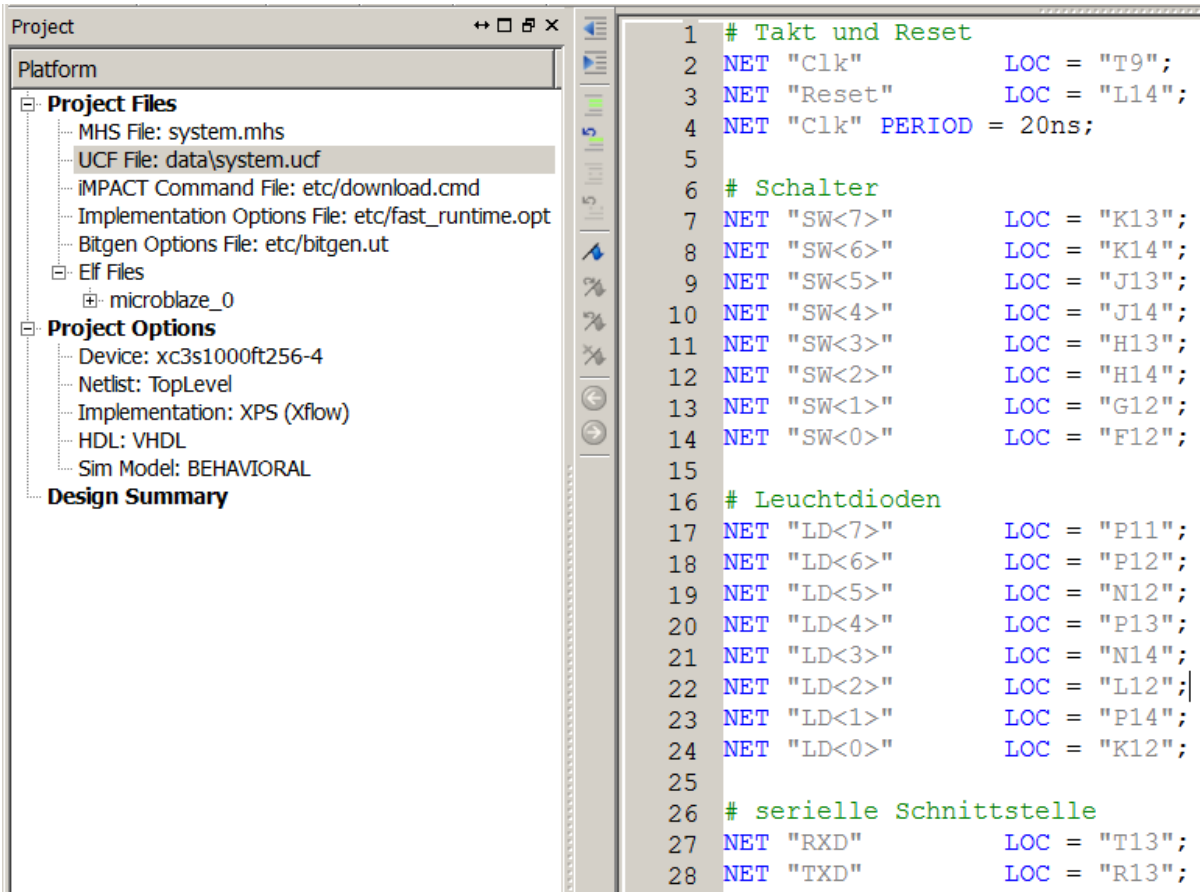


Abbildung 10: : Erforderliche Einträge in der ucf-Datei

In dem Fenster folgende Eingaben vornehmen:

- Project Name: Basissystem (eingeben)
- Hardware Platform: Aufg1_hw_platform (lassen)
- standalone (lassen).

Ergebnis im Projektordner ».../Aufg1« ein Unterordner »SDK/SDK_workspace« mit den in Abb. 11 dargestellten Unterordnern. In »Aufg1_hw_platform werden die hardware-spezifischen Dateien inc. der Bit-Datei für die Programmierung der Prozessorschaltung in den FPGA abgelgt. Der Ordner enthält alle für die Software-Entwicklung benötigten Dateien, insbesondere in »include« die Header-Dateien mit den Konstanten und Unterprogrammen für die Ansteuerung der Hardware aus C-Programmen.

SDK kann auch direkt aus der Menü-Leiste von Linux über

- »Anwendungen« ▷ »Umgebung« ▷ »Xilinx Plattformstudio (SDK)«

und für Windows über

- »All Programms« ▷ »Xilinx Design Tools« ▷ »ISE Design Suite 14.6« ▷ »EDK« ▷ »Xilinx System Developement Kit«

gestartet werden. Nach einem Start von SDK ist im Fenster »Workspace Launcher« das Arbeitsverzeichnis

- S:\users\<Benutzername>\Documents\Softprozessor\Aufg1\SDK\SDK_workspace

auszuwählen.

2 Software-Entwicklung mit SDK

Anlegen eines Softwareprojekts:

- »File« ▷ »New« ▷ »Application Project«

Im Application Project Fenster:

- Project Name: HalloWelt (eintragen)
- Use existing: Basissystem

Rest lassen, Next

- "Empty" auswählen

Finish. Eine C-Quellfile anlegen:

- »File« ▷ »New« ▷ »Source File«

Im aufgehenden Fenster:

- Source Folder: HalloWert
- Sorce File: Hallo.c (Endung .c ist wichtig)

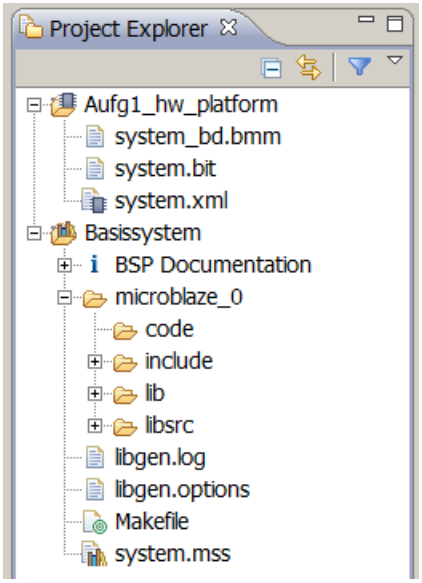


Abbildung 11: : Dateibaum mit den hardware-spezifischen Dateien zur Software-Entwicklung, wie sie in SDK angezeigt wird

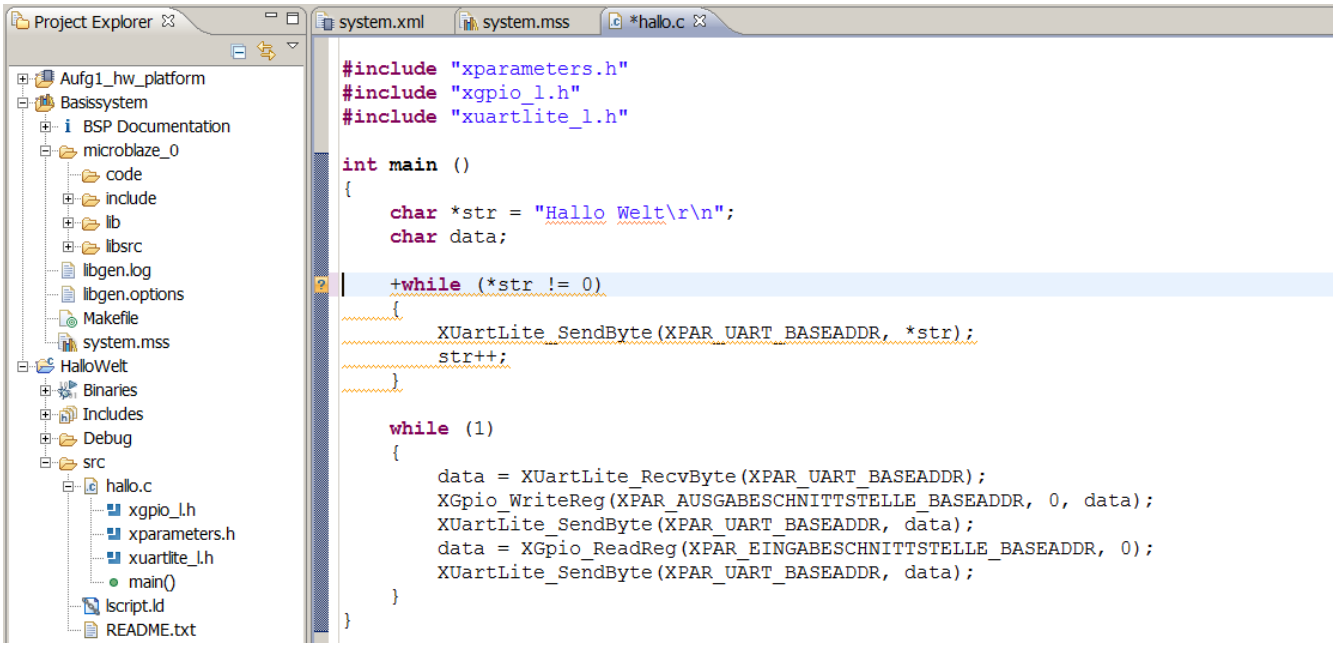


Abbildung 12: : Beispielprojekt mit erkanntem Programmierfehler

Eingabe des Programmtextes aus Abb. 12 in »Hallo.c« und Speichern. Standardmäßig ist »Build autmaticaly« eingestellt, so dass das Programm bei jedem Speichern automatisch auf Fehler überprüft und übersetzt wird. In Abb. 12 enthält die Zeile, vor der das gelbe Fragezeichen steht, einen Fehler – das Plus vor dem while – der, bevor das Programm richtig übersetzt werden kann, zu beseitigen ist.

2.1 Das Beispielprogramm

Das Beispielprogramm in Abb. 12 beginnt mit den include-Anweisungen

```
#include "xparameters.h"
#include "xgpio_l.h"
#include "xuartlite_l.h"
```

Diese importieren die benötigten Konstanten, Makros und Unterprogramme zur Schnittstellenansteuerung. Für das Projekt werden genutzt, aus »xparameters.h« die Basisadressen der Eingabe- und Ausgabeschnittstellen

```
XPAR_AUSGABESCHNITTSTELLE_BASEADDR
XPAR_EINGABESCHNITTSTELLE_BASEADDR
XPAR_UART_BASEADDR
```

aus »xgpio_l.h« die Funktionen bzw. Makros für die parallele Ein- und Ausgabe

```
XGpio_WriteReg(BaseAddress, RegOffset, Data)
XGpio_ReadReg(BaseAddress, RegOffset)
```

und aus »xuartlite_l.h« die Funktionen bzw. Makros für die serielle Ein- und Ausgabe

```
XUartLite_RecvByte(XPAR_UART_BASEADDR)
XUartLite_SendByte(XPAR_UART_BASEADDR, data)
```

Das Hauptprogramm, das in eingebetteten Systemen und Mikrorechnern nach der (Neu-) Initialisierung gestartet wird, heißt in »c« immer »main«. Im Beispielprogramm wird am Programmanfang eine Zeichenkettenkonstante »Hallo Welt\r\n« im Befehlsspeicher angelegt, eine Zeigervariable »str«, die auf den Anfang der Zeichenkette zeigt, und eine Variable zur Speicherung eines Zeichens vereinbart:

```
int main()
 {
 char *str = "Hallo Welt\r\n";
 char data;
```

Zeichenketten werden in »c« mit einer echten »0« abgeschlossen. Die beiden Steuerzeichen »\r\n« haben die Werte 0c0D und 0x0A (siehe ASCII-Tabelle Abb. 13). Insgesamt entspricht die Zeichenkettenkonstante der Zahlenfolge »0x48 0x61 0x6c 0x6c 0x6F 0x20 0x57 0x65 0x6c 0x74 0x0D 0x0A 00«[6]. Die nachfolgende While-Schleife versendet, bis der Zeichenwert null ist, das ausgewählte Zeichen und erhöht den Zeiger. Auf diese Weise wird die gesamte Zeichenfolge inkl. Zeilenumbruch an den PC versendet:

```
 while (*str != 0)
 {
  XUartLite_SendByte(XPAR_UART_BASEADDR, *str);
  str++;
 }
```

In der nachfolgenden Endlosschleife wird jeweils auf ein Byte von der seriellen Schnittstelle gewartet, der Bytewert auf die Leuchtdioden ausgegeben und zurückgesendet, der Byte-Wert von den Schaltern eingelesen und gleichfalls an den PC gesendet. Um z.B. ein Leerzeichen zwischen zwei empfangenen Zeichen zurückzusenden, ist mit den Schaltern eine 0x20 einzustellen:

```
  while (1)
  {
   data = XUartLite_RecvByte(XPAR_UART_BASEADDR);
   XGpio_WriteReg(XPAR_AUSGABESCHNITTSTELLE_BASEADDR, 0, data);
   XUartLite_SendByte(XPAR_UART_BASEADDR, data);
   data = XGpio_ReadReg(XPAR_EINGABESCHNITTSTELLE_BASEADDR, 0);
   XUartLite_SendByte(XPAR_UART_BASEADDR, data);
  }
}
```

[6]Bitte anhand der ASCII-Tabelle in Abb. 13 und später bei der Programmabarbeitung im Schrittbetrieb kontrollieren.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Abbildung 13: ASCII-Tabelle zur Umrechnung der Zeichen in Zeichenwerte

2.2 Programmtest

Vor dem Programmtest sind

- die zu programmierende Baugruppe an die Versorgungsspannung und an das Programmierkabel anzuschließen und mit
- »Xilinx Tools« ▷ »Program FPGA«

unter Bebehaltung der Pfadeinstellungen

- Bitstream: C:\Users\gkemnitz\Documents\Softprozessor\Aufg1\SDK\SDK_workspace\Aufg1_hw_pl
- BMM File: C:\Users\gkemnitz\Documents\Softprozessor\Aufg1\SDK\SDK_workspace\Aufg1_hw_p

die Konfigurationsdatei für die Hardware in den programmierbaren Logikschaltkreis zu laden. Weiterhin ist die Baugruppe über ihre serielle Schnittstelle mit dem Rechner zu verbinden uns auf dem Rechner ein Monitorprogramm für die serielle Kommunikation zu starten. Unter Linux ist das derzeit »Kermit« und unter Windows »Putty«[7] installiert. Unter Linux das Terminalprogramm (kermit) starten und kongurieren:

[7] Putty funktioniert aktuell, vermutlich wegen irgendwelcher Windows-Updates nicht mehr. Betreuer nach Alternativen Fragen oder Terminal des Debuggers nutzen (Abb. 16).

```
...$ kermit
... C-Kermit > set line /dev/ttyS0
... C-Kermit > set baud 9600
... C-Kermit > set carrier-watch off
... C-Kermit > set flow-control none
... C-Kermit > c
```

Danach muss eine Meldung der Form erscheinen: Connecting to /dev/ttyS0, speed 9600
:

- »Linux Menü-Leiste« ▷ »Anwendungen« ▷ »Zubehör« ▷ »Serial port terminal«

Unter Windows »Putty« starten

- »Start« ▷ »All Programs« ▷ »...« ▷ »Putty«

und konfigurieren

- »serial« auswählen
- Schnittstelleneinstellungen über »Connection ▷ Serial« vornehmen
- »Open«

Die Einstellungen für die serielle Verbindung müssen mit denen der seriellen Schnittstelle des Softprozessors übereinstimmen: Speed 9600, ungerade (odd) Parität, acht Datenbits, ein Stopbit und Kontrollflusssteuerung »none«. Bei Anschluss der FPGA-Versuchsbaugruppe über ein normales serielles Kabel ist der serielle Port des Monitors unter Linux in der Regel »/dev/ttyS0«. Bei Verwendung eines USB-nach-seriell-Adapters schaut man am besten, welcher Port beim Anstecken der Baugruppe hinzukommt. Unter Windows ist es COM1 oder COMx mit USB-Adapter. Wenn keine Übertragung zustande kommt, ist entweder der falsche Port ausgewählt oder es läuft noch ein anderes Programm auf dem PC, das den richtigen Port blockiert. Bei letzterem muss das andere Programm beendet und das Monitor-Programm neu gestartet werden. Auffällige Datenverfälschungen bei der seriellen Übertragung haben in der Regel eine falsch eingestellte Baudrate als Ursache. Alternativ kann auch eine falsche Taktfrequenz in der Konfiguration der UART des Softprozessors in Abb. 6 b die Quelle des Übels sein.

Nach erfolgreicher Programmierung des Schaltkreises und Start des Monitors wird mit

- »Run« ▷ »Run« (beim ersten Start »Launch on Hardware (GDB)«)

das Programm gestartet. Nach dem Start und nach jeder Betätigung von »BTN3« schickt es an den Monitor den Text »Hallo Welt«, einen Zeilenwechsel und einen Wagenrücklauf, wartet danach bei jedem Durchlauf der zweiten While-Schleife auf eine Zeicheneingabe im Monitor-Programm auf dem PC, zeigt den Zeichenwert auf den Leuchtdioden an, sendet das Zeichen und anschließend das mit den Schaltern eingestellte Zeichen zurück zur Anzeige auf dem Ausgabefenster des Monitor-Programms. Abbildung 14 zeigt als Beispiel die Monitor-Ausgaben für die Eingabezeichenfolge »abzd<Enter>a109« und die Schalterstellung 0x20 (Leerzeichen).

Empfänger und Sender der seriellen Schnittstelle des Softprozessors haben einen 8 Zeichen großen Pufferspeicher, der möglicherweise zum Programmstart noch mit Zeichen gefüllt ist. Das kann unerwartete Ein- und Ausgaben nach dem Programmstart verursachen.

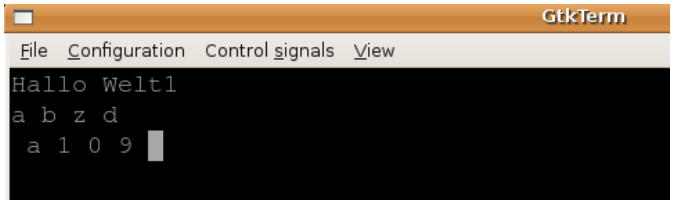


Abbildung 14: Beispieltestausgabe im Ausgabefenster des Monitor-Programms

2.3 Debugger

Vor dem Start des Debuggers ist wie im Vorabschnitt die Hardware-Konfiguration in den Schaltkreis zu laden und das Monitor-Programm auf dem PC zu starten. Der Debugger wird gestartet mit:

- »Run« ▷ »Debug«

Auf die Fragen

- ... terminate previous launch: yes
- ... Confirm Perspective Switch: yes

(Rückkehr zur Programmieransicht über »Window« ▷ »Open Perspective« ▷ »C/C++«.)

Der übliche Weg, ein Programm gründlich zu testen und erkannte Fehler zu lokalisieren, ist seine Abarbeitung im Schrittbetrieb oder in kleinen Stücken. Dazu muss das Programm nach jedem Testschritt angehalten werden, um die Zwischenergebnisse zu kontrollieren. Der Debugger bietet folgende Möglichkeiten, ein Programm zu stoppen:

- »Run« ▷ »Step Into«: Stopp nach der nächsten Anweisung, bei einem Unterprogrammaufruf, nach der ersten Anweisung des Unterprogramms.
- »Run« ▷ »Step Over«: Stopp nach der nächsten Anweisung, bei Unterprogrammaufrufen nach dem Unterprogramm.
- Setzen von Haltepunkten: »rechte Maustaste vor der Zeile« ▷ »Toggle Breakpoint«
- Programm starten/fortsetzen (»Run« ▷ »Resume«): Stopp am nächsten Haltepunkt.

Für Haltepunkte lässt sich ferner mit der rechten Maustaste über »Breakpoint Properties ...« eine Bedingung, die zusätzlich erfüllt sein muss, damit das Programm anhält, und ein »Ignore count« einstellen, um die Abarbeitung erst nach einer mehrmaligen Abarbeitung der Anweisung z.B. in einer Schleife anzuhalten. Zur Visualisierung der Variablenwerte bei angehaltenem Programm platziert man einfach den Mauszeiger über der Variablen. In Abb. 14 wird in dem gelben Fenster der Zeigerwert von »str« und die Zeichenkettenkonstante, die ab dieser Adresse bis zur nächsten echten Null im Speicher steht, angezeigt. Beim Test des Beispielprogramms mit dem Debugger ist zu beachten, das die Funktion »XUartLite_RecByte(...)« in einer internen Schleife auf die Eingabe eines Zeichen vom PC wartet. Ohne diese Eingabe oder ohne gestartetes Monitor-Programm wartet der Debugger dauerhaft auf den Funktionsabschluss.

Um einen Debugger zu beherrschen, muss man ein wenig ihm spielen: Einzelschritte und Anweisungsblöcke abarbeiten, Variablenwerte gezielt ändern und geeignet anzeigen lassen, die Fenster umsortieren, ... Abbildung 16 zeigt eine anders eingestellte Debugger-Oberfläche, in der der serielle Monitor des Debuggers für die Kommunikation mit der Baugruppe konfiguriert und Variablenwerte in Tabellenform angezeigt werden.

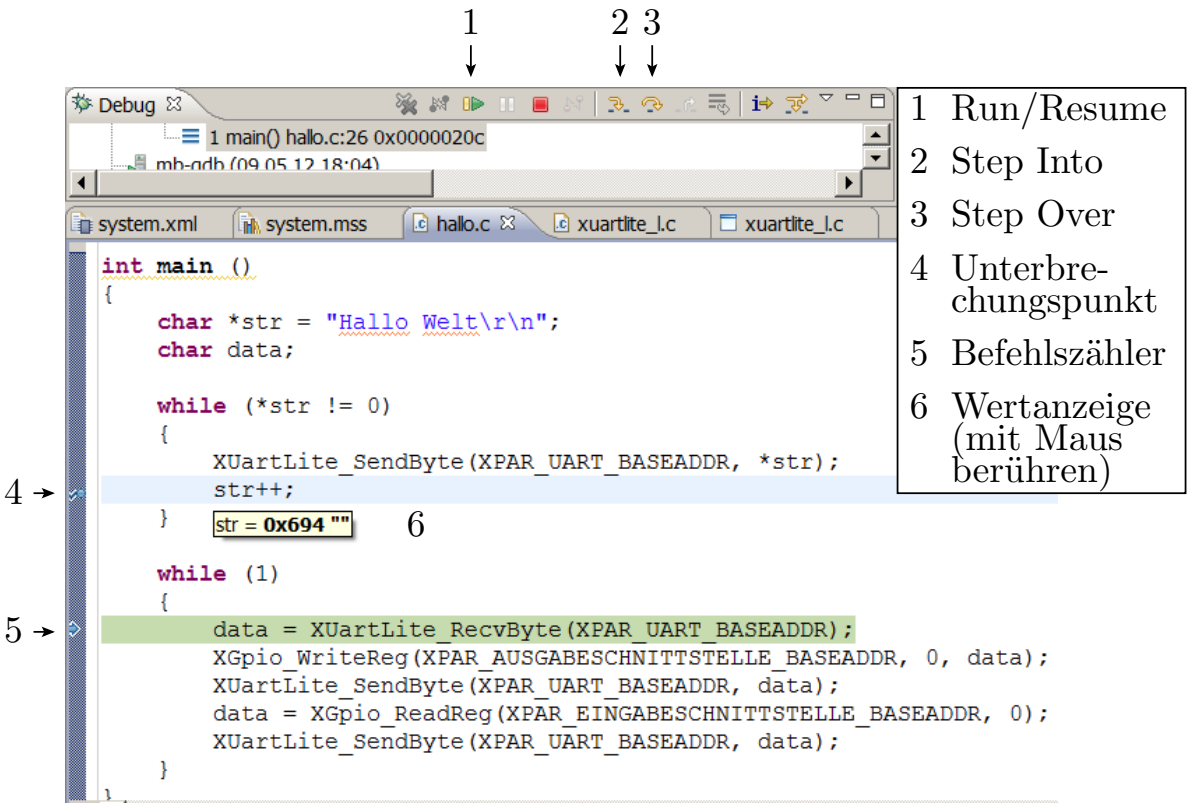


Abbildung 15: Programmoberfläche des Debuggers

3 Aufgaben

Aufgabe 1.1: EDK-Referenzentwurf

Führen Sie den Hardware-Entwurf mit EDK in der beschriebenen Weise durch.

Aufgabe 1.2: SDK-Referenzentwurf

Exportieren Sie den Hardware-Entwurf nach SDK und führen Sie den Software-Entwurf in der beschriebenen Weise durch.

1. Testen Sie das Programm zuerst mit »Run«.
2. Wiederholen Sie den Test mit dem Debugger im Schrittbetrieb.

   - Notieren Sie, wie sich der Wert des Zeigers »str« bei jedem Durchlauf der ersten While-Schleife ändert und was jeweils als Inhalt der Zeichenkette, auf die der Zeiger zeigt, vom Debugger angezeigt wird.
   - Setzen Sie einen Unterbrechungspunkt vor die Anweisung zum Einlesen der Schalterwerte. Untersuchen Sie, ob das Programm vor oder nach dem Einlesen der Schalterwerte stoppt.

Aufgabe 1.3: Erstes eigenes Programm

Legen Sie mit »File« »New« »Xilinx C Project« ein neues Projekt »Aufg1_3« an. (»Empty Application«, nächstes Fenster »Target an existing Board Support Package« und dort »Basissystem« auswählen.) Erzeugen Sie in diesem Projekt eine leeres C-Quellprogramm mit dem Namen »main_a1_3.c«. Kopieren Sie in die neue Programmdatei den Inhalt von »hallo.c«. Ändern Sie das Programm so ab, dass der von der seriellen Schnittstelle empfangene und auf die Leuchtdioden ausgegebene Wert bitweise mit dem von den Schaltern eingelesenen Wert EXOR-verknüpft, addiert oder in einer anderen Weise verändert wird. Füllen Sie für ihr gewähltes Programm die nachfolgende Tabelle mit Testbeispielen aus und führen Sie die Tests durch.

seriell empfangenes Zeichen	Zeichenwert	Schaltereingabe	LED-Ausgabe
'a'		0x43	
'1'		0x65	

4 Aufgaben zur Vorbereitung

- Wie lange dauert die Übertragung des Wortes »Hallo« mit den eingestellten Parametern mindestens (Baud-Rate von 9600 Baud, ein Startbit, acht Datenbits, ein Paritätsbit und ein Stoppbit)?
- Schreiben Sie Ihren Namen gefolgt von einem Zeilenumbruch als Zahlenfolge im ASCII-Code.

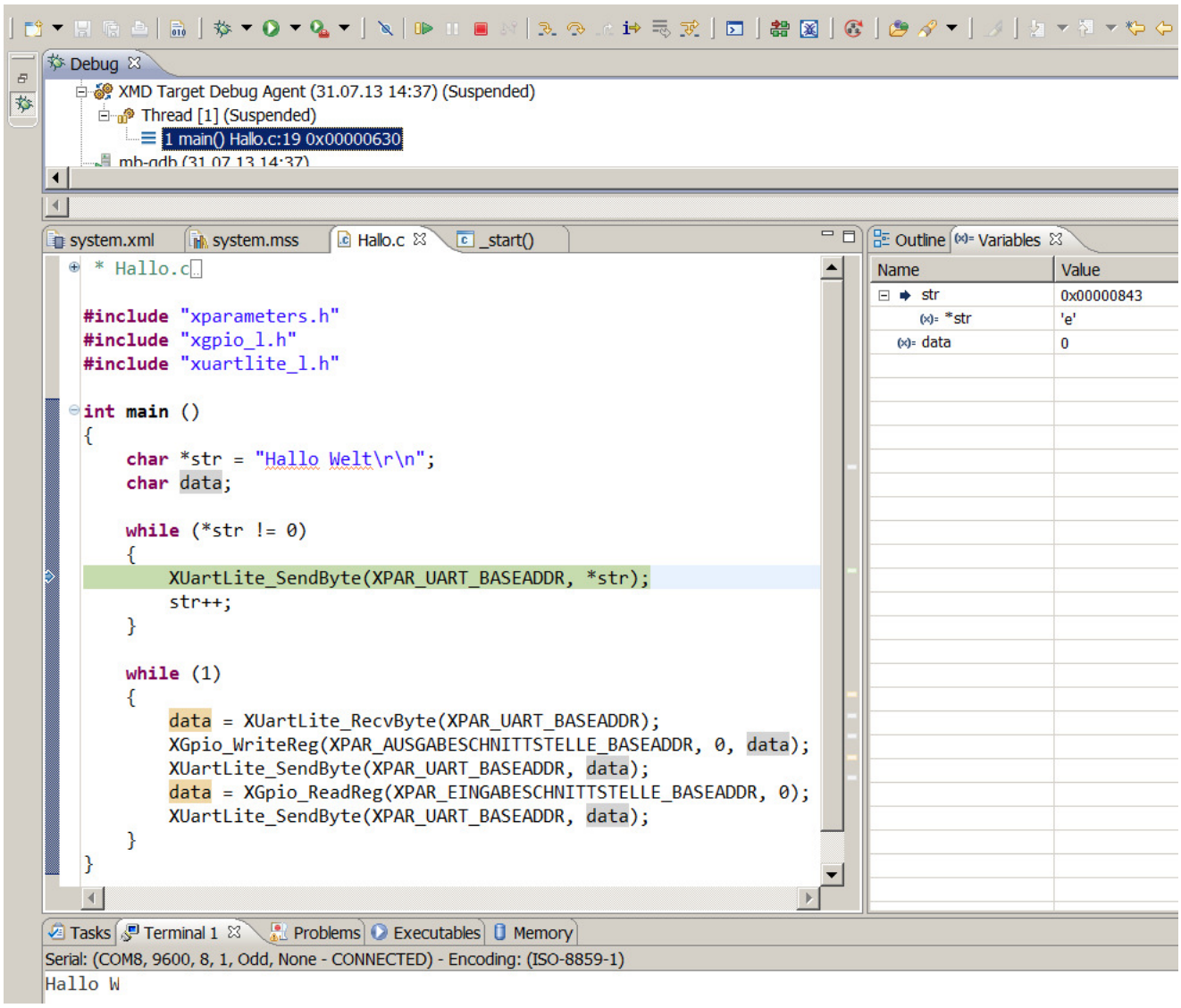


Abbildung 16: Anders eingestellte Debugger-Oberfläche, in der der serielle Monitor des Debuggers für die Kommunikation mit der Baugruppe konfiguriert und die Variablenwerte des Programms angezeigt werden.