

Schritt-für-Schritt-Anleitung für den Hardware-Test mit Chipscope

Prof. G. Kemnitz, TU Clausthal, Institut für Informatik

22. April 2016

In das Minimalsystem aus den beiden ersten Anleitungen sollen zwei integrierte Logikanalysatoren eingefügt werden, einen zur Beobachtung der Programmabarbeitung auf dem Prozessor und einen zweiten zur Untersuchung der Signalverläufe auf dem AXI-Bus. Das Testprogramm ist eine Schleife, in der der Wert einer Variablen hochgezählt und auf die LEDs ausgegeben wird. Das erste Ziel wird sein, die wichtigsten Trace-Signale des Prozessors für einen Schleifendurchlauf zu visualisieren. Zur Interpretation der Trace-Signale wird das disassemblierte Programm und der erste integrierte Logikanalysator benötigt. Es wird sich zeigen, dass der Prozessor 10 der 25 Takte für die Abarbeitung der innersten Schleife allein für die Ausgabe des Zählwerts auf die LEDs benötigt. Im zweiten Experiment wird mit dem zweiten Logikanalysator untersucht, was an der Ausgabe an die LEDs so viel Zeit verbraucht. Im abschließenden Versuch werden beide Logikanalysatoren miteinander gekoppelt, um die Zeitbeziehungen zwischen den Trace-, AXI-Bus- und LED-Ausgabesignalen zu untersuchen.

1 Übernahme der Beschreibung des Minimalsystems

Statt eines Neuentwurfs soll die Rechnerbeschreibung des bisherigen Minimalsystems aus der mhs- und der ucf-Datei auf der Webseite erzeugt werden. Dazu müssen die beiden Dateien »min_sys.mhs« und »min_sys.ucf« von der Web-Seite ins Download-Verzeichnis geladen werden. EDK ist zu starten und »Create New Blank Project« auszuwählen (Abb. 1).

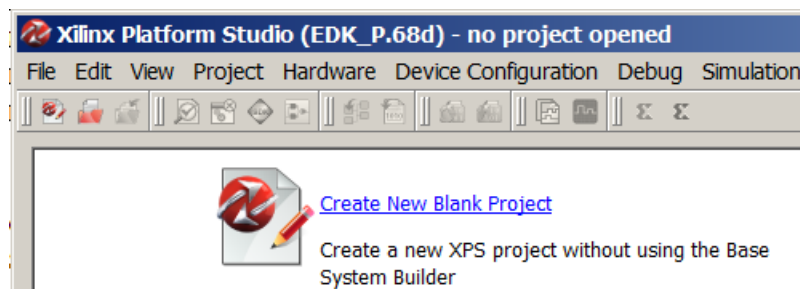


Abbildung 1: Erzeugen des neuen Hardware-Projekts

In dem sich öffnenden Fenster, Abb. 2, wird festgelegt, in welchem Ordner die Projektdatei angelegt werden soll. Das sollte ein leerer Ordner sein, der sich unter »Browse, New« direkt erzeugen lässt. Der Typ des zu programmierenden Logikschaltkreises (spartan6, xc6slx16, ...) und das zu importierende mhs-File sind auszuwählen. Alle anderen Einträge bleiben leer. Bestätigung mit »OK«.

Die zu übernehmende ucf-Datei »..\Download\min_sys.ucf« kann in das Projektunterverzeichnis »...\data« kopiert und in der Projektbaumansicht, Abb. 3, mit »Rechtsklick ▷

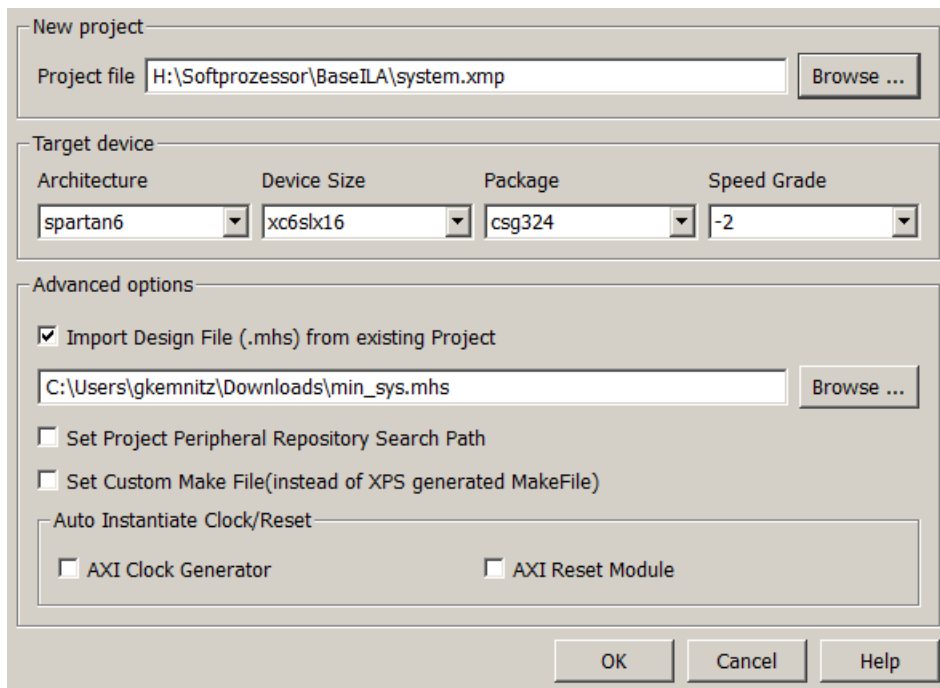


Abbildung 2: Festlegung des Pfads für das Software-Projekt

Change« gegen die automatisch erzeugte »system.ucf« ausgetauscht werden. Die Alternative ist, beide ucf-Dateien im Editor zu öffnen und den Inhalt rüberzukopieren.

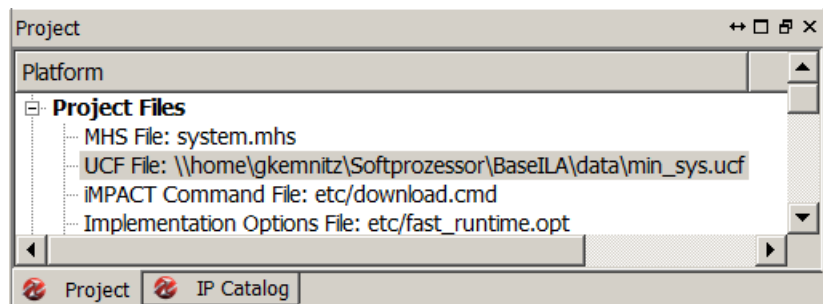


Abbildung 3: Ersatz der ucf-Datei

Mit den beschriebenen Schritten lässt sich jeder Entwurf in einem neuen Projekt wiederherstellen, um Bausteine erweitern, mit »Run DRCs« kontrollieren, mit »Export Design« abschließen und nach SDK exportieren.

2 Konfiguration der integrierten Logikanalysatoren

Am einfachsten lassen sich die integrierten Logikanalysatoren in EDK mit dem Debug-Konfigurator einfügen. Der Debug-Konfigurator wird über das Menü:

Debug ▷ Debug Configuration

gestartet. Im aufgehenden Fenster ist unten links »Add Chipscope Peripheral..« auszuwählen. Im nächsten sich öffnenden Fenster, Abb. 4, gibt es drei Auswahlmöglichkeiten. Für die Beobachtung frei wählbarer zu beobachtender Signale ist wie in Abb. 4 der mittlere Punkt vor »adding ILA« auszuwählen und die Auswahl mit »OK« zu bestätigen.

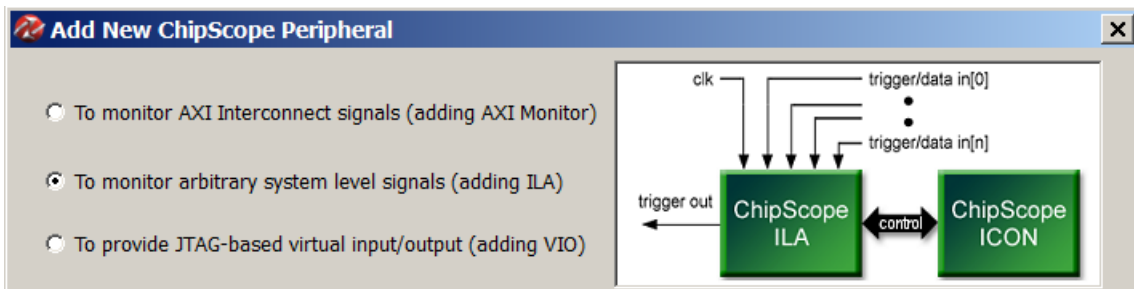


Abbildung 4: Auswahl des Chipscope-Typs

Die Graphik im Fenster zeigt, wie ein integrierter Logikanalysator angeschlossen wird, zum einen über einen Kontrollbus an den Controller »ICON«, der auch mit in das System eingebaut werden muss, zum anderen an die zu beobachtenden Dateneingänge und den Abtasttakt. Die abzutastenden Dateneingänge können gleichzeitig Trigger-Eingänge sein, deren logische Werte für den Aufzeichnungsbeginn mit ausgewertet werden. Der Takt wird bei uns der Systemtakt sein.

Im nächsten Fenster, Abb. 5, werden unter dem Reiter »Basic« der Aufzeichnungstakt, die Speichertiefe des Logikanalysator und die zu beobachtenden Signale ausgewählt. Links unter »Available Ports on Instance« können die Bausteine des Rechnersystems und darunter die für den Anschluss eines integrierten Logikanalysators verfügbaren Ausgänge ausgewählt werden. In Abb. 5 ist der Prozessor und vom Prozessor sind die Trace-Signale für den Befehlscode und den Befehlszählerstand des zuletzt fertig gestellten Befehls, das Gültigkeitssignal für den Befehl und das Signal, das anzeigt, ob die Datenspeicher-Pipeline läuft, ausgewählt. Alle vier Signale werden im Bild der Trigger-Gruppe »TRIG0« zugeordnet.

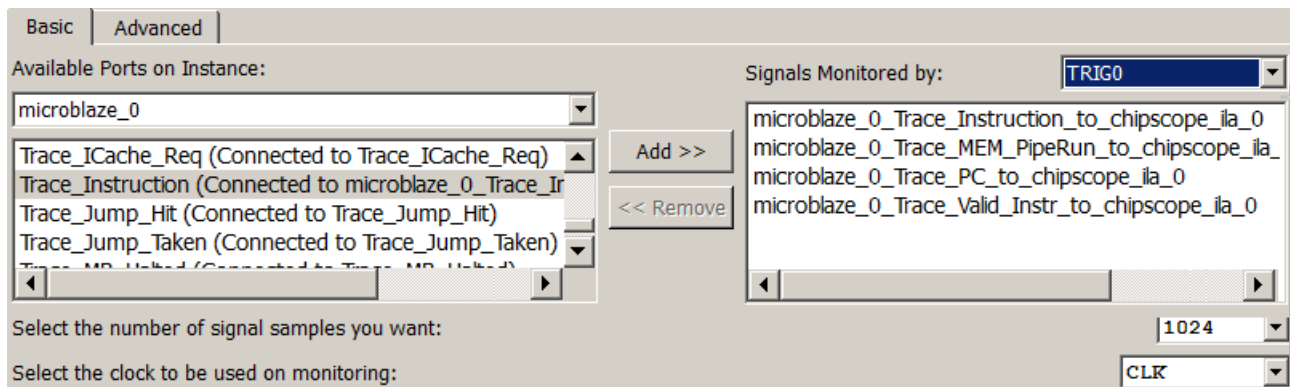


Abbildung 5: Auswahl der an Trigger-Port 0 anzuschließenden Trace-Signale

In Abb. 6 ist die Ausgabeinheit und von ihr das Ausgabebyte an die LEDs ausgewählt, um es mit Trigger-Eingang »TRIG1« zu verbinden. Trigger-Eintrag ändern!

Im selben Menü unter »Advanced« gibt es eine Reihe weiterer Einstellungen (Abb. 7). »Use the ILA Trigger Signals as Data Signals« bedeutet, dass die Signale an allen Trigger-Eingängen aufgezeichnet werden. Für die beiden Trigger-Eingänge ist nur die einfachste Funktionalität eingestellt: »Basic« (Zustands-Triggerung), nur eine »Match Unit«, kein Zähler, d.h. Aufzeichnungsbeginn mit oder vor dem ersten Trigger-Ereignis. »Enable Trigger Out Signal ..« bedeutet, dass das Trigger-Signal herausgeführt und an weitere integrierte Logikanalysatoren als Trigger-Eingang verwendet werden kann. So ist es möglich, für mehrere integrierte Logikanalysatoren die Aufzeichnungen zeitlich am selben Trigger-Ereignis auszurichten.

Nach Tätigung aller Einstellungen für den ersten integrierten Logikanalysator soll unten links mit dem Button »Add Chipscope Peripheral..« ein weiterer Logikanalysator hinzugefügt

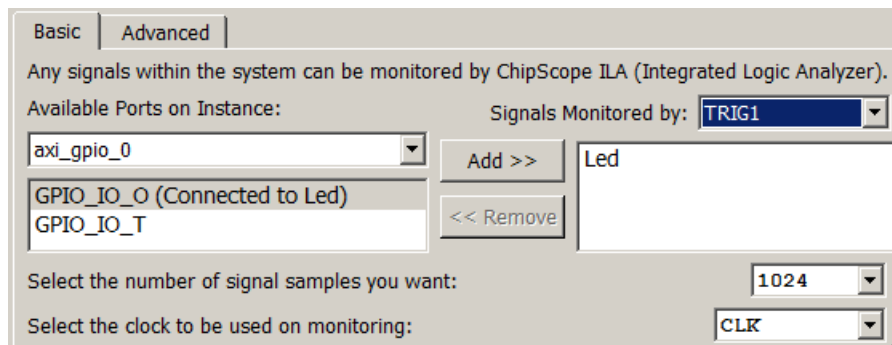


Abbildung 6: Auswahl der an Trigger-Port 1 anzuschließenden Ausgabesignale

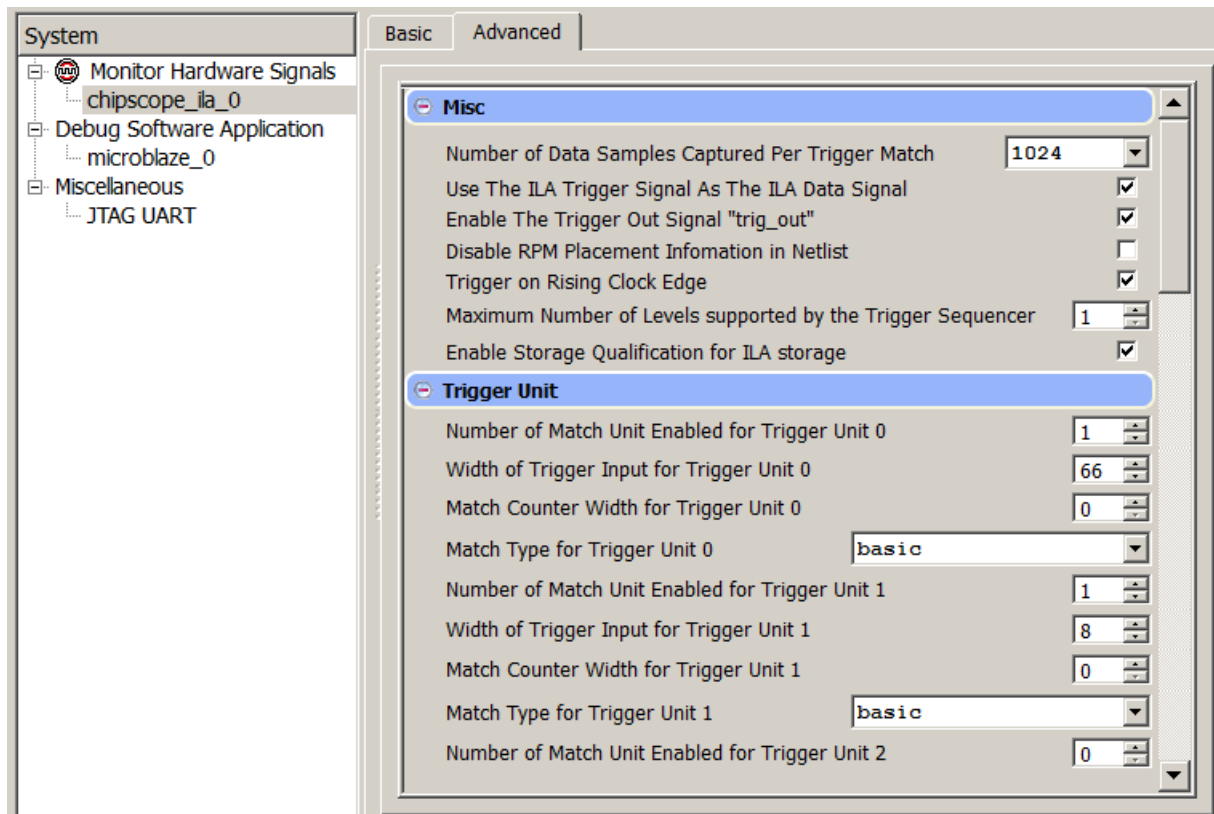


Abbildung 7: Zusätzliche Einstellungen für den ersten Logikanalysator

werden. Auf dem sich öffnenden Fenster, Abb. 8, soll dieses mal »AXI-Monitor« ausgewählt werden. Der AXI-Monitor hat eine Schnittstelle für den AXI-Bus, optionale Trigger-Ein- und Ausgänge und verwendet den AXI-Takt für die Aufzeichnung.

In dem Fenster für die Grundeinstellungen, Abb. 9, ist nur der zu beobachtende AXI-Bus und die Aufzeichnungstiefe auszuwählen. Ein zusätzlicher Haken bei »Hardware/Software Co-debug« würde so Trigger-Eingänge und -Ausgänge des Logikanalysators mit Steuersignalen des Debuggers verbinden, dass die Aufzeichnung des AXI-Monitors über die Haltepunkte des Debuggers gesteuert werden kann. Das wird in diesem Beispiel jedoch nicht gebraucht.

Unter »Advanced/AXI Settings« wird der AXI-Bustyp ausgewählt. Darunter unter »ILA-Settings« die Aufzeichnungstiefe und die Grundeinstellungen für den Trigger. Der AXI-Monitor soll einen 9-Bit »Trigger Input Port« für die Verbindung mit dem Trigger-Ausgang des anderen Logikanalysators und für die zusätzliche Aufzeichnung der LED-Ausgaben erhalten. Die »Global Port Settings« sollen bleiben wie gezeigt, insbesondere »Set number of match units for GLOBAL port« gleich eins. Denn der unter »ILA Settings« konfigurierte »Trigger Input Port« gehört zu

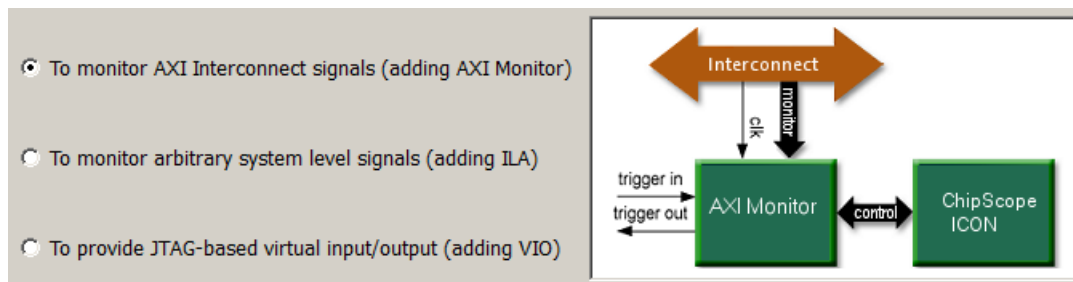


Abbildung 8: Auswahl eines AXI-Monitors

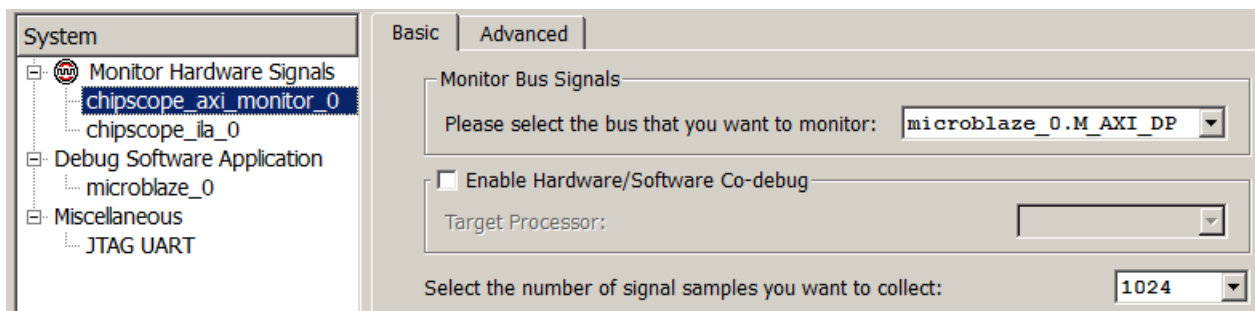


Abbildung 9: Grundeinstellungen des AXI-Monitors

dieser Gruppe.

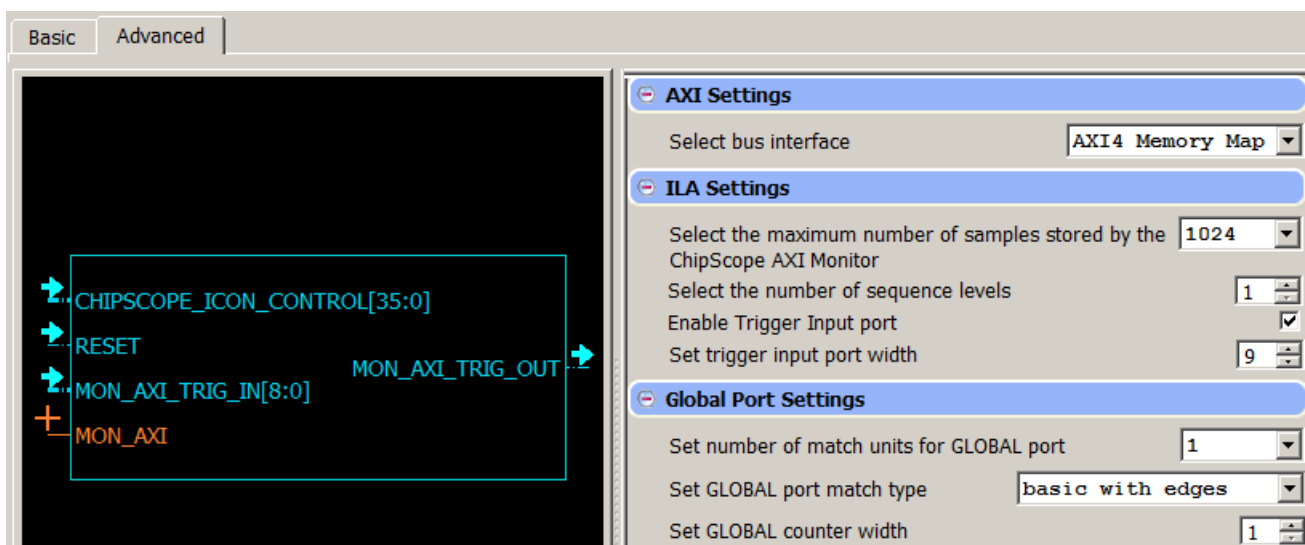


Abbildung 10: Erweiterte Einstellungen des AXI-Monitors

Der AXI-Bus besteht aus fünf ähnlichen Teilbussen: »Write Data Port«, »Write Adress Port«, »Read Data Port«, »Read Adress Port« und »Write Response Port«. Jeder dieser Busse bestehen wiederum aus Daten- bzw- Adressleitungen, Steuerleitungen und optionalen (in unserem Entwurf nicht vorhandenen) anwendungsspezifischen Leitungen. Unser Rechnersystem hat nur eine Ausgabeschnittstelle, so dass nur die Leitungen für die Schreiboperationen gebraucht werden:

- vom Bus »Write Data Port« die Signale »WDATA« und »WDATACONTROL«
- vom Bus »Write Adress Port« die Signale »WADDR« und »AWADDRCONTROL« und
- vom Bus »Write Response Port« das Signal »BRESP«.

Für diese drei Signalgruppen sind jeweils »number of match units« auf »1« zu belassen und für die übrigen auf »0« zu setzen. Abb. 11 zeigt die Einstellungen beispielhaft für die beiden Datenbusse. »AXI Stream Ports« und der »Protocol Checker« sind nicht verfügbar bzw. bleiben deaktiviert.

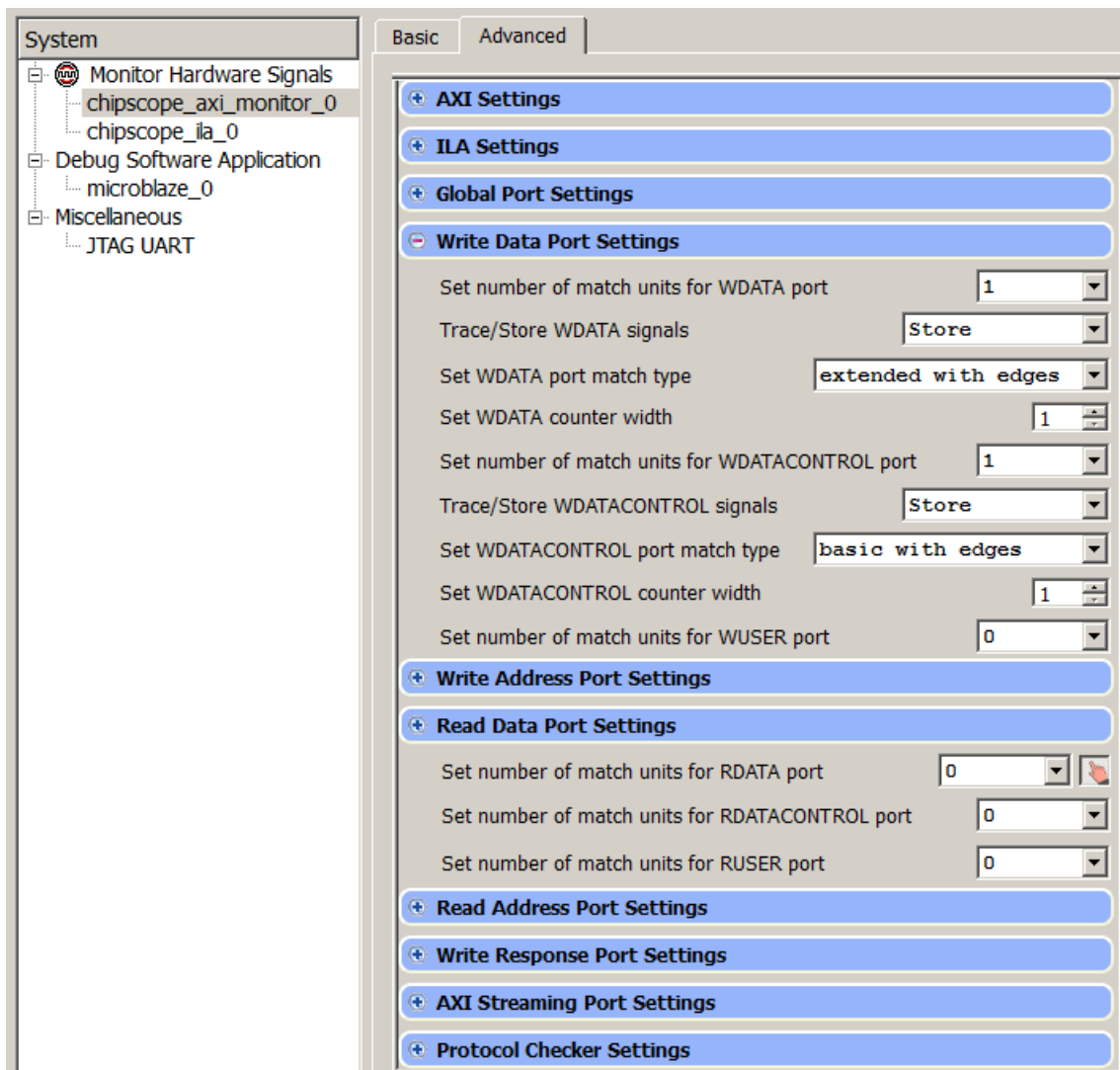


Abbildung 11: Erweiterte Einstellungen des AXI-Monitors

Im Fenster für die Debug-Konfiguration lässt sich weiterhin für den Debugger einstellen, wie viele Trigger-Einheiten er für Befehlsadressen (hardware breakpoints), Leseadressen (read address watch points) und Schreibadressen (write address watch points) haben soll (Abb. 12). Hier soll nichts geändert werden. Die JTAG UART im Systemhaus darunter soll auch aktiviert bleiben. Die gesamte »Debug Configuration« ist mit »OK« zu beenden¹.

Abb. 13 zeigt das Gesamtergebnis. Der erste Logikanalysator hat zwei »Match Units«. An »TRIG0« sind die Trace-Signale des Prozessors, an »TRIG1« die LED-Signale und als Abtasttakt ist der 100MHz-Takt der Baugruppe angeschlossen. Die Steuerung und PC-Kommunikation erfolgt über den Logikanalysator-Controller »ICON«, Steuer-Port »CTRL0«. Ein Trigger-Ausgang ist vorhaben, aber noch nirgends angeschlossen. Der zweite Logikanalysator ist als AXI-

¹Die »Debug Configuration« kann nach dem Schließen wieder zur weiteren Bearbeitung geöffnet werden. Das System speichert jedoch dann gewisse nachträgliche Änderungen nicht. Das ist daran erkennbar, dass sie beim wiederholten Öffnen nicht mehr da sind. Mögliche Workarounds sind, den betroffenen integrierten Logikanalysator zu löschen, neu anzulegen und komplett neu zu konfigurieren oder die Korrekturen direkt im mhs-File vorzunehmen.

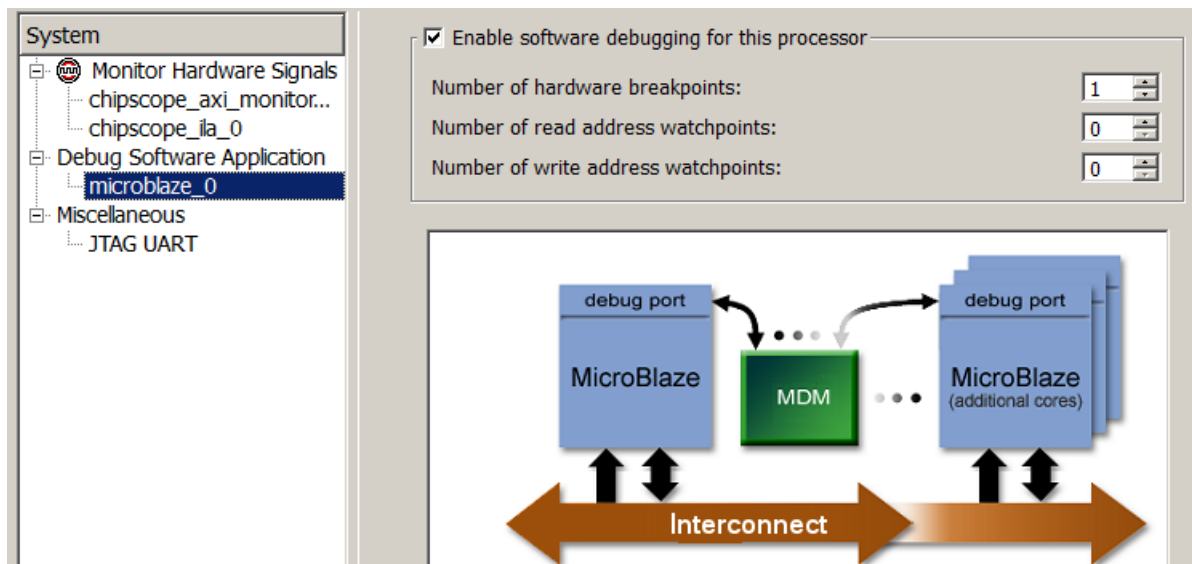


Abbildung 12: Konfiguration des Debuggers

Monitor konfiguriert. Am ersten Trigger-Port sind die globalen Signale, insbesondere der 9-Bit-Bus »TRIG_IN« angeschlossen. Am Trigger-Port für den AXI-Datenbus für Schreiboperationen ist außer den Daten ein Gültigkeits- und ein Bestätigungssignal angeschlossen. Ähnliches gilt für den AXI-Adressbus. Von dem Bestätigungsbus für Schreiboperationen soll im weiteren nur das Gültigkeitssignal interessieren, obwohl hier auch noch weitere Signale angeschlossen sind. Der Aufzeichnungstakt ist der AXI-Bustakt, der im Beispielenwurf gleichfalls der 100MHz-Baugruppentakt ist. Die Ansteuerung und PC-Kommunikation erfolgt über den Logikanalysator-Controller »ICON«, Steuer-Port »CTRL1«.

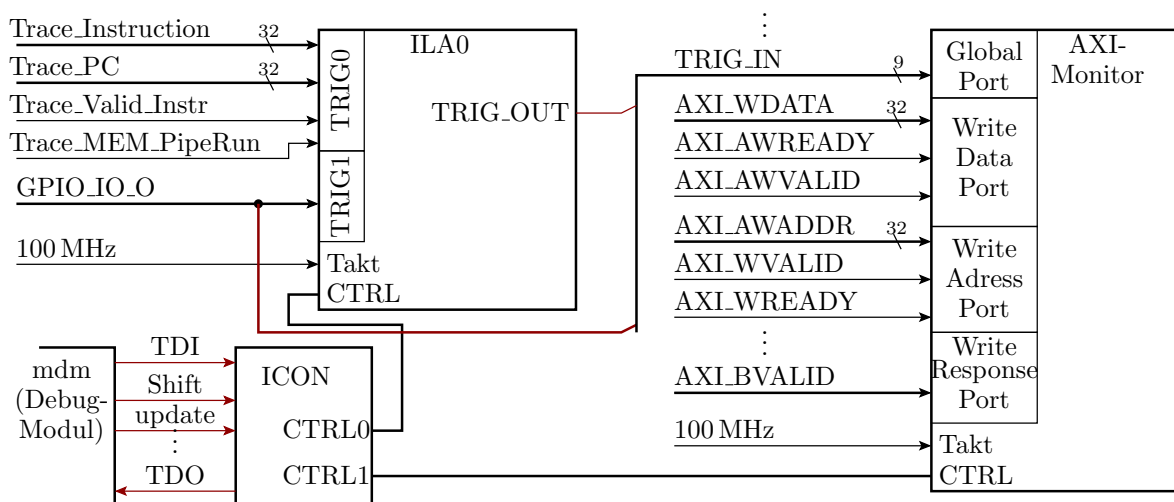


Abbildung 13: Gesamtbeschaltung der eingefügten Logikanalysatoren

Die in Abb. 13 rot eingezeichneten Verbindungen

- »TRIG_OUT« von »ILA1« und die LED-Signale an »TRIG_IN« des AXI-Monitors und
- die Boundary-Scan-Verbindungen vom »mdm« zum »ICON«

fehlen noch und sind, wie im Folgeabschnitt beschrieben wird, manuell im mhs-File zu ergänzen.

3 Ergänzung der fehlenden Verbindungen

Ein »Design Rule Check« nach Schließen des Debug-Konfigurators mit »Run DRCs« liefert die Warnungen in Abb. 14. Die ersten sieben Warnungen besagen, dass am »ICON« die Boundary-Scan Anschlüsse fehlen. Die unteren beiden Warnungen, dass zwei der an den AXI-Monitor angeschlossenen Signale nicht existieren und statt dessen Nullen aufgezeichnet werden, weisen auf eine Unschönheit, aber kein wirkliches Problem hin.

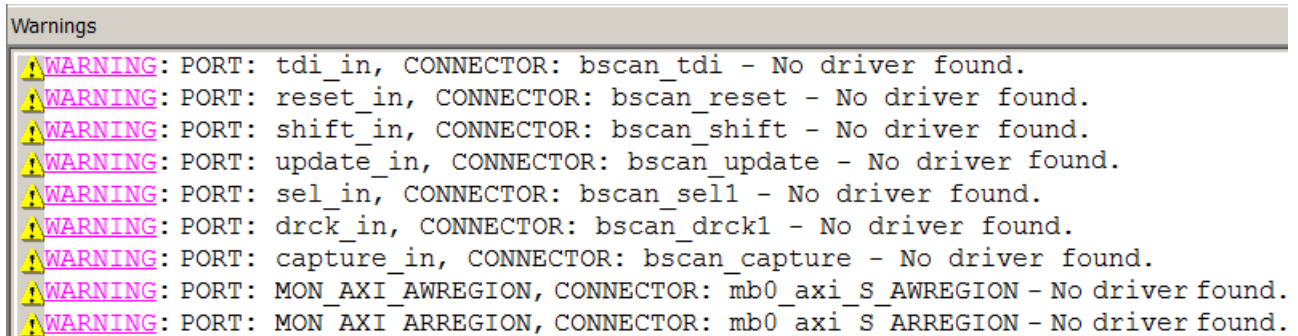


Abbildung 14: Warnungen des »Design Rule Checks« nach Einfügen der Logikanalysatoren

Boundary-Scan ist ein serieller Testbus zur Kommunikation eines Diagnoserechners mit Schaltkreisen einer Baugruppe und in unserem FPGA mit den einzelnen Diagnose- und Programmierereinheiten des Schaltkreises. Ohne Boundary-Scan-Anbindung ist der »ICON« und sind die integrierten Logikanalysatoren nicht vom Arbeitsplatzrechner aus ansteuerbar. In einem integrierten Rechnersystem mit Debugger-Modul »mdm«, wie in unserem Beispiel, erfolgt der Boundary-Scan-Anschluss über dieses, das entsprechend konfiguriert werden muss. Das Konfigurationsfenster ist zu öffnen und unter »Advanced« für »BSCAN location« wie in Abb. 15 »ICON« auszuwählen. Alternativ kann auch im mhs-File für den »mdm« »PARAMETER C_USE_BSCAN = 1« ergänzt werden.

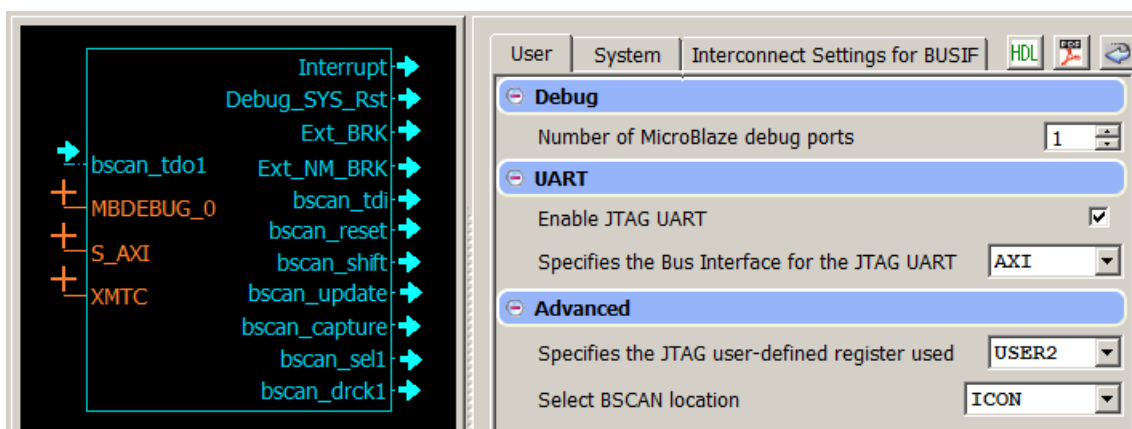


Abbildung 15: Grundeinstellungen des AXI-Monitors

Die fehlenden Verbindungen zwischen Debug-Modul »mdm« und dem Logikanalysator-Controller »chipscope_icon« sind im mhs-File, wie in Abb. 16 gezeigt, zu ergänzen. Fehler bei den Port-Namen führen dazu, dass sich die »Assembly View« bis zu deren Beseitigung nicht mehr öffnen lässt. Nach Speichern des mhs-Files und wiederholtem Aufruf von »Run DRCs« dürfen keine sich auf Boundary-Scan- (BScan-) Anschlüsse beziehende Warnungen mehr erscheinen.


```

90 BEGIN mdm
91 PARAMETER INSTANCE = debug_module
103 # ergänzte BScan-Verbindungen
104 PORT bscan_tdi = v_tdi
105 PORT bscan_reset = v_reset
106 PORT bscan_shift = v_shift
107 PORT bscan_update = v_update
108 PORT bscan_capture = v_capture
109 PORT bscan_sell = v_sell
110 PORT bscan_drck1 = v_drck1
111 PORT bscan_tdo1 = v_tdo
112 END
151 BEGIN chipscope_icon
152 PARAMETER INSTANCE = chipscope_icon_0
157 # ergänzte BScan-Verbindungen
158 PORT tdi_in = v_tdi
159 PORT reset_in = v_reset
160 PORT shift_in = v_shift
161 PORT update_in = v_update
162 PORT capture_in = v_capture
163 PORT sel_in = v_sell
164 PORT drck_in = v_drck1
165 PORT tdo_out = v_tdo
166 END

```

Abbildung 16: Ergänzung der BScan-Verbindungen im mhs-File

Zur Verbindung des Trigger-Ausgangs vom »chipscope_ila« und dem LED-Ausgabeport »axi_gpio« mit dem 9-bit Trigger-Eingang »TRIG_IN« des AXI-Monitors wird im mhs-File in Abb. 17

- in Zeile 119 ein Signal »Led_intern« an den Ausgang »GPIO_IO_O« des LED-Ausgabeport »axi_gpio« und
- in Zeile 16 an den Ausgang »Led« der Gesamtschaltung angeschlossen.
- In Zeile 143 wird an den Trigger-Ausgang »TRIG_OUT« des ersten Logikanalysators »chipscope_ila« ein Signal »ILA0_TRIG_OUT« und
- in Zeile 177 wird an den 9 bit breiten Trigger-Eingang des AXI-Monitors die Konkatenation des Trigger-Signals »ILA0_TRIG_OUT« und des internen LED-Signals »Led_intern« angeschlossen.

```

16 PORT Led = Led_intern, DIR = 0, VEC = [7:0]
111 BEGIN axi_gpio
119 PORT GPIO_IO_O = Led_intern
120 END
131 BEGIN chipscope_ila
143 PORT TRIG_OUT = ILA0_TRIG_OUT
144 END
163 BEGIN chipscope_axi_monitor
177 PORT MON_AXI_TRIG_IN = ILA0_TRIG_OUT & Led_intern
178 END

```

Abbildung 17: Einträge im mhs-File zur Verbindung des Trigger-Eingangs des AXI-Monitors mit dem Trigger-Ausgang des ersten Logikanalysators und den LED-Signalen

4 Erstellen und Test eines Beispielprogramms

Der Hardware-Entwurf kann mit den Dateien »system.bit«, »system.xml« und »system_db.bmm« auf der Web-Seite in der Tabellenzeile zu diesem Projekt wie in Anleitung »Debugger«, Abschn. 2 übersprungen werden. Egal, ob die Hardware-Beschreibung selbst erstellt oder von der Web-Seite übernommen wurde, sind in SDK wie in den ersten beiden Anleitungen beschrieben

- ein »Board Support Package«,
- ein leeres »Application Project« (der Name sei diesmal »CS_Test«) und
- eine C-Quelldatei (der Name sei diesmal »cst_main.c«)

anzulegen. In die C-Quelldatei ist das Programm in Abb. 18 einzugeben bzw. aus der entsprechenden Datei auf der Web-Seite in die Datei zu kopieren².

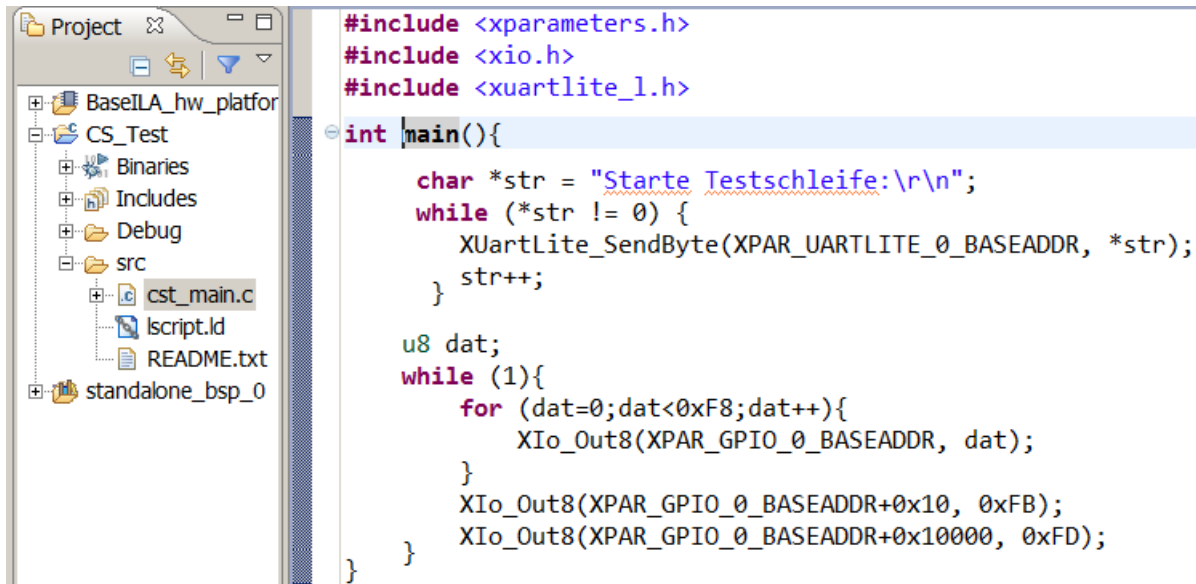


Abbildung 18: Beispielprogramm zum Ausprobieren der integrierten Logikanalysatoren

Das Beispielprogramm gibt einen Text aus, zählt in einer For-Schleife den Ausgabewert für die LEDs von 0 bis 0xF7 hoch. In der äußeren Endlosschleife wird zwischen der Ausgabe von 0xF7 und 0x00 auf die Adresse des Ausgaberegisters, der Wert 0xFB auf eine um zehn höhere Adresse (d.h. einer Adresse, die im Adressbereich der Ausgabeschnittstelle liegt) und der Wert 0xFD auf eine um 0x1000 höhere Adresse außerhalb des Adressbereichs der Ausgabeschnittstelle ausgegeben³.

Nach Anlegen des Gesamtprojekts, der Programmeingabe und der Beseitigung eventueller Fehler erfolgt die Inbetriebnahme und der Programmstart mit denselben Schritten wie in der Anleitung »Minimalsystem«, Abschn. 4 (Software-Entwurf):

- Programmierung der Hardware-Funktion:

Xilinx Tools ▷ Program FPGA

- Einrichten der »Run Configuration« Falls nicht automatisch erfolgt:

Run ▷ Run Configuration

Dabei ist wieder unter »STDIO Connection« der Haken bei »Connect STDIO to Console« zu setzen und »JTAG UART« auszuwählen.

- und starten des Programms mit »Run« bzw. über die Schaltfläche .

²Die C-Quelldatei kann auch von der Web-Seite heruntergeladen, in den Projektordner kopiert und dann in das Projekt eingebunden werden.


³Die beiden Ausgaben auf andere Adressen dienen dazu, bestimmte Eigenarten des AXI-Busses und der Ausgabeschnittstelle zu demonstrieren.

Nach dem Start erscheint auf der Konsole die Ausgabe »Starte Testschleife:«. Danach geht das Programm in die Endlosschleife, deren Signalverläufe zu untersuchen sind. Die Ausgabe zählt dabei so schnell hoch, dass alle acht Leuchtdioden (LEDs) dauerhaft leuchten.

5 Disassemblieren der LED-Ausgabeschleife

Zur Untersuchung und Auswertung der Signalverläufe im Rechner wird das disassemblierte Programm benötigt. Am einfachsten ist das disassemblierte Programm im Debugger einsehbar. Dazu ist die Programmabarbeitung mit dem Debugger zu starten:

Debug ▷ Debug

(oder F11 oder einen Klick auf das Käfersymbol ). Die möglicherweise gestellten Fragen »... terminate previous launch?« und »... switch perspective?« sind wieder mit »Yes« zu beantworten. Nach dem Wechsel in die Debug-Ansicht ist das Fenster mit dem disassemblierten Text zu öffnen:

Windows ▷ Show view ▷ Disassembly

Abb. 19 zeigt den disassembliert Programmausschnitt mit der Ausgabeschleife an die LEDs.

```

0000063c:   sbi r0, r19, 32
00000640:   bri 32 // 0x660 <main+136>
24      XIo_Out8(XPAR_GPIO_0_BASEADDR, dat);
00000644:   imm 16384
00000648:   addik r3, r0, 0 // 0x40000000
0000064c:   lbui r4, r19, 32
00000650:   sbi r4, r3, 0
23      for (dat=0;dat<0xF8;dat++){
00000654:   lbui r3, r19, 32
00000658:   addik r3, r3, 1
0000065c:   sbi r3, r19, 32
00000660:   lbui r4, r19, 32
00000664:   addik r3, r0, 247 // 0xf7 <__do_global_dtors_aux+115>
00000668:   cmpu r18, r4, r3
0000066c:   bgei r18, -40 // 0x644 <main+108>
26      XIo_Out8(XPAR_GPIO_0_BASEADDR+0x10, 0xFB);
00000670:   imm 16384
00000674:   addik r3, r0, 16 // 0x40000010
00000678:   addik r4, r0, -5
0000067c:   sbi r4, r3, 0
27      XIo_Out8(XPAR_GPIO_0_BASEADDR+0x10000, 0xFD);
00000680:   imm 16385
00000684:   addik r3, r0, 0 // 0x40010000
00000688:   addik r4, r0, -3
0000068c:   sbi r4, r3, 0
28      }
00000690:   bri -84 // 0x63c <main+100>

```

Abbildung 19: Disassembliertes Programm in der Debug-Ansicht

Die Befehlsfolge von Adresse 0x644 bis 0x650 schreibt den Wert der Variablen »dat« auf die Ausgabeschneittstelle (Adresse 0x40000000). Die Befehle von 0x654 bis 0x658 erhöhen den Wert von »dat« um eins. Die weitere Befehlsfolge bis zur Adresse 0x66C liest den Wert von »dat«, vergleicht ihn mit 247 (0xf7) und springt, solange er kleiner ist, zum Schleifenanfang (Adresse

0x644). Die Befehlsfolgen auf den Adressen 0x670 bis 0x67C bzw. von 0x680 bis 0x68c geben die Konstanten 0xFB (-5) bzw. 0xFD (-3) auf die Adressen 0x4000010 bzw. 0x4001000 aus. Der Befehl auf Adresse 0x690 ist der Rücksprung am Beginn der Endlosschleife.

Die Alternative zur Erzeugung des disassemblierten Programms ist ein Dump auf der Kommandozeile. Dazu ist die XMD-Konsole zu öffnen mit:

```
Xilinx Tools ▷ Launch Shell
```

Auf der Konsole ist mit »cd« in das Verzeichnis mit dem elf-File⁴ zu wechseln, das sich im Debug-Unterverzeichnis des Software-Projekts befindet. Der Verzeichnisinhalt lässt sich mit »ls« einsehen. Die nachfolgende Kommandozeilenanweisung disassembliert und schreibt das Ergebnis in eine Textdatei »cst_main.asm« (vergl. Abb. 20):

```
mb-objdump -d HalloWelt.elf > cst-main.asm
```



```
C:\Windows\system32\cmd.exe
H:\Softprozessor\SW_BaseILA\CS_Test\Debug>ls
CS_Test.elf          CS_Test.elf.size  objects.mk  src
CS_Test.elf.elfcheck makefile          sources.mk
H:\Softprozessor\SW_BaseILA\CS_Test\Debug>mb-objdump -d CS_Test.elf > disass.txt
H:\Softprozessor\SW_BaseILA\CS_Test\Debug>
```

Abbildung 20: Erzeugen eines Objekt-Dumps auf der XMD-Konsole

Der interessante Bereich ist identisch mit dem aus Abb. 19. Nur die Zwischenzeilen mit den Hochsprachenanweisungen fehlen. Dafür enthält das so erzeugte Assemblerprogramm die Befehlswoorte, die zum Debuggen mit einem Logikanalysator benötigt werden.

Adresse	Befehlswort	Assemblerbefehl	// realisierte Funktion
0x63c:	0xf0130020	sbi r0, r19, 32	
0x640:	0xb8000020	bri 32	// springe zu Adresse 0x660
0x644:	0xb0004000	imm 16384	// mem(0x4000000) := dat
0x648:	0x30600000	addik r3, r0, 0	
0x64c:	0xe0930020	lbui r4, r19, 32	
0x650:	0xf0830000	sbi r4, r3, 0	
0x654:	0xe0730020	lbui r3, r19, 32	// dat := dat+1
0x658:	0x30630001	addik r3, r3, 1	
0x65c:	0xf0730020	sbi r3, r19, 32	
0x660:	0xe0930020	lbui r4, r19, 32	// wenn dat>0xf7 dann
0x664:	0x306000f7	addik r3, r0, 247	// springe 40 Bytes
0x668:	0x16441803	cmpu r18, r4, r3	// zurück zu Adresse 0x644
0x66c:	0xbcb2ffd8	bgei r18, -40	
0x670:	0xb0004000	imm 16384	// mem(0x4000010) := dat
0x674:	0x30600010	addik r3, r0, 16	
0x678:	0x3080fffb	addik r4, r0, -5	
0x67c:	0xf0830000	sbi r4, r3, 0	
0x680:	0xb0004001	imm 16385	// mem(0x4001000) := dat
0x684:	0x30600000	addik r3, r0, 0	
0x688:	0x3080fffd	addik r4, r0, -3	
0x68c:	0xf0830000	sbi r4, r3, 0	
0x690:	0xb800ffac	bri -84	// springe zu Adresse 0x63c

Abbildung 21: Mit »mb-objdump« disassembliertes Programm

⁴ELF (Executable and Linkable Format) – Dateiformat für ausführbare Programme.

6 Untersuchung der Abarbeitung der innersten Schleife

Für die nachfolgenden Experimente ist das Programm in SDK mit »Run« (d.h. ohne Debugger) zu starten. Nachdem das Programm sich auf der Konsole mit »Starte Testscheife:« gemeldet hat, ist die Logikanalysator-Software zu starten. Das erfolgt über »All Programms« und dann weiter wie in Abb. 22 a. Es muss die 64-Bit-Version der Logikanalysator-Software sein.

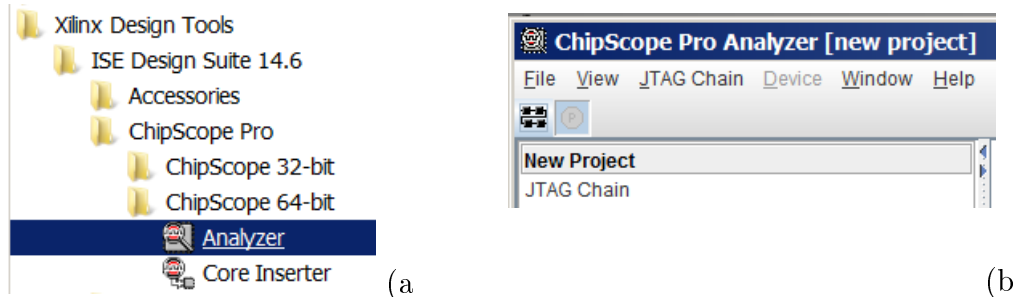



Abbildung 22: a) Start des Chipscope-Analyzers b) Das sich öffnende Fenster

Nach einem Klick auf das Icon  in Abb. 22 b oben links werden alle Schaltkreise der JTAG-Kette angezeigt. Auf unserem Board ist es nur der eine FPGA (Abb. 23).

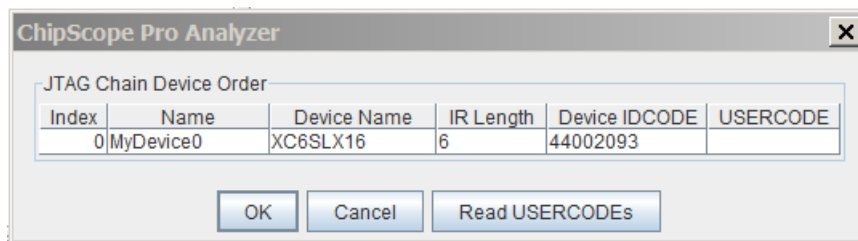



Abbildung 23: Erkannter Schaltkreis am JTAG-Bus

Nach Auswahl mit »OK« werden links im Projektfenster, Abb. 24, die gefunden integrierten Logikanalysatoren des ausgewählten Schaltkreises angezeigt. Im Beispiel ist »MyILA0« der Logikanalysator für die Trace-Signale und »MyILA1« der Logikanalysator für den AXI-Bus. Rechts im Bild ist die Oberfläche für die Trigger-Einstellungen »Trigger Setup« des ersten integrierten Logikanalysators dargestellt. Die initiale Einstellung im Bild, Match-Wert alles »X« (ohne Bedeutung) und Trigger wenn »M0« aktiv ist, bedeutet, dass nach einem Klick auf den Aufzeichnungs-Button  die Aufzeichnung sofort beginnt. Unter dem Fenster für die Trigger-Einstellung werden die (hier ab einem zufälligen Startzeitpunkt) aufgezeichneten Signale dargestellt.

Für die Auswertung der Aufzeichnungsergebnisse ist es zweckmäßig, alle Signale so umzubenennen, wie sie in der Schaltung heißen, und die Einzelbits genau wie in der Hardware-Beschreibung zu Bitvektoren zusammenzufassen. Der Debug-Konfigurator für die integrierten Logikanalysatoren im EDK hinterlegt hierfür im Unterordner »implementation« des Hardware-Projekts im Wrapper-Verzeichnis eines jeden integrierten Logikanalysators eine cdc-Datei, die mit

File ▷ Import

einzulesen ist. Abb. 25 zeigt die Auswahleinträge für die cdc-Dateien beider Logikanalysatoren.

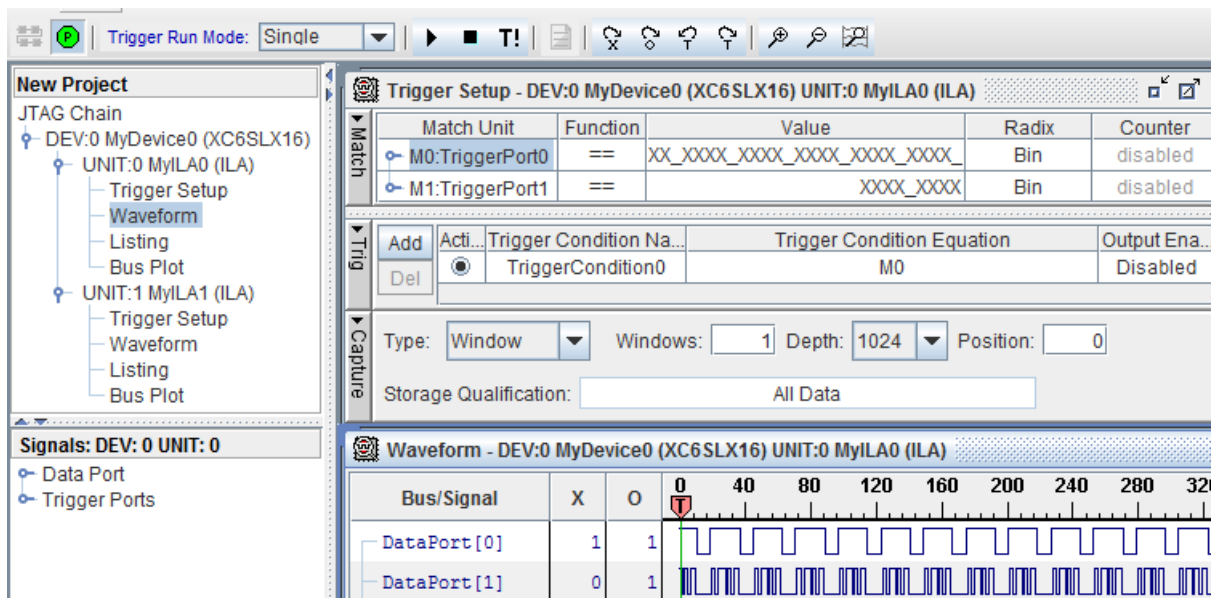


Abbildung 24: Projektbaum beider Logikanalysatoren sowie das Trigger- und das Waveform-Fenster des ersten Logikanalysators mit den initialen Einstellungen

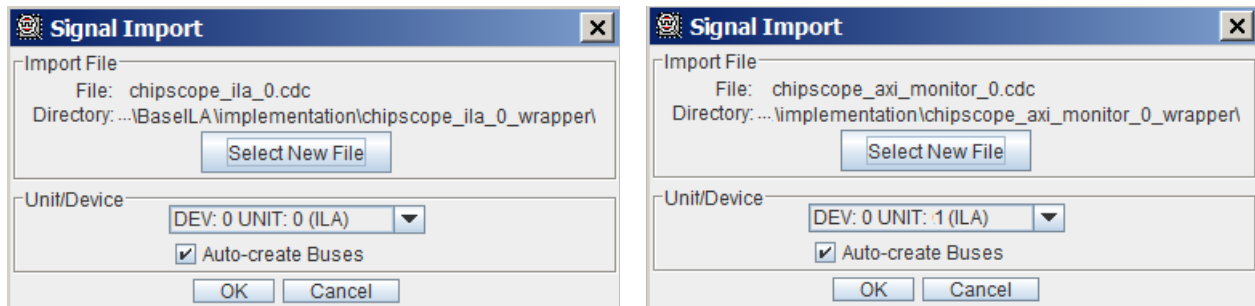



Abbildung 25: Import-Fenster für die cds-Dateien der beiden integrierten Logikanalysatoren

Um einen gewünschten Signalverlaufsabschnitt aufzuzeichnen, ist eine geeignete Trigger-Bedingung festzulegen. Im Fenster »Match« werden die Vergleichswerte für die Signale eingestellt, die zum Trigger-Zeitpunkt definierte Werte haben sollen, und im Fenster darunter die Trigger-Bedingung als logische Verknüpfung von einem oder mehreren Vergleichsergebnissen. Im Beispiel in Abb. 26 soll der Trigger beim erstmaligen Auftreten des LED-Ausgabewerts 0xF7 an »M1:TRIG1« ausgelöst werden. Im Fenster »Capture« darunter ist zusätzlich festgelegt, dass die Aufzeichnung acht Takte vor dem Trigger-Ereignis beginnen und 1024 Werte umfassen soll. Nach einem erneuten und jedem weiteren Klick auf den Aufzeichnungs-Button ► wird ein Signalverlauf aufgezeichnet, bei dem zum Zeitpunkt null (▼) der Ausgabewert an den LEDs, dem die cdc-Datei den Bezeichner »GPIO_IO_O« gegeben hat, nach 0xF7 wechselt. Das Fenster darunter zeigt die mit Zoom vergrößerten Signalverläufe⁵.

Die innere Schleife umfasst nach Abb. 19 in Abschnitt 5 den Programmbereich 0x644

⁵Gelegentlich tritt der Fehler auf, dass der Logikanalysator für einige Samples vernünftige Werte und danach nur noch Einsen aufzeichnet. Das tritt insbesondere auf, wenn nach Start des Logikanalysators das Programm in SDK erneut gestartet wird. Der Workaround dazu ist, die Kabelverbindung zwischen der Chipscope-Software auf dem PC und dem Schaltkreis mit

JTAG Chain ► Close Cable

zu schließen, mit  erneut zu öffnen und die Aufzeichnung mit ► noch einmal zu starten. Die Trigger- und andere Einstellungen bleiben dabei erhalten.

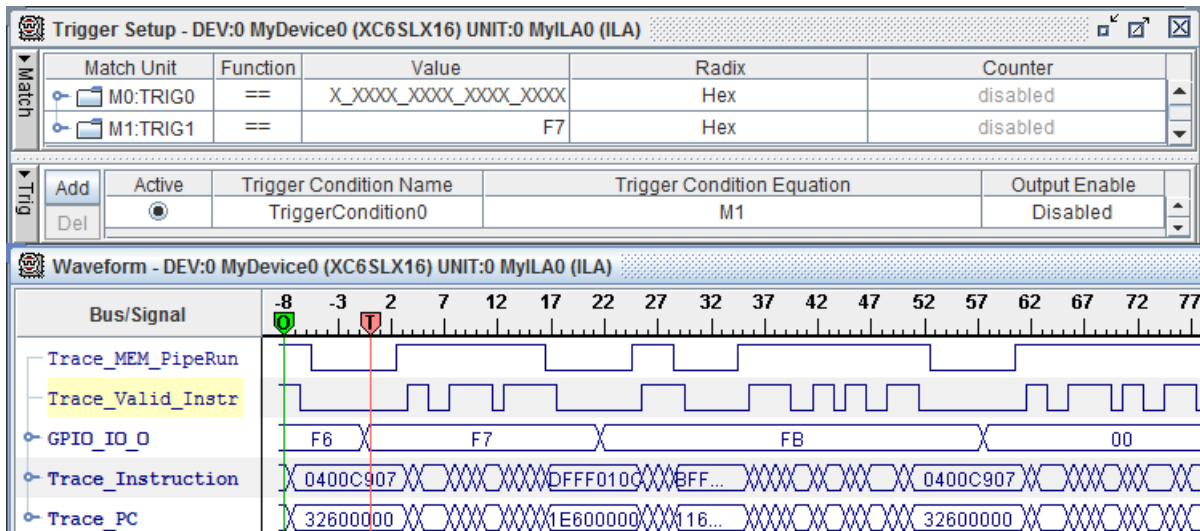


Abbildung 26: Trigger-Einstellungen und aufgezeichnete Trace-Signalverläufe

bis 0x670. Beim Vergleich mit den aufgezeichneten Befehlszählerwerten fällt auf, dass die in Abb. 26 dargestellten Werte bitgespiegelt sind. Eine Umkehrung der Bitreihung in Bussen erfolgt mit Rechtsklick auf den Signalnamen und der Auswahl »Reverse Bus Order«. Auch für »Trace_Instruction« ist die Bitreihung umzukehren, damit die Befehlswoorte denen in Abb. 21 entsprechen. Zur Visualisierung der nur einen Takt anliegenden Adressen und Befehlswoorte ist der Zoom zu vergrößern. Alternativ gibt es in Abb. 24 links unter Trigger und Wave auch die Auswahl »Listing«, in der die aufgezeichneten Signalwerte tabellarisch dargestellt werden. Abb. 27 zeigt das »Listing« für einen Durchlauf der innersten Schleife, beginnend und endend mit einem Sprung zur Adresse 0x604. Ausgewählt wurde dafür der mit dem 100-ten Aufzeichnungswert beginnende Durchlauf, in dem sich der Ausgabewert von 0x01 auf 0x02 erhöht.

Sample	Valid Instr	Trace_PC	Instruction	GPIO_IO_O	MEM Pipe	
100	0	00000670	B0004000	01	1	
101	1	00000644	B0004000	01	1	imm 0x400
102	1	00000648	30600000	01	1	addik r3, r0, 0
103	1	0000064C	E0930020	01	1	lbui r4, r19, 32
104	0	0000064C	E0930020	01	1	
105	0	0000064C	E0930020	01	0	
106	0	0000064C	E0930020	01	0	
107	0	0000064C	E0930020	01	0	
108	0	0000064C	E0930020	01	0	
109	0	0000064C	E0930020	01	0	
110	0	0000064C	E0930020	02	0	
111	0	0000064C	E0930020	02	0	
112	0	0000064C	E0930020	02	0	
113	0	0000064C	E0930020	02	1	
114	1	00000650	F0830000	02	1	sbi r4, r3, 0
115	1	00000654	E0730020	02	1	lbui r3, r19, 32
116	0	00000654	E0730020	02	1	
117	0	00000654	E0730020	02	1	
118	1	00000658	30630001	02	1	addik r3, r3, 1
119	1	0000065C	F0730020	02	1	sbi r3, r19, 32
120	1	00000660	E0930020	02	1	lbui r4, r19, 32
121	1	00000664	306000F7	02	1	addik r3, r0, 0xF7
122	0	00000664	306000F7	02	1	
123	1	00000668	16441803	02	1	cmpu r18, r4, r3
124	1	0000066C	BCB2FFD8	02	1	bgei r18, -40
125	0	00000670	B0004000	02	1	
126	0	00000670	B0004000	02	1	
127	1	00000644	B0004000	02	1	

Abbildung 27: Aufgezeichnete Trace-Signale als Listing

Das Signal »Valid_Instr« signalisiert mit einer »1«, dass die Adresse und das Befehlsword in den nachfolgenden Spalten gültig sind. Den gültigen Befehlen wurden unter Verwendung des disassemblierten Programms aus Abb. 21 Assemblernotationen zugeordnet. Zweierlei ist zu erkennen:

- Die Adressen und Befehle stimmen mit denen aus Abb. 21 überein.
- Manche Befehle werden unmittelbar nacheinander abgearbeitet und zwischen manchen gibt es Lücken.

Insbesondere die große Lücke von »Sample« 105 bis 112 fällt auf. Da in dieser Lücke auch das Trace-Signal »MEM_PipeRun« inaktiv ist, entsteht diese offenbar während der Schreiboperation auf den AXI-Bus. Das Befehlsword dafür »sbi ..« (Speichere r4 als Bytewert auf die Adresse r3+0) erscheint aber im Trace erst nach der Ausgabe (Sample 114). Daraus ist ableitbar, dass die Trace-Signale die Adressen und Befehlswords der abgeschlossenen, und nicht der begonnen oder gerade bearbeiteten Befehle wiedergeben. Anders sieht das bei dem Sprung am Schleifenende aus. Nach dem abgeschlossenen Sprungbefehl erscheint im Trace zweimal das Befehlsword auf Adresse 0x670, d.h. das nach dem Sprungbefehl, das nicht auszuführen ist. Das legt die Vermutung nahe, dass die beiden ungenutzten Takte nach dem Sprungbefehl, in denen das Signal »Valid_Intr« inaktiv ist, zum Sprungbefehl gehören.

7 Untersuchung der AXI-Bus-Signale

In diesem Abschnitt soll untersucht werden, ob es wirklich die Ausgabe an die LEDs ist, die fast 50% der Abarbeitungszeit der innersten Schleife benötigt. Dazu ist das Trigger- und Waveform-Fenster des AXI-Monitors zu öffnen. Die Trigger-Bedingung sei dieselbe wie im ersten Versuch (LED-Ausgabewert gleich 0xF7). In dem in Abb. 25 b importierten cdc-File gibt es die LED-Anschlüsse noch nicht, sondern noch die neun ».._TRIG_IN« Anschlüsse an »CH: 47« bis »CH: 55«, die wie in Abb. 28 b umzubenennen sind. Zusätzlich ist »CH: 55«, an dem der Trigger-Ausgang des ersten Logikanalysators angeschlossen ist, mit Rechtsklick und »Remove from Bus« in ein Einzelsignal umzuwandeln. Die selben Umbenennungen sind auch darunter bei den Trigger-Ports nochmal erforderlich.

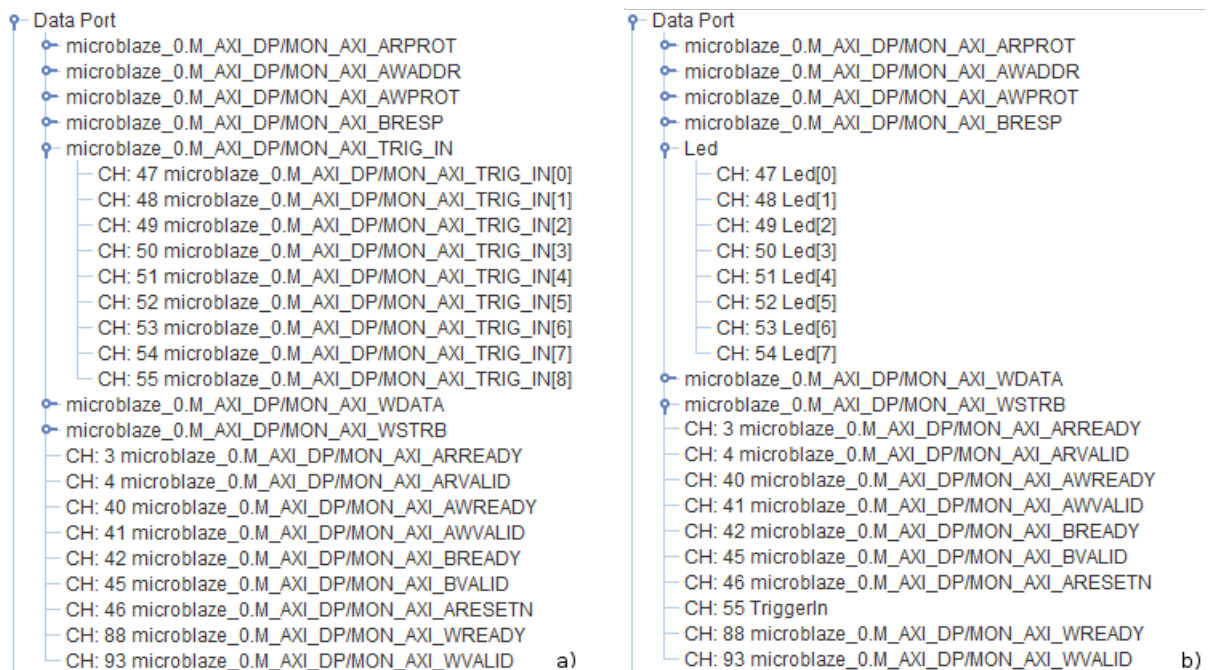


Abbildung 28: Umbenennung der Signale am Trigger-Eingang des AXI-Monitors

Die LED-Signale, auf die getriggert werden soll, gehören zur Match-Gruppe »M4:MON-_AXI_GLOBAL« (Abb. 29). Der Vergleichswert $0b11110111=0xF7$ ist den mittleren acht Bits zuzuordnen. Der Vergleichswert für die beiden anderen Bits ist »X« (beliebig). Aufzeichnungstiefe sei wie im ersten Versuch 1024 und Aufzeichnungsbeginn acht Takte vor dem Trigger-Ereignis.

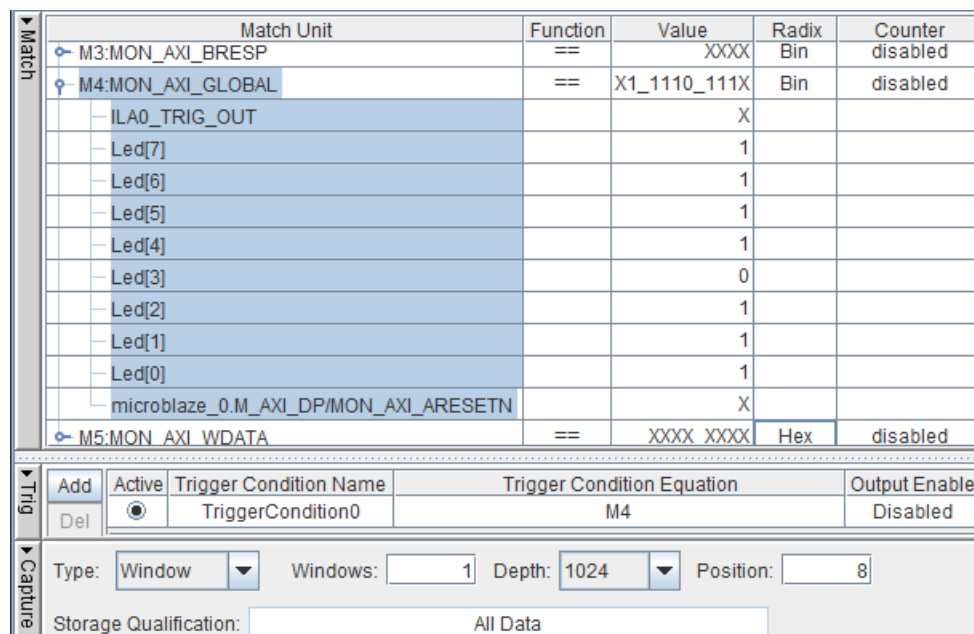


Abbildung 29: Trigger-Einstellung für den AXI-Monitor

Nach der Trigger-Einstellung ist die Aufzeichnung mit ▶ zu starten. Abb. 30 zeigt einen Ausschnitt der aufgezeichneten Daten. Die nicht dargestellten Signale inkl. der höherwertigen 24 Datenbits von »..._WDATA« wurden mit Rechtsklick und »Remove from Viewer« aus der Darstellung entfernt.

Die Ausgabeadresse und die Ausgabedaten liegen bereits fünf Takte vor der LED-Ausgabe auf den entsprechenden AXI-Teilbussen an. Die Adresse wird durch »..._AWVALID« drei Takte

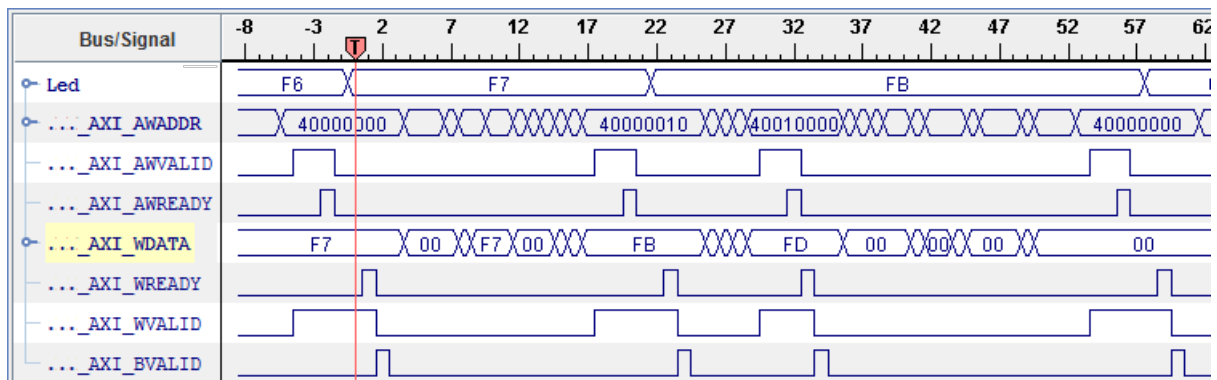


Abbildung 30: Aufgezeichnete AXI-Bussignale

für gültig erklärt. Das Adressgültigkeitssignal schaltet nach Aktivierung des Bestätigungssignals »..._AWREADY« ab. Das Gültigkeitssignal für die Daten »..._WVALID« ist etwas länger aktiv und schaltet gleichfalls nach der Aktivierung seines Bestätigungssignals, hier »..._WREADY« ab. Die Bestätigung der Adresse erfolgt vor der Ausgabe an die LEDs, d.h. die Adresse bzw. die Ausgabefreigabe muss noch irgendwo zwischengespeichert werden. Die Daten sind bis nach der Ausgabe gültig und müssen nicht zwischengespeichert werden. Die Ausgabe von 0xFB auf die Adresse 0x40000010 verursacht denselben Zeitablauf und ändert gleichfalls den LED-Ausgabewert. Bei der Ausgabe auf die Adresse 0x4001000, die nicht zum Adressbereich der Ausgabereinheit gehört, bleibt das Ausgaberegister unverändert und das Bestätigungssignal kommt früher. Der AXI-Bus hat offenbar für die Ausgabe eines Bytes in ein Register ein recht kompliziertes und zeitaufwändiges Protokoll.

8 Kopplung beider Logikanalysatoren

Als nächstes soll untersucht werden, wie die Trace-Signale des Prozessors und die AXI-Bus-Signale zeitlich zueinander ausgerichtet sind. Dazu müssen beide Logikanalysatoren gleichzeitig getriggert werden. Für den AXI-Monitor soll der Trigger wie in Abb. 31 so umgestellt werden, dass die Aufzeichnung 8+10 Takte⁶ vor Aktivierung des Trigger-Ausgabesignals des ersten Logikanalysators beginnt.

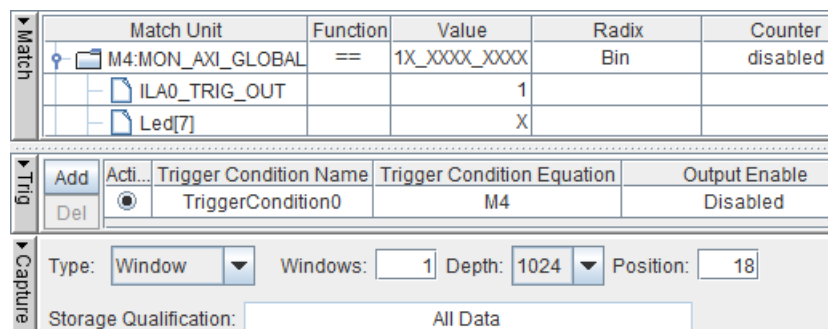


Abbildung 31: Trigger-Einstellung des AXI-Monitors für die gekoppelte Aufzeichnung

Für den ersten Logikanalysator soll die Trigger-Bedingung das erste Auftreten des LED-Werts 0xF7 bleiben. Nur der Trigger-Ausgang ist auf »Pulse (High)« umzustellen (Abb. 32).

Zu Aufzeichnung ist zuerst das Trigger-Fenster des AXI-Monitors auszuwählen und mit ▶ die Aufzeichnung für den AXI-Monitor zu starten. Dann ist das Trigger-Fenster des ersten

⁶Der AXI-Monitor zeichnet gegenüber ILA0 aus einem nicht näher erkennbaren Grund mit 10 Takten Verzögerung auf.

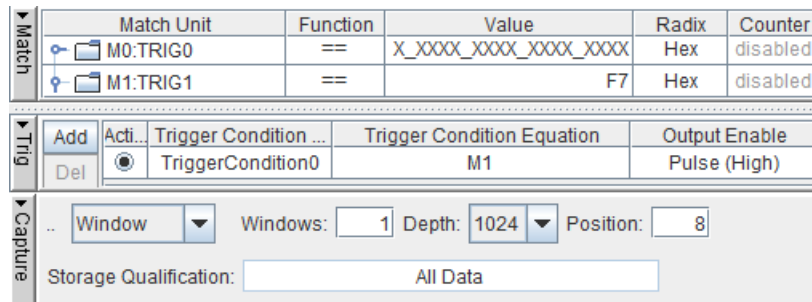


Abbildung 32: Trigger-Einstellung des ersten Logikanalysators für die gekoppelte Aufzeichnung

Logikanalysators auszuwählen und gleichfalls mit ▶ die Aufzeichnung für diesen zu starten. Wenn dieser triggert, wird über den Trigger-Ausgang der AXI-Monitor mitgestartet. Abb. 33 zeigt ausgewählte Signalwerte als Listing für denselben »Sample«-Bereich wie in Abb. 27. In beiden Aufzeichnungen wechselt die LED-Ausgabe in »Sample« 118 von »01« nach »02«. In »Sample« 121 ist im Trace-Signal die Ausgabe abgeschlossen und in »Sample« 111 der Befehl davor. In »Sample« 113 wird das Trace-Signal »MEM_PipeRun« (MP_RUN) deaktiviert, was vermutlich bedeuten soll, dass die Ausgabe-Pipeline anhält. Einen Schritt später sind die Daten auf dem AXI-Bus gültig. Das Gültigkeitssignal »WVALID« wird nach Aktivierung des Bestätigungssignals »WREADY« deaktiviert. Im Folgetakt wird das Signal »BVALID« für die Schreibbestätigung für einen Takt gültig und erst im nächsten Takt ist die Ausgabe eines Bytes in ein Register für die LED-Anzeige abgeschlossen. Das AXI-Protokoll verlangt offenbar, dass die Ausgabewerte bereits ab vier Takte vor der Ausgabe auf dem Bus AXI-Bus anliegen müssen und der Prozessor noch drei Takte nach der Ausgabe auf die Bestätigung wartet.

Listing - DEV:0 MyDevice0 (XC6SLX16) UNIT:0 MyILA0 (ILA)							Listing - DEV:0 MyDevice0 (XC6SLX16) UNIT:1 MyILA1						
<->	Sample	ValInstr	Trace_PC	Instruct	Led	MP Run	<->	Sample	WDATA	WVALID	WREADY	BVALID	Led
	100	1	00000658	30630001	01	1		100	02	0	0	0	01
	101	1	0000065C	F0730020	01	1		101	01	0	0	0	01
	102	1	00000660	E0930020	01	1		102	01	0	0	0	01
	103	1	00000664	306000F7	01	1		103	01	0	0	0	01
	104	0	00000664	306000F7	01	1		104	F6	0	0	0	01
	105	1	00000668	16441803	01	1		105	00	0	0	0	01
	106	1	0000066C	BCB2FFD8	01	1		106	00	0	0	0	01
	107	0	00000670	B0004000	01	1		107	00	0	0	0	01
	108	0	00000670	B0004000	01	1		108	00	0	0	0	01
	109	1	00000644	B0004000	01	1		109	F7	0	0	0	01
	110	1	00000648	30600000	01	1		110	02	0	0	0	01
	111	1	0000064C	E0930020	01	1		111	02	0	0	0	01
	112	0	0000064C	E0930020	01	1		112	02	0	0	0	01
	113	0	0000064C	E0930020	01	0		113	02	0	0	0	01
	114	0	0000064C	E0930020	01	0		114	02	1	0	0	01
	115	0	0000064C	E0930020	01	0		115	02	1	0	0	01
	116	0	0000064C	E0930020	01	0		116	02	1	0	0	01
	117	0	0000064C	E0930020	01	0		117	02	1	0	0	01
	118	0	0000064C	E0930020	02	0		118	02	1	0	0	02
	119	0	0000064C	E0930020	02	0		119	02	1	1	0	02
	120	0	0000064C	E0930020	02	0		120	02	0	0	1	02
	121	0	0000064C	E0930020	02	1		121	02	0	0	0	02
	122	1	00000650	F0830000	02	1		122	00	0	0	0	02
	123	1	00000654	E0730020	02	1		123	00	0	0	0	02
	124	0	00000654	E0730020	02	1		124	00	0	0	0	02
	125	0	00000654	E0730020	02	1		125	00	0	0	0	02
	126	1	00000658	30630001	02	1		126	03	0	0	0	02
	127	1	0000065C	F0730020	02	1		127	02	0	0	0	02

Abbildung 33: Zeitgleich aufgezeichnete Signale beider Logikanalysatoren

Das erklärt zu einem großen Teil, wenn auch nicht vollständig, warum die Ausgabe so lange dauert. Der AXI-Bus ist definitiv nicht die ideale Lösung für ein einfaches Rechnersystem.