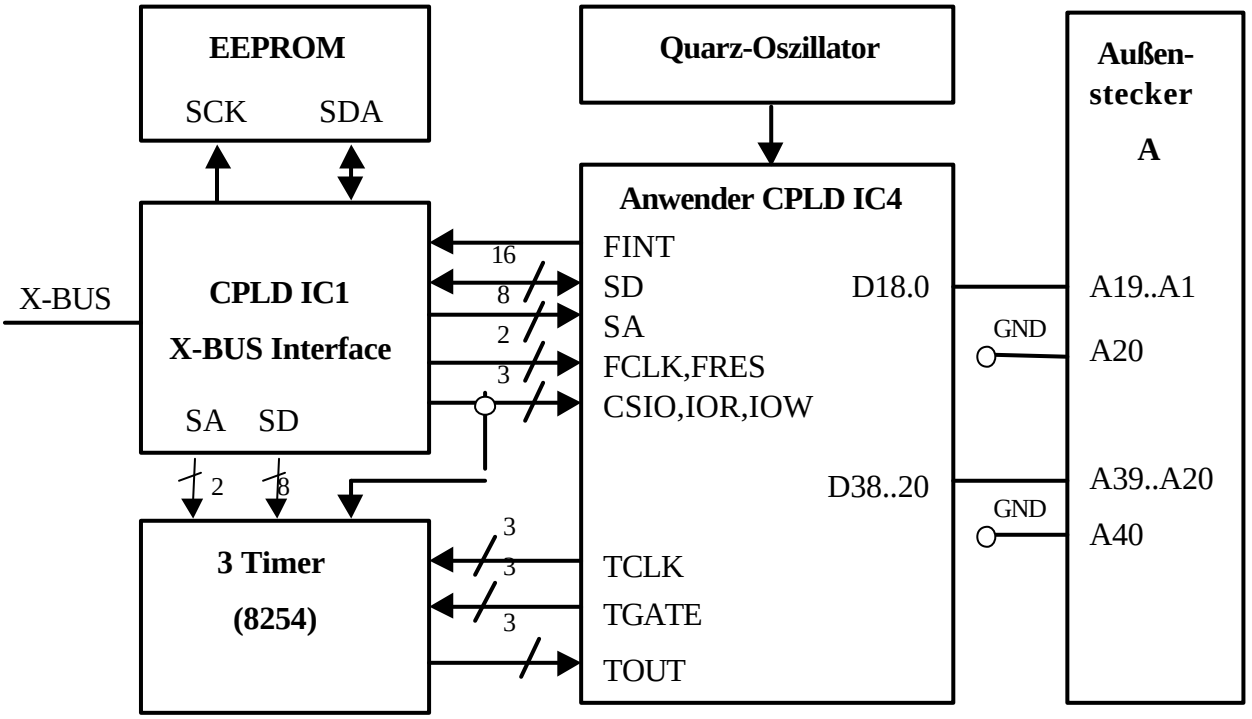
Application Note AN092
Eigene Hardwaredesigns für das MAX-Modul X-CPLD-38

Inhaltsverzeichnis

Übersicht über den Aufbau des Moduls X-CPLD-38



Eigenschaften des Moduls X-CPLD-38

Das Modul X-CPLD-38 bietet die Möglichkeit, eigene Hardwaredesigns zu entwickeln und diese in die X-BUS Welt zu integrieren. Auf dem Modul befindet sich ein frei programmierbares CPLD (= IC4) von XILINX mit 288 Makrozellen (XC95288XL), das über ein weiteres CPLD (= IC1) an den X-BUS angekoppelt ist. Der dem Anwender zur Verfügung stehende Funktionsbereich im I/O Bereich des X-BUS umfaßt 224 Byte. Es können wahlweise Wort- oder Byte-Zugriffe durchgeführt werden.

Außerdem sind 38 Pins des CPLDs nach außen auf den Modul-Stecker A geführt und können somit direkt als Aus- oder Eingänge verwendet werden.

Verwendung des 82C54 Timer-Chips von Intel

Auf dem Modul befindet sich außerdem noch ein Timer Baustein. Dabei handelt es sich um den 82C54-Chip von Intel. Die Programmierung erfolgt über das CPLD IC1. Die jeweils 3 Clock-, Gate- und Out-Leitungen des Timer-Chips sind direkt zum CPLD IC4 durchgeschleift und können vom Anwender frei benutzt werden. Für die Clock-Leitungen steht dem Anwender eine von außen eingespeiste 1 MHz Clock zur Verfügung.

Alles Weitere über die Verwendungsmöglichkeiten der Timer kann aus dem Datenblatt entnommen werden. Herunterladbar z.B. unter:

www.developer.intel.com/design/periphrl/datashts/231244.htm

Programmiersoftware

Zum Programmieren des CPLDs kann das Programmpaket WebPACK von XILINX verwendet werden. Es enthält auch eine kostenlose Programmiersoftware, die Designs in VHDL, Verilog oder ABEL ermöglicht.

Herunterladbar über: www.xilinx.com

Pinbelegung des Kunden-programmierbaren CPLDs

Es muß beim Programmieren des CPLDs IC4 darauf geachtet werden, daß die Pinbelegung des gefitteten Designs zur Verschaltung des CPLDs paßt. Dafür gibt es von SORCUS .ucf-Files, xcpld38a.ucf für ABEL-Designs und xcpld38v.ucf für VHDL-Designs. Beide Files können über die SORCUS Homepage heruntergeladen werden. Das entspechende .ucf-File kopiert man in das Projektverzeichnis, indem das Design entstehen soll, und gibt ihm den selben Namen, den auch das Projekt trägt. Führt man nun einen „Fitting“-Vorgang durch, so werden die Informationen dieses Files automatisch in das Projekt übernommen. Zur Sicherheit empfiehlt es sich, vor dem Programmieren des Designs in das CPLD im „Fitting“-Report nachzuschauen, ob die Pinbelegung auch tatsächlich korrekt ist, da sonst durch eventuell verursachte Kurzschlüsse Schäden am Modul entstehen können.

Im folgenden ist das xcplda.ucf-File für ABEL abgedruckt. Benutzt man VHDL, so müssen alle numerierten Pins geändert werden : sd3 -> sd<3>.

```
#PINLOCK_BEGIN

NET "ior"      LOC = "S:PIN2";
NET "d36"      LOC = "S:PIN4";
NET "d3"       LOC = "S:PIN5";
NET "sd3"      LOC = "S:PIN7";
NET "d17"      LOC = "S:PIN10";
NET "d1"       LOC = "S:PIN11";
NET "d21"      LOC = "S:PIN12";
NET "d22"      LOC = "S:PIN13";
NET "d23"      LOC = "S:PIN14";
NET "d2"       LOC = "S:PIN15";
NET "sd1"      LOC = "S:PIN17";
NET "sd6"      LOC = "S:PIN20";
NET "d0"       LOC = "S:PIN21";
NET "d12"      LOC = "S:PIN22";
NET "d15"      LOC = "S:PIN23";
NET "d16"      LOC = "S:PIN24";
NET "d20"      LOC = "S:PIN25";
NET "fclk"     LOC = "S:PIN30";
NET "d24"      LOC = "S:PIN35";
NET "d25"      LOC = "S:PIN39";
NET "d37"      LOC = "S:PIN40";
NET "d4"       LOC = "S:PIN41";
NET "d5"       LOC = "S:PIN43";
NET "d31"      LOC = "S:PIN46";
NET "d11"      LOC = "S:PIN49";
NET "d13"      LOC = "S:PIN51";
NET "d14"      LOC = "S:PIN52";
NET "d27"      LOC = "S:PIN53";
NET "d7"       LOC = "S:PIN54";
NET "sd9"      LOC = "S:PIN58";
NET "sd11"     LOC = "S:PIN60";
NET "sd7"      LOC = "S:PIN66";
NET "sd15"     LOC = "S:PIN69";
NET "sd14"     LOC = "S:PIN70";
NET "fint"     LOC = "S:PIN71";
NET "sa7"      LOC = "S:PIN91";
NET "d29"      LOC = "S:PIN79";
NET "tclk2"    LOC = "S:PIN80";
NET "d38"      LOC = "S:PIN81";
NET "sd2"      LOC = "S:PIN82";
NET "d9"       LOC = "S:PIN83";
NET "tclk1"    LOC = "S:PIN86";
NET "tgate2"   LOC = "S:PIN88";
NET "sa6"      LOC = "S:PIN138";
NET "d26"      LOC = "S:PIN92";
NET "d35"      LOC = "S:PIN93";
NET "d6"       LOC = "S:PIN94";
NET "d10"      LOC = "S:PIN98";
NET "d18"      LOC = "S:PIN100";
NET "d33"      LOC = "S:PIN102";
NET "´sd0"     LOC = "S:PIN103";
NET "iow"      LOC = "S:PIN104";
NET "tgate0"   LOC = "S:PIN105";
NET "tgate1"   LOC = "S:PIN106";
NET "tclk0"    LOC = "S:PIN107";
NET "sd13"     LOC = "S:PIN111";
NET "csio"     LOC = "S:PIN112";
NET "qclk"     LOC = "S:PIN113";
```

```
NET "tout1"     LOC =  "S:PIN115";
NET "tout0"     LOC =  "S:PIN116";
NET "sa3"       LOC =  "S:PIN128";
NET "sd12"      LOC =  "S:PIN118";
NET "d32"       LOC =  "S:PIN119";
NET "d34"       LOC =  "S:PIN120";
NET "led"       LOC =  "S:PIN121";
NET "sa2"       LOC =  "S:PIN129";
NET "sa1"       LOC =  "S:PIN130";
NET "sa0"       LOC =  "S:PIN74";
NET "d28"       LOC =  "S:PIN131";
NET "d8"        LOC =  "S:PIN132";
NET "sd4"       LOC =  "S:PIN133";
NET "sd8"       LOC =  "S:PIN134";
NET "sd5"       LOC =  "S:PIN135";
NET "sa4"       LOC =  "S:PIN117";
NET "sd10"      LOC =  "S:PIN137";
NET "sa5"       LOC =  "S:PIN136";
NET "d30"       LOC =  "S:PIN139";
NET "tout2"     LOC =  "S:PIN140";
NET "fres"      LOC =  "S:PIN143";

#PINLOCK_END
```

Beispieldesign

Als Ausgang für eine Eigenentwicklung steht ein Beispieldesign in ABEL und in VHDL zur Verfügung, welches die Verwendung der Register, das Auslesen der Eingänge, das Schalten der Ausgänge, die Verwendung der Modul-LED und die Einbindung von Interrupts erläutert. Das Beispieldesign kann von der SORCUS Homepage heruntergeladen werden: www.sorcus.com.

Verwendung von Interrupts im eigenen Design

Es ist möglich ein oder mehrere der digitalen Eingänge als Interruptquellen zu verwenden. Hierfür ist eine Interruptleitung (FINT) vom Kunden-programmierbaren CPLD zu demjenigen CPLD, welches die Anbindung an den X-BUS vornimmt, geführt. Bei der Aufschaltung von Interrupts auf diese Leitung sind für einen reibungslosen und geschwindigkeitsoptimierten Ablauf einige Vorschriften zu beachten:

1) Es können maximal 16 voneinander unterscheidbare Interruptquellen verwendet werden.
2) Alle Interruptquellen müssen miteinander verodert und auf die Interruptleitung aufgeschaltet werden. Auf diese Weise ist die Interruptleitung solange aktiv, bis kein Interrupt mehr vorliegt.
3) Alle 16 Interrupt-Pending Bits, die einen Interrupt auslösen können, müssen in ein Interrupt-Register gelegt werden. Dieses muß unter der Adresse 10h im Kunden-CPLD realisiert werden.
4) Es muß garantiert werden, daß alle verwendeten Pending-Bits durch einen Lese-Zugriff auf das Interrupt-Register automatisch gelöscht werden, und daß dadurch die Interruptleitung zurückgenommen wird.

Wie diese Forderungen realisiert werden können, ist im Beispieldesign gezeigt.

Sind diese Voraussetzungen erfüllt, so läuft die Interruptbehandlung wie folgt ab. Zuerst löst z.B. eine Flanke an einem digitalen Eingang einen Interrupt aus. Daraufhin wird die Interruptleitung aktiv und das X-BUS CPLD löst einen Interrupt auf der Basiskarte aus. Ein Modul-Device-Treiber liest das Interrupt-Register aus und löscht damit automatisch alle aufgetretenen Interrupts. Nun ruft er alle Anwenderroutinen auf, die unter dem oder den aufgetretenen Interrupts eingehängt sind. Die Anwenderroutinen führen die gewünschten Aktionen aus und warten anschließend auf den nächsten Interrupt.

Anschlußbelegung am Modul-Stecker

38 Pins des CPLDs sind direkt auf den Modul-Stecker herausgeführt:

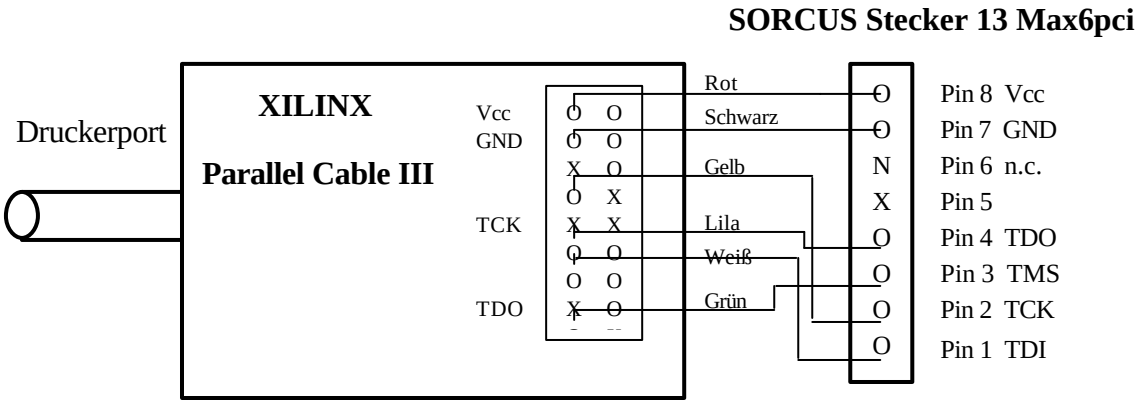
D18..D0 -> Steckerpin A19..A1
GND -> Steckerpin A20
D38..D20 -> Steckerpin A39..A21
GND -> Steckerpin A40

Programmieren mittels XILINX Parallel Cable III

Programmieren läßt sich das CPLD IC4 mit Hilfe einer Programmiersoftware, die im WebPACK Paket enthalten ist. Startet man diese Software, so werden beide CPLDs (IC1 und IC4) des Moduls angezeigt. Für das erste CPLD wählt man ein „Dummy"-Design (dummy.jed) aus, welches sich wie das Beispieldesign über unsere Homepage herunterladen läßt. Ausgewählt und programmiert wird allerdings nur das zweite CPLD mit dem selbst erzeugten JEDEC-File

Des weiteren benötigt man einen Programmieradapter: XILINX Parallel Cable III. Dieser kann über SORCUS bezogen werden unter der Bestellnummer KF-3199. Der Programmieradapter wird an den Druckerport des Rechners gesteckt und bietet am anderen Ende zwei 9 polige Steckerleisten mit jeweils 6 herausgeführten Pins. Verwendet wird die Steckerleiste, die für JTAG vorgesehen ist. Diese Pins werden über ein Kabel mit dem JTAG Stecker für das Modul verbunden. Die Pinbelegung der JTAG Stecker der SORCUS Produkte werden wie folgt mit dem XILINX Adapter verbunden:

Max6Pci: Im Falle der Max6Pci-Karte wird als JTAG-Stecker Stecker 13 auf der Max6pci verwendet. Beim Programmiervorgang muß die Max6pci Karte mit Strom versorgt sein, das zu programmierende Modul X-CPLD-38 muß sich auf Steckplatz 6 befinden. Die anderen Steckplätze dürfen nicht besetzt sein.



X: Pin nicht vorhanden

N: Pin wird nicht angeschlossen

Historie dieses Dokumentes

3.3.03	dh	Erste Version der Application Note AN092_C.doc fertiggestellt.
27.2.03	dh	Ein Abschnitt zur Interruptverwendung wurde hinzugefügt. Das Beispieldesign wurde entsprechend abgeändert.
9.4.02	dh	xd403a2a.abl mußte geändert werden, da Timer-Zugriffe auf ungerade Adressen nicht funktionierten. Das neue Design xd403a2b.abl bedingt zwar kein neues Beispieldesign für das IC4, aber die Pinbelegung, und damit die .ucf Files müssen geändert werden.