USB Core

Dokumentation

Version 1.0

Inhalt

Änderungen

Datum	Version	Bemerkung
05.09.2007	1.0	Dokument erstellt

Allgemeines

Der Usb Core ermöglicht eine einfache und schnelle Kommunikation zwischen einem USB Host und dem FPGA Design.

- Keine CPU notwendig
- USB 2.0 High Speed
- Hoher Datendurchsatz
- Einfache Anbindung der Daten über Fifo Interface
- Zusätzlicher Control Bus für Status und Steuerungsinformation

LibUsb

LibUsb und LibUsb-Win32 sind Open Source Projekte.

Die entsprechenden Seiten sind

http://libusb.sourceforge.net/
http://libusb-win32.sourceforge.net/

LibUsb stellt ein plattformunabhängige Schnittstelle zur Kommunikation über Usb zur Verfügung.
LibUsb-Win32 ist eine Implementierung von LibUsb für Windows.

LibUsb kann verwendet werden, um den Usb Core anzusprechen. Die beiden Projekte stehen aber sonst in keinem Zusammenhang.
Der USB Core wurde mit LibUsb Win-32 mit der Version 1.0.10.0 unter Windows XP getestet.

Kurze Beschreibung der Signale

UTMI Interface		
clk	Input	60 MHz Takt. Das Signal muss mit einem IBUFG aus dem Signal CLKOUT erzeugt werderden. Alle folgenden Signale sind synchron zu diesem Takt.
DATABUS16_8	Output	Verbindung mit Tranceiver Chip
RESET	Output	Verbindung mit Tranceiver Chip
XCVRSELCT	Output	Verbindung mit Tranceiver Chip
TERMSELECT	Output	Verbindung mit Tranceiver Chip
OPMODE0	Output	Verbindung mit Tranceiver Chip
OPMODE1	Output	Verbindung mit Tranceiver Chip
LINESTATE	Input	Verbindung mit Tranceiver Chip
TXVALID	Output	Verbindung mit Tranceiver Chip
TXREADY	Input	Verbindung mit Tranceiver Chip
VALIDH	Output	Verbindung mit Tranceiver Chip
RXVALID	Output	Verbindung mit Tranceiver Chip
RXACTIVE	Input	Verbindung mit Tranceiver Chip
RXERROR	Input	Verbindung mit Tranceiver Chip
DOut	Output	Verbindung mit Tranceiver Chip D[15:8]
Din	Input	Verbindung mit Tranceiver Chip D[7:0]
Triansmit Fifo (IN endpoint, EP1)		
TxFifoWrite	Input	Strobe Signal. Muss gleichzeitig mit den Daten aktiv sein.
TxFifoData	Input	8 bit Daten
TxFifoFull	Output	TxFifo ist fast voll (8 Byte Puffer)
Receive Fifo (OUT endpoint, EP2)		
RxFifoRead	Input	Strobe Signal zum lesen. Die Daten erscheinen einen Taktzyklus nach dem Strobe Signal.
RxFifoData	Output	8 bit Daten
RxFifoEmpty	Output	RxFifo ist leer
Control Bus (zusätzliche Status- und Steuersignale)		
CtrlMainIndex	Input	Teil der Adressen
CtrlSubIndex	Input	Teil der Adressen
CtrlWriteNow	Input	Write-Signal. Index und Daten müssen gleichzeitig gültig sein.
CtrlWriteData	Input	32 bit Daten, die der PC schreibt
CtrlReadData	Output	32 bit Daten, die der PC liest. Müssen gleichzeitig mit dem Index gültig sein.
Sonstige Signale		
UsbReset	Output	Hgh bei USB Reset Kommando
HighSpeed	Output	High wenn High Speed, Low bei Full Speed

Detaillierte Beschreibung der Signale und Funktionen

UTMI Interface

Die Signale müssen mit dem Tranceiver Chip verbunden werden.
Es gelten hierbei folgende Timing Constraints:

```
NET clk PERIOD= 60 MHz;
OFFSET = IN 12 ns BEFORE "CLKOUT";
OFFSET = OUT 12 ns AFTER "CLKOUT";
```

Transmit Fifo

Der Transmit Fifo ermöglicht es dem FPGA Design, Daten über den IN Endpoint zu übertragen. Die Nummer des IN Endpoints ist 1.
Die Größe des Transmit Fifos ist 2048 Bytes.

Die 8 bit breiten Daten TxFifoData werden mit TxFifoWrite in den Fifo getaktet.

Das Signal TxFifoFull weißt eine Besonderheit auf: Es ist bereits high, wenn noch 8 Werte im Fifo Platz haben.
Dieses Verhalten erleichtert das zusammenhängende Schreiben von Werten, die breiter als 8 bit sind.

Die geschriebenen Daten können in der Software 1:1 gelesen werden.

```
ret= usb_bulk_read(handle, 0x81, buffer, length, msTimeout);
```

Der Rückgabewert gibt die Anzahl der tatsächlich gelesenen Bytes an.
Die Blocklänge des IN Endpoints ist 1024 bytes.

Achtung:
Wenn nicht bekannt ist, wie viel Daten der Endpoint zur Verfügung stellen kann, dann muss die zu lesende Länge ein Vielfaches von 1024 sein.

Receive Fifo

Über den Receive Fifo liest das FPGA Design Daten, die über den OUT Endpoint geschrieben wurden.
Die Nummer des Out Endpoints ist 2.

Die Daten RxFifoData erscheinen einen Taktzyklus nachdem das Signal RxFifoRead. (Kein „First Word Fall through“).
RxFifoEmpty ist high, wenn sich keine Daten im Receive Fifo befinden.

So werden die Daten in der Software in den OUT Endpoint geschrieben:

```
ret= usb_bulk_write(handle, 0x2, buffer, length, msTimeout);
```

Der Rückgabewert gibt die Anzahl der tatsächlich geschriebenen Bytes zurück.

Die Blocklänge des OUT Endpoints ist 512 byte.

Achtung:
Wenn nicht bekannt ist, wie viel Daten der Endpoint aufnehmen kann, dann muss die Länge kleiner oder gleich 512 byte sein.
Wenn der Endpoint nicht alle Daten aufnehmen kann, dann kehrt die Funktion mit einem Fehlercode zurück. Die Anzahl der tatsächlich geschriebenen Byte ist in diesem Fall ungewiss.

Control Bus

Über den Control Bus können zusätzliche Informationen übertragen werden. (Parametrierung, Control- und Statusregister)

Die Signale MainIndex und SubIndex dienen als Adressleitungen. MainIndex dient der groben Selektierung, was gelesen oder geschrieben wird. SubIndex gibt die Daten ganauer an.
Die genaue Definition von MainIndex und SubIndex bleibt der Applikation überlassen.

Die beiden Signale können auch zu einem durchgehenden 24 bit Adressbus verbunden werden.

Das Schreiben wird über das Signal CtrlWriteNow gesteuert. Es ist bei einem Schreibzugriff einen Taktzyklus lang high.
CtrlMainIndex, CtrlSubIndex und CtrlWriteData sind gleichzeitig mit CtrlWriteNow gültig.

Für das Lesen ist kein Strobe Signal notwendig.
Bei Änderungen von CtrlMainIndex oder CtrlSubIndex können die Daten entweder im aktuellen oder im nächsten Taktzyklus bereit gestellt werden.

In der Software kann der Control Bus über zwei Funktionen bedient werden, die in der Demo Software enthalten sind.

Die Funkton CtrlWrite schreibt einen Wert 32 bit Wert über den Control Bus.
Die Werte mainIndex, subIndex geben die gewünschte Speicherstelle an.
Der Parameter writeValue ist der zu schreibende Wert.

Rückgabewert:
Negativ bei Fehler, sonst o.k.

```
int CtrlWrite(usb_dev_handle *handle, int mainIndex, int subIndex,
              int writeValue)
{
      unsigned char transBuf[4];
      for (int i= 0; i<4; i++)
            transBuf[i]= (unsigned char)(writeValue >> (8*i));
      int value= 0xff00+mainIndex;

      return  usb_control_msg(handle, 0x00, 0x7, value, subIndex,
                              (char*)transBuf, 4, 1000);
}
```

Die Funktion CtrlRead liest einen Wert 32 bit Wert über den Control Bus.

Die Werte mainIndex und subIndex geben den gewünschten Wert an.
Der gelesene Wert wird über *retValue zurück gegeben.

Rückgabewert:
Negativ bei Fehler, sonst o.k.

```
int CtrlRead(usb_dev_handle *handle, int mainIndex, int subIndex,
             int *retValue)
{
      unsigned char retBuf[4];
      int value= 0xff00+mainIndex;
      int err=  usb_control_msg(handle, 0x80, 0x6,
                      value, subIndex, (char*)retBuf, 4,
                      1000);
      if (err < 0)
            return err;
      int ret= 0;
      for (int i= 0; i<4; i++)
      {
            ret+= retBuf[i]<<(8*i);
      }
      *retValue= ret;
      return 0;
}
```

Vorgangsweise bei der ersten Inbetriebnahme

Aufgrund der einfachen Struktur des Usb Cores ist die Inbetriebnahme und die Implementierung in eigene Design sehr einfach.

Gehen Sie vor wie folgt:

- Entpacken Sie das Demo Design und den Demo Treiber
- Öffnen Sie das Demo Design mit Xilinx ISE
- Schließen Sie die Hardware an die Versorgung und das Xilinx Kabel an.
- Laden Sie das Design auf die Hardware
  Die grüne LED sollte nun langsam blinken
- Verbinden Sie die Hardware und den PC mit dem USB Kabel
  Die grüne LED sollte nun schnell blinken
- Windows fordert Sie zur Eingabe des Treibers auf. Wählen Sie manuell den Demo Treiber. Dieser Schritt ist nur das erste mal erforderlich.
- Starten sie die Demo Software

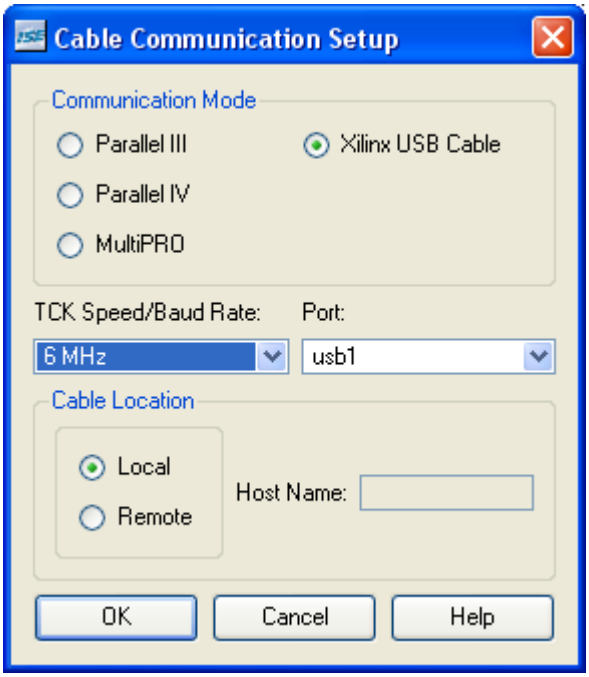
Wichtige Punkte

Geschwindigkeit der Konfiguration

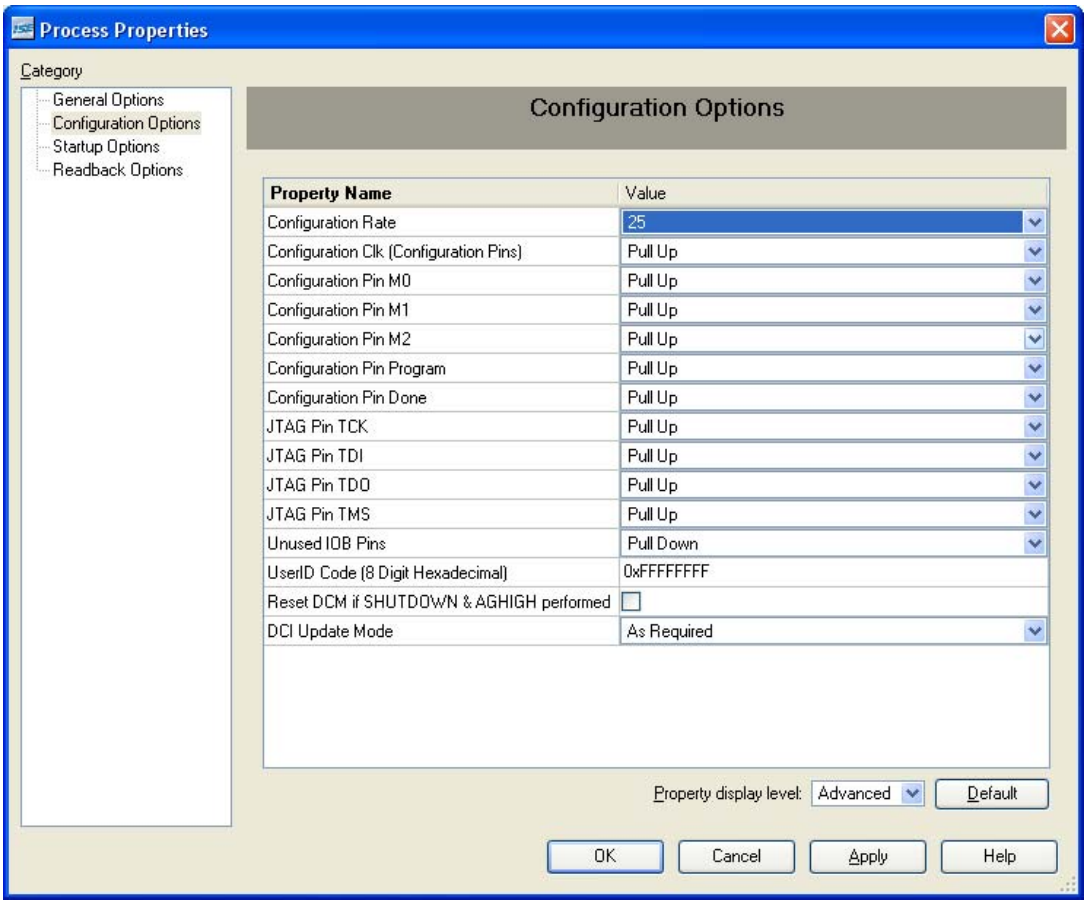
Es ist darauf zu achten, dass der FPGA möglichst schnell konfiguriert wird. Speziell gilt dies, wenn der FPGA über USB versorgt wird.

Wenn der FPGA zu langsam konfiguriert, dann erscheint am PC kurzzeitig eine Fehlermeldung, dass das Gerät nicht erkannt wurde.

Bei der Verwendung des Xilinx USB Kabels sollte der Wert TCK Speed/Baud Rate im Menü „Output, Cable Setup“ mindestens auf 12 MHz gestellt werden.



Wenn aus dem Prom gebootet wird, dann sollte die Configuration Rate mindestens auf 25 gestellt werden.



Xilinx USB Kabel

Wenn das Xilinx USB Kabel verwendet wird, dann sind besondere Vorkehrungen zu treffen.
Während dem Download des Bitfiles darf das konfigurierte Gerät darf nicht am selben PC hängen als das Xilinx Kabel.
Sonst meldet die Xilinx Software Fehler, ein korrekter Download ist nicht möglich.

Wenn nur ein PC zur Verfügung steht, dann muss das Gerät während der Konfiguration kurz abgesteckt werden. Daraus folgt, dass das Gerät in der Entwicklungsphase nicht über USB versorgt werden kann.

LibUsb Versionen

Es ist darauf zu achten, dass alle Komponenten von UsbLib in der selben Version vorliegen. (dll, treiber, library)

Die Versionsnummern werden vom LibUsb Testprogramm ausgegeben, das mit LibUsb automatisch installiert wird.

Probleme bei Map

Es müssen beide Fifos verwendet werden. Wenn ein Fifo offen bleibt, bringt MAP eine Fehlermeldung. Das Problem sollte laut Xilinx ab ISE Version 10 behoben sein.