trenz electronic
Dünner Kirchweg 77
32257 Bünde
Germany
www.trenz-electronic.de

Full-Speed USB 1.1 Function Controller

2001-May-3

Application: SoC with Xilinx Virtex

Introduction

Application of IP-Cores requires in-depth knowledge of the core's behavior. Building up this know-how is greatly simplified by comprehensive reference applications. The following application note was developed to give the engineer a quick hands-on experience and a good starting point for their own developments. To do so, a commercial off-the-shelf FPGA board was chosen as the base platform.

General Overview

This application note describes a System-on-Chip with USB consisting of the following building blocks:

- *USB Function Controller.* The *TE-USB* Core implements the complete USB transaction layer. It is completely hardwired for speed and needs no firmware intervention. The functions of this entity include: frame recognition/ generation, parallel/ serial conversion, bit-stuffing/ de-stuffing, CRC checking/ generation, PID verification/ generation, address recognition and handshake evaluation/ generation.
- *Microcontroller & Firmware.* The *TE-51* Core in conjunction with the firmware implements the USB Device Framework (Chapter 9 Commands). Being compatible to the industry standard MCS 51 family, the TE-51 Core protects your software investment and cuts down development time.
- *Functional Block.* This functional block implements the application's unique device functionality. In this application note these are two simple 7-segment displays with registers which can be written to and read from.

To ease reuse of the USB interface circuit even for inexperienced developers, the USB Function Controller, the microcontroller and firmware are combined in a simple USB Macro.

Architectural Description

XESS XSV-300 Board

The *XSV-300* board is a versatile FPGA development board providing a 300-Kgate FPGA, programmable clock oscillators, standard PC interface components (e.g. a USB-Transceiver), Flash memory, RAM and 7-segment displays. *Figure 1* shows the component side of the board.

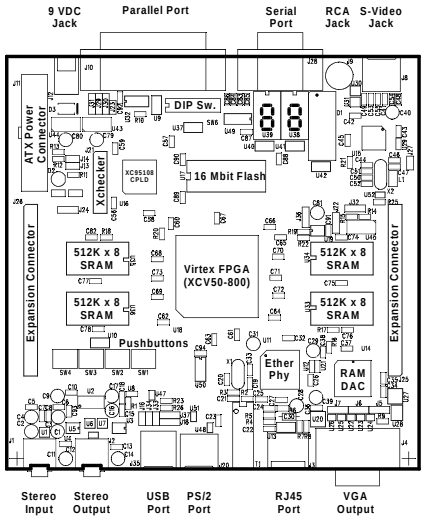

Figure 1: XSV-300 Board

Using proven off-the-shelf hardware which is flexible and feature rich, guarantees short development cycles and allows for rapid prototyping with minimum initial effort.

This design was verified and tested with an XSV-300 board. However, other XSV series boards can be used. The minimum requirement of this application note is the XSV-100.

FPGA Pin Mapping

The entity *xsvPads* is the top level of the design. It splits the buses into their separate signals and assigns IOBs to the signals. Furthermore a global clock buffer is instantiated here to enforce proper buffering of the 48MHz clock. An asynchronous reset signal is derived from a pushbut-

ton using an inverter. Figure 5 shows the block diagram.

FPGA Core Logic

The entity *xsvCore* implements the core logic of the design. It instantiates the USB Macro and the Functional Block. Furthermore it interfaces with the external peripherals. Figure 6 shows the block diagram.

USB Transceiver Interface

The entity *xsvPHY* interfaces the USB Function Controller with the Philips Semiconductors's USB Transceiver PDIUSBP11A. This is achieved by simple combinational logic.

The logic is designed to work with the Transceiver's MODE pin being tied to GND.

USB Macro

The entity *usbMacro* combines the USB Function Controller, the microcontroller plus firmware, and the glue logic into a single entity. This entity was designed to be easily reused in other simple designs. It provides the following features:

- 16-bit output
- 16-bit input
- HID-compatible firmware

This macro is provided free-of-charge under the Gnu Public License as a synthesized netlist. It combines the features of Trenz Electronic's IP Products *TE-51* and *TE-USB* and illustrates their use in a System-on-Chip environment. See Figure 8 for the block diagram.

Entity *usbEP0* implements the USB Function Controller. A 48MHz oscillator drives *clk48. A*n asynchronous, active-high reset on *rst* is required during power-on. The 12MHz clock is generated by the internal digital PLL, therefore *clk12* is driven by *clk12o*. The generics *epin_mask*, *epout_mask*, *epsetup_mask* and *episo_mask* are set up to define Endpoint Zero as the control endpoint, and Endpoint One as an IN-only Endpoint without the isochronous transfer capability. Refer to Trenz Electronic's *TE-USB* Product Specification for further details.

Entity *te51c* implements the MCS 51-compatible microcontroller. As this is a SoC design, Port 0 multiplexing was omitted, resulting in a slightly modified footprint for this entity. Refer to Trenz Electronic's *TE-51* Product Specification for further details.

Entity *xsvROM* implements the HID-compatible device firmware. This entity is implemented using a CoreGenerator module for the highest possible density.

The entity *xsvGlue* implements glue logic to connect the USB Function Controller with the microcontroller. Datapathes are included to access the FIFO, the Control/Status Word and the Device Address. All items are memory-mapped for easy implementation. In addition to this, the endpoint control logic for the 16-bit I/Os is implemented here. This is basically two simple state machines plus output registers. 16-bit I/Os were chosen to provide increased flexibility for simple devices. In case more than 16 I/Os are required, some of the output signals may be used for addressing purposes.

The entity *clkdiv* implements a simple clock divider, creating a 24MHz clock used by the microcontroller.

Functional Block

The entity *xsvFunc* implements the functional block of this application note which is self-explanatory: As Figure 7 details, it just routes the 16-bit output to the two 7-segment displays and feeds the value back to the input. Feel free to add more to it...

Synthesis and Implementation

Synthesis and Implementation of the design was done using Xilinx Foundation Express Software, version F3.1i.

To synthesize the design, create a project containing the following files:

- *xsvFUNC.vhd* (entity *xsvFunc*)
- *xsvCORE.vhd* (entity *xsvCore*)
- *xsvPHY.vhd* (entity *xsvPhy*)
- *xsvPADS.vhd* (entity *xsvPads*)

Set the project's top level to entity *xsvPads*. Because entity *usbMacro* is provided as an EDIF netlist, it remains unexpanded during synthesis. Most synthesis tools will issue a warning, which can be safely ignored.

Next, follow these steps to implement the design:

- Add file *usbMacro.edf* implementing entity *usbMacro* to the flow. To do so, the Xilinx *ngdbuild* requires *usbMacro.edf* to reside in the same directory, as the synthesis output.
- Constrain timing according to Table 2. By doing so, Place&Route is advised of fast datapathes and ensures proper timing. Even though the signals *clk12* and *clk24* are buried inside usbMacro, proper setup of timing constraints is vital for error-free operation.
- Constrain pins according to Table 3. By doing so, Place&Route is advised to assign pin locations according to the *XSV-300* board's hardware. Please note, that the pin *rcv_p9* is optimized away during synthesis. For this reason, no constraint is given here. However, a constraint must be added, whenever the design uses *rcv_p9* due to future modifications.

The file *usb-xsv.ucf* contains all required constraints for this design. You should review the Pad Report and Timing Report after implementation to verify that all constraints are met.

Table 1 summarizes the resource usage of the design implemented in a Xilinx XCV300 device. Please note, that these figures may vary slightly with other tools or future versions.

Resource	Usage
Number of Slices	718/ 23%
Number of Slice Flip Flops	489/ 7%
Total Number 4 input LUTs	1,207/ 19%
Number used as LUTs	1,135
Number used for 32x1 RAMs	64
Number used as 16x1 RAMs	8
Number of bonded IOBs	21/ 12%
Number of Tbufs	40/ 1%
Number of Block RAMs	2/ 12%
Number of GCLKs	3
Number of GCLKIOBs	1
Total equivalent gate count for design	53,222

Table 1: XCV300 Resource Usage

signal	constraint
clk_p89	PERIOD 48MHz, HIGH 50%
Ucore/Uusb/CLK12	PERIOD 12MHz, HIGH 50%
Ucore/Uusb/CLK24	PERIOD 24MHz, HIGH 50%

Table 2: Timing Constraints

pin	location
clk_p89	P89
sw1_p174	P174
rcv_p11	unused (P11)
vp_p10	P10
vm_p9	P9
vpo_p13	P13
fse0_p17	P17
oe_p12	P12
suspnd_P176	P176
sl0_p177	P177
sl1_p167	P167
sl2_p163	P163
sl3_p156	P156
sl4_p145	P145
sl5_p138	P138
sl6_p134	P134
sr0_p124	P124
sr1_p132	P132
sr2_p133	P133
sr3_p139	P139
sr4_p141	P141
sr5_p144	P144
sr6_p147	P147

Table 3: Pin LOC Constraints

Setup of XSV Board

CPLD Interface

This application drives both 7-segment displays from the FPGA. As the FPGA and CPLD are both connected to the 7-segment displays, the CPLD should tristate all its lines going to the displays in order to avoid contention. This is best achieved by loading the *usbdwnldpar.svf* interface into the CPLD. *Figure 2* shows the circuit diagram.

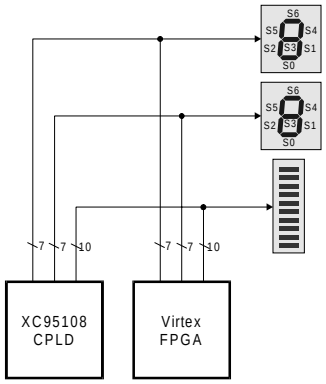

Figure 2: 7-Segment displays.

Note that you may cause permanent damage to the *XSV-300* board when driving the 7-segment displays from both the CPLD and the FPGA simultaneously.

Oscillator

The *TE-USB* Core requires a clock frequency of 48MHz which must be generated by using an external clock oscillator. To do so, the clock signal is fed into *J27* with *J36* being jumpered to *2-3*. See *Figure 3* for further details.

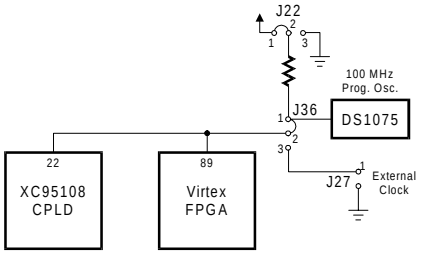

Figure 3: Clock oscillator.

Do not use a 50 MHz clock generated by the onboard programmable oscillator as this is not within the tolerance required for proper operation of the TE-USB core.

USB Connector

The *TE-USB* is a full-speed device. This is reflected by setting both *J18* and *J37* to *2-3*. *Figure 4* shows the circuit diagram.

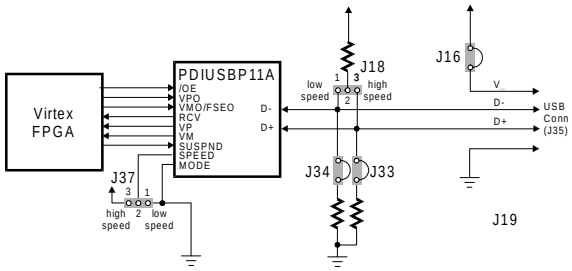

Figure 4: USB port.

The *XSV-300* provides a Type-A USB Receptable. This type of connector is typically used as the Downstream Output of a USB Host Controller or Hub. As this application note implements a USB Device, an A-A cable is required (which according to USB 1.1 Spec is a prohibited Cable Assembly).

USB Enumeration

When connecting the board to a USB host controller, the host will enumerate the design with the following device descriptor:

```
bcdUSB:             0x0110
bDeviceClass:         0x00
bDeviceSubClass:      0x00
bDeviceProtocol:      0x00
bMaxPacketSize0:      0x08 (8)
idVendor:           0x0BD0
idProduct:          0x0100
bcdDevice:          0x0100
iManufacturer:        0x01
0x0409: "Trenz Electronic"
iProduct:             0x02
0x0409: "TE-USB"
iSerialNumber:        0x00
bNumConfigurations:   0x01
```

This declares the device to be compatible with the *Device Class Definition for Human Interface Devices (HID)* specified by the USB Implementers Forum. Following the HID specification ensures maximum compatibility with standard operating systems, and avoids the need for custom driver development.

Host software

To allow easy verification of the USB functionality, a simple host software running on Windows is provided. After starting the software, you will be asked for the Vendor and Product ID, which should be entered in hexadecimal format:

```
VID: 0x0bd0
PID: 0x0100
```

After entering a *w*, you can write data to the device:

```
r, w, x: w
1: 0x55
2: 0xaa
hidWrite: 55 AA
```

The 7-Segment displays will light up according to the data being sent.

After entering an *r*, the software reads data from the device:

```
r, w, x: r
hidRead: 55 AA
```

The data being read, will reflect the current status of the 7-Segment displays.

After entering an *x*, the software quits.

The host software is provided in source form “as-is” and is meant to serve as a good starting point for your own developments. Refer to the *References* Section for further readings.

References

Literature

- Full-Speed USB 1.1 Function Controller
  Product Specification
  Trenz Electronic
  http://www.trenz-electronic.de
- MCS 51 Compatible
  8-bit Microcontroller Core
  Product Specification
  Trenz Electronic
  http://www.trenz-electronic.de
- USB Design by Example
  John Hyde
  Intel Press
  http://www.usb-by-example.com
- Universal Serial Bus
  Specification
  USB Implementers Forum
  http://www.usb.org

Hardware

- XESS Corporation
  2608 Sweetgum Drive
  Apex NC 27502
  http://www.xess.com
- Xilinx Inc.
  2100 Logic Drive
  San Jose, CA 95124
  http://www.xilinx.com

Web Resources

- xsboard-users newsgroup
  http://groups.yahoo.com/group/xsboard-users
- Jan Axelson's Lakeview Research
  http://www.lvr.com
- USB-IF Developers WebBoard
  http://www.usb.org/forums/developers/webboard.html
- FPGA newsgroup
  comp.arch.fpga
- VHDL newsgroup
  comp.lang.vhdl

Revisions History

Version	Date	Who	Description
1.0	01mar19	FB	Initial version
1.1	01may03	FB	XSV-300 version

Table 4: Revisions History

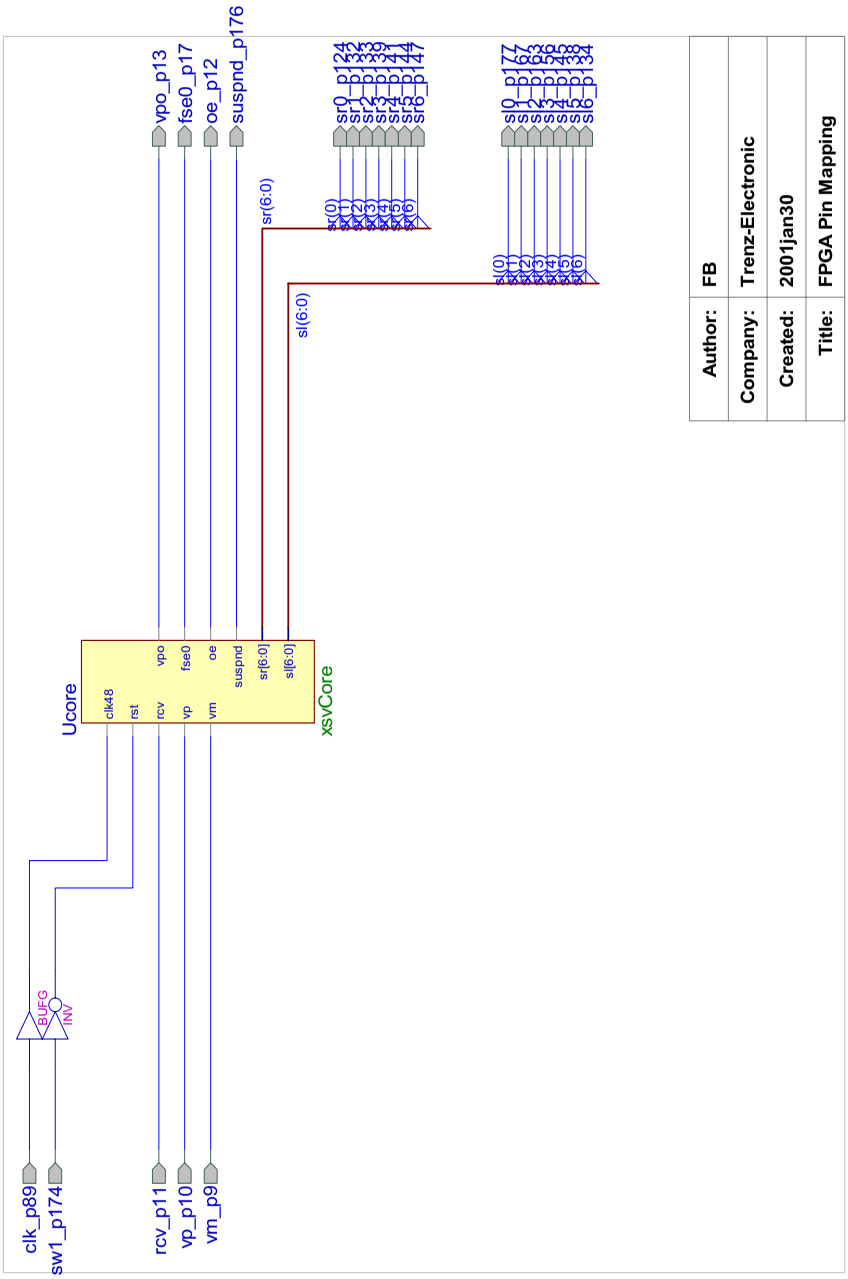

Figure 5: FPGA Pin Mapping

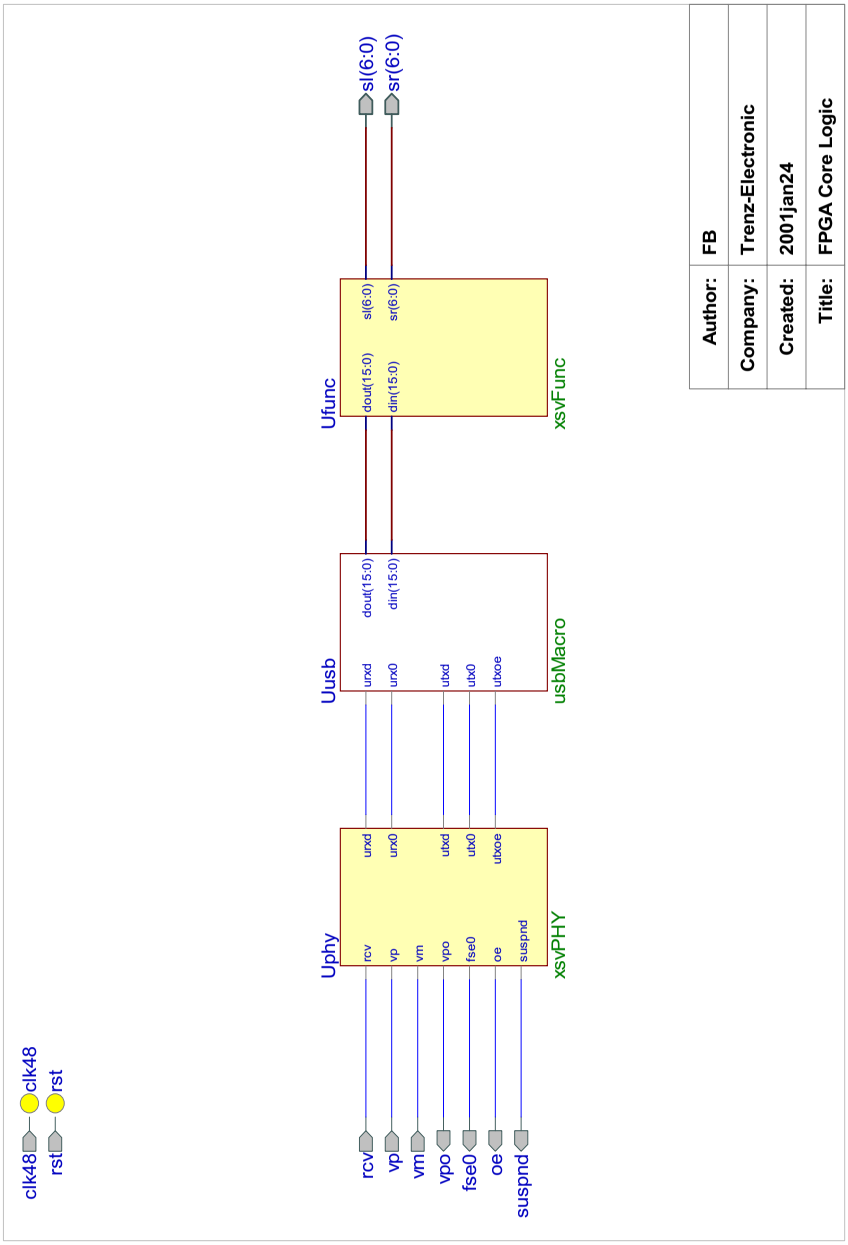

Figure 6: FPGA Core Logic

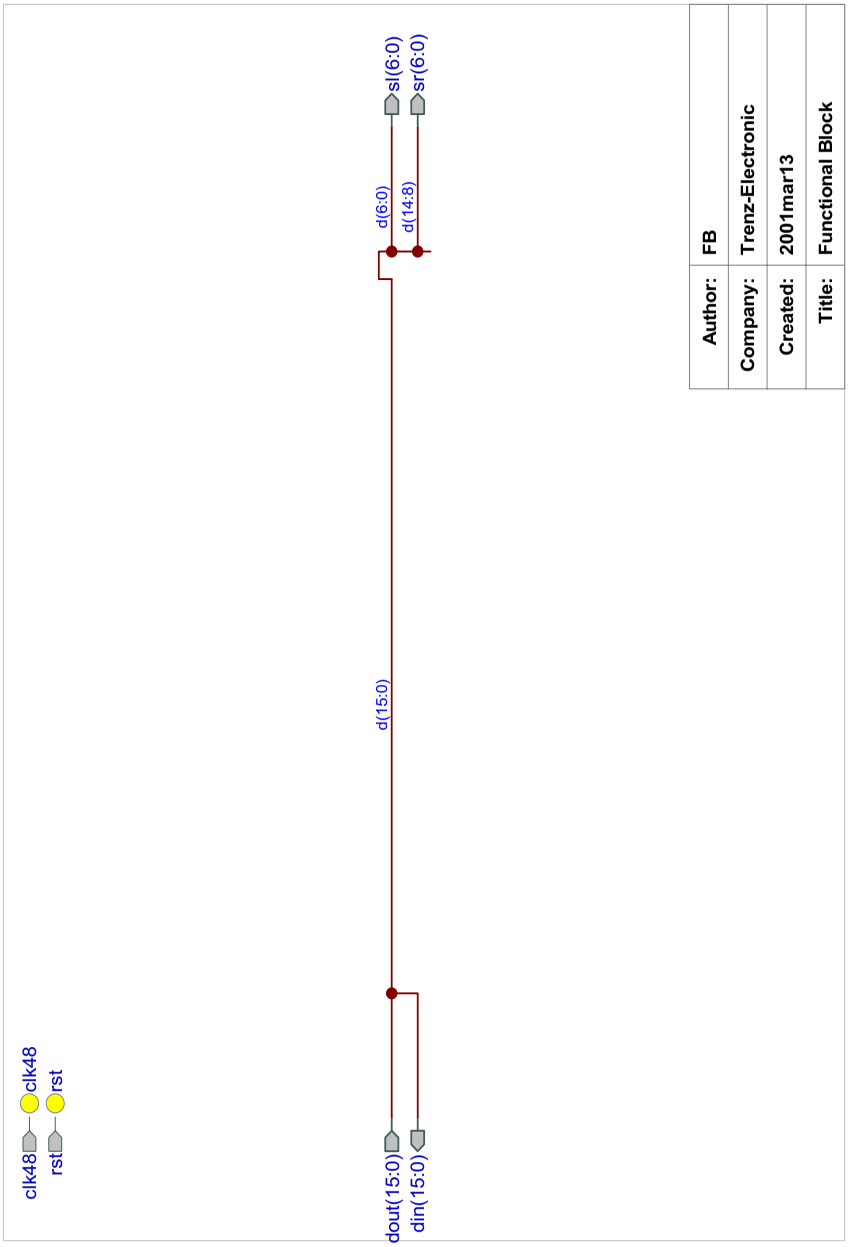

Figure 7: Functional Block

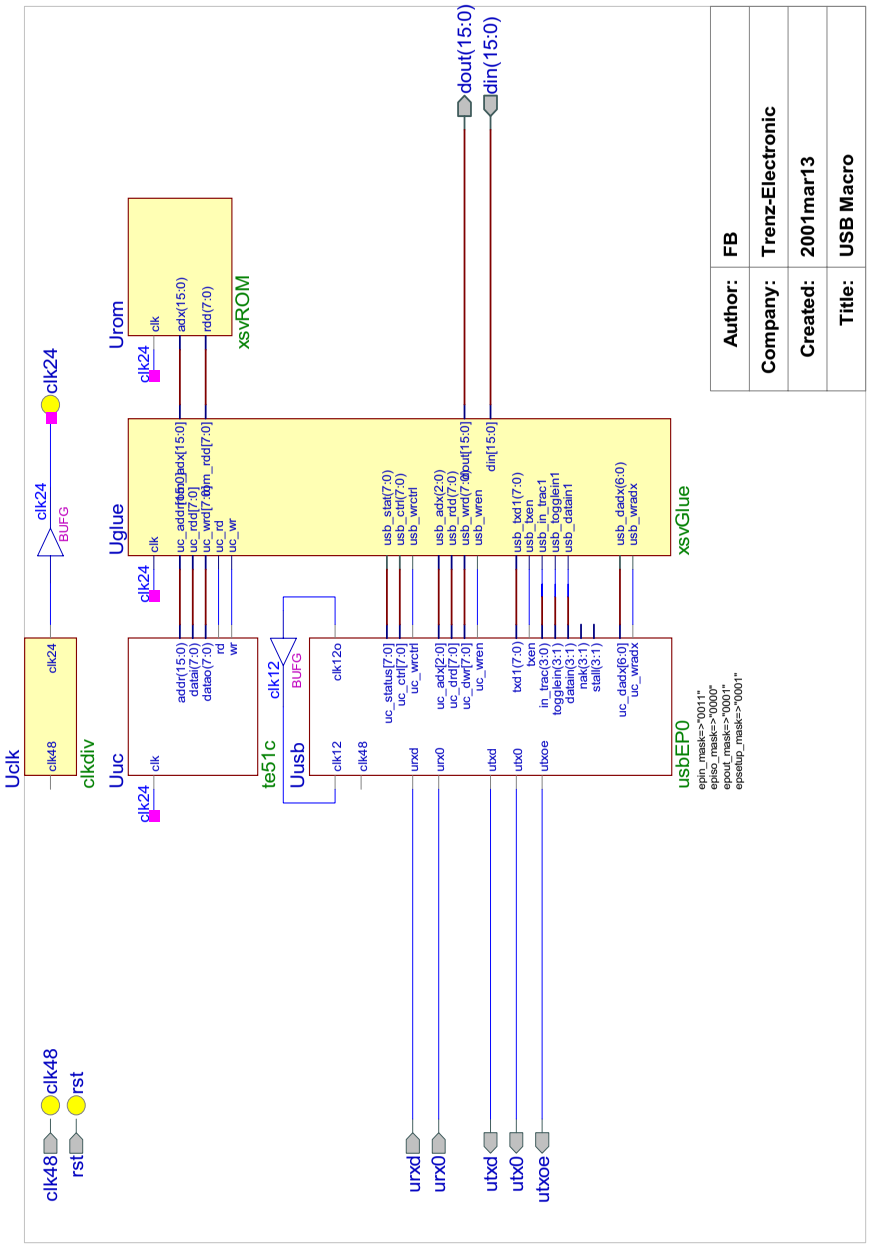

Figure 8: USB Macro