**Overview**

**USB2-SOFT** is the VHDL source code to connect an FPGA-based USB device to a USB host. The code supports both high-speed (480 Mbits/s) and full-speed (12 Mbits/s) connections.

This FPGA code interfaces with a physical layer transceiver (PHY). The code is compatible with two industry standard interfaces commonly found in USB PHYs:

- 12-pin ULPI [2] (example: USB3300 from SMSC [5])
- 22-pin UTMI

The code is structured so that the ULPI interface is a wrapper around the UTMI SIE.

The code is a Serial Interface Engine (SIE) implementing the Device side of the USB 2.0 standard protocol layer [1] including control and bulk endpoints.

**USB2-SOFT** does NOT support the following: Host-side protocol, OTG, low-speed, Interrupt and Isochronous endpoints.

The component’s very efficient implementation makes it suitable for instantiation within a small FPGA. For example, it takes 22% of a small Spartan-6 XC6SLX16 [3][4].

**Block Diagram**

(*) The code is written for two bi-directional virtual channels (4 endpoints). Developers can easily change the number of channels/endpoints by editing the `USB20.vhd` source code.

**Target Hardware**

The code is written in generic VHDL so that it can be ported to a variety of FPGAs. The code was developed and tested on a Xilinx Spartan-6 XC6SLX FPGA.

It can be easily ported to any Xilinx Virtex-5, Virtex-6, Spartan-6, Spartan-3 FPGAs and other FPGAs capable of running at 60 MHz.

**Device Utilization Summary**

<table>
<thead>
<tr>
<th>Device: Xilinx Spartan-6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of slices</td>
</tr>
<tr>
<td>Flip Flops</td>
</tr>
<tr>
<td>LUTs</td>
</tr>
<tr>
<td>RAMB16BWERs</td>
</tr>
<tr>
<td>DSP48A1s</td>
</tr>
<tr>
<td>GCLKs</td>
</tr>
<tr>
<td>DCMs</td>
</tr>
</tbody>
</table>
**User Interface**

The user interface is synchronous with the user-supplied processing clock CLK_P. In order to meet the timing requirements, CLK_P must be defined as a global clock (i.e. a BUFG output).

CLK_P does not have to be the same as, or be related to the 60 MHz PHY clock.

Each data stream, whether input or output, follows the same timing diagram as illustrated below:

![Timing Diagram](image)

Each DATA byte is read at the rising edge of CLK_P when SAMPLE_CLK = ‘1’. In a sense, SAMPLE_CLK is really an ‘enable’ or ‘valid’ qualifying signal.

The stream data flow is controlled by means of the SAMPLE_CLK_REQ signal. ‘0’ means that the receiving end does not have room for any more data. The data source should not send data unless SAMPLE_CLK_REQ = ‘1’.

For maximum throughput, 16Kbit elastic buffers are included in both tx/rx directions at the user interface. This allows the USB engine to multi-task, sending data to the PHY while accepting subsequent data from the user and vice versa.

The USB protocol is invisible to the user. The user has no control, nor visibility of the data segmentation into DATA0/DATA1 packets, frame check sequence (CRC) insertion and removal.

**PHY Interface**

The PHY interface is synchronous with the PHY-supplied 60 MHz reference clock USB_CLK60.

Recommendation for best performance:

(a) At the time of PCB layout, connect the PHY 60 MHz reference clock to a FPGA global clock GCLK input port.

(b) Inside the FPGA, re-generate the 60 MHz reference clock through a DCM or PLL followed by a global buffer BUFG. An example is available in [6].

The most critical timing in this design is the time it takes for the USB_ULPI_DIR and USB_ULPI_NXT input signals to propagate through the FPGA and generate the appropriate USB_ULPI_DATA output by the next clock. It is thus important to define FPGA timing constraints as follows:

```
INST "USB_ULPI_NXT" TNM = USB_ULPI_IN;
INST "USB_ULPI_DIR" TNM = USB_ULPI_IN;
TIMEGRP "USB_ULPI_IN" OFFSET = IN 13.666 ns BEFORE USB_CLK60;  #3ns at ULPI PHY output, period 16.6ns
```

**Configuration**

There are no run-time configuration parameters. The two most likely customizations a developer may be tasked to implement are:

(a) adding/removing endpoints, and

(b) changing the descriptors.
Adding/Removing Endpoints
It is quite easy to add or remove USB endpoints (for example to create an application with more transmit streams than receive streams). To do so, one must cut/paste/edit sections of the USB20.vhd component.

To create an additional bulk-out endpoint, cut and paste the code between
--// ENDPOINT 2 TRANSFER STATE MACHINE
and
---//END OF ENDPOINT 2 TRANSFER STATE MACHINE
then rename the interface from IF0 to IF?.

Likewise, to create an additional bulk-in endpoint, cut and paste the code between
--// ENDPOINT 3 TRANSFER STATE MACHINE
and
---//END OF ENDPOINT 3 TRANSFER STATE MACHINE
then rename the interface from IF0 to IF?.

Changing Descriptors
USB devices report their attributes using descriptors. Descriptor strings include manufacturer’s name, product’s name, number of interfaces and endpoints, etc.

Descriptors are stored in read-only block RAM according to the following memory map:
0x00 - 0x11 Device descriptor
0x12 - 0x1B Device_Qualifier descriptor
0x1C - 0x52 Configuration descriptor (includes interface and endpoints)
0x53 - 0x89 Other-speed configuration descriptor (offset by x37 from configuration descriptor)
0xA0 - 0xA2 String0 descriptor
0xA3 - 0xBC String1 descriptor
0xBD - 0xCE String2 descriptor
0xE5 - 0xFA String4 descriptor

Descriptors are formatted as specified in the USB 2.0 specifications, section 9.5.

Exclusions
USB2-SOFT does NOT support the following:
- Host-side protocol
- OTG
- low-speed (1 Mbits/s)
- Isochronous endpoints
- Interrupt endpoints

Software Licensing
USB2-SOFT is supplied under the following key licensing terms:

1. A nonexclusive, nontransferable license to use the VHDL source code internally, and
2. An unlimited, royalty-free, nonexclusive transferable license to make and use products incorporating the licensed materials, solely in bitstream format, on a worldwide basis.

The complete VHDL/IP Software License Agreement can be downloaded from http://www.comblock.com/download/softwarelicense.pdf

Reference documents
[2] UTMI+ Low Pin Interface (ULPI) Specification Revision 1.1 October 20, 2004
[4] COM-1600 development platform schematics
[5] SMSC USB3300 Hi-Speed USB Host,Device or OTG PHY with ULPI Low Pin Interface
[6] COM-1600 VHDL code template
Configuration Management
The current software revision \( rev \) is 2. The software comprises:

[a] VHDL source code in directory
   USB20_{rev}/src

[b] Xilinx .ucf constraint statements (to be copied to
   the project top level .ucf constraint file)
   USB20_{rev}/src\USB20ULPI.ucf

[c] synthesized .ngc component:
   USB20_{rev}/bin\USB20ULPI.ngc

where \( rev \) is the current revision number.

VHDL development environment
The VHDL software was developed using the following development environment:

(a) Xilinx ISE 13.4 with XST as synthesis tool
and ISim simulator.

(b) COM-1600 Spartan-6 FPGA development
   platform, including the USB3300 PHY

Ready-to-use Hardware
The binary component (.ngc) is freely available for use on the following Comblock hardware modules:

- COM-1600 FPGA + ARM + DDR2 + NAND + USB2 development platform
- COM-1500 FPGA + ARM + DDR2 SODIMM development platform

The schematics are available in this CD.

Xilinx-specific code
The VHDL source code is written in generic VHDL with few Xilinx primitives. No Xilinx CORE is used. The Xilinx primitives are:

- IOBUF
- IBUFG
- BUFG (global clocks)
- RAM block: RAMB16_S9_S9

Top-Level VHDL hierarchy
The code is stored with one, and only one, component per file.

The root entity (highlighted above) is USB20ULPI.vhd. It is a wrapper that converts the natural PHY interface from UTMI to the lower-pin count ULPI. This ULPI wrapper can be removed if the FPGA interfaces with the external USB PHY over a UTMI interface.

The root also includes the following components:

- USB20.vhd is the Serial Interface Engine for a USB device. It can interface directly with a PHY over a UTMI interface.
- The CRC5.vhd verifies the CRC for incoming USB tokens.
- The CRC16.vhd computes the 16-bit CRC to be appended to tx packets and to check rx. The CRC computation is performed 8 data bits at a time.

Test Environment
A testbench (tbusb20ulpi.vhd) together with a simple PHY simulator (USB_PHY_SIM.vhd) are included. This allows to simulate the SIE and ULPI interface during high-speed negotiation and token exchange. All ULPI transactions are simulated: receive command, transmit command, data transmit, data receive, register read and register write.

To shorten the simulation, the constant SIMULATION in usb20.vhd should be set to ‘1’.
**Clock / Timing**

The software uses two main clocks:

- a 60 MHz reference clock (USB_CLK60G) generated by the PHY and ‘cleaned’ through a DCM or PLL.

- A user-selected processing clock (CLK_P) to send and receive data from the user application.
State Machine

The state machine is described by the SDL flowcharts below:

Initial conditions at power up:
- XCVR_SELECT = 0
- TERM_SELECT = 0
- OPMODE = 00 (normal)
- TXVALID = 0
- DATA_OUT = Zs
- SPEED_BEFORE_SUS = '0'

Device detached

Initial conditions at power up:
- XCVR_SELECT = 0
- TERM_SELECT = 0
- OPMODE = 00 (normal)
- TXVALID = 0
- DATA_OUT = Zs
- SPEED_BEFORE_SUS = '0'

Device attached

1 CLK wide pulse

VBUS_SENSE = 1?

XCVR_SELECT <= '1'
TERM_SELECT <= '1'
go to FS

Device detached

Initial conditions at power up:
- XCVR_SELECT = 0
- TERM_SELECT = 0
- OPMODE = 00 (normal)
- TXVALID = 0
- DATA_OUT = Zs
- SPEED_BEFORE_SUS = '0'

Device attached

1 CLK wide pulse

VBUS_SENSE = 1?

XCVR_SELECT <= '1'
TERM_SELECT <= '1'
go to FS

Debounce
Debounce

T1

FS_ACTIVE
Wait for reset or suspend from upstream

SPEED_BEFORE_SUS = '1'

LINESTATE = J for >3ms

LINESTATE = SE0 for > 2.5us

RESET

1 CLK wide pulse

Reset device & PHY

OPMODE <= "10"
TXVALID <= '1'
DATA <= (others => '0')
XCRVSELECT <= '0'
disable bit stuffing
go to HS mode
send K (chirp)

1ms min + margin
66000 clock cycles

Send Chirp K
OPMODE <= "00"  TXVALID <= '0'
still in HS mode stop transmitting K chirp

T1 0.1 ms
Wait until we can assert Chirp K from upstream

8
Wait for port chirp

T1

T1 2 ms

9
check for chirp

T1
revert to FS

XCRVSELECT <= '1'

T1 100 ms

10

CHIRP_STATE = 3

TERMSELECT <= '0'

4
HS_active

USB_SOF

T1

3
FS_active
SPEED_STATE = 9 < check chirp
chirp_count = 0

chirp_count = chirp_count + 1
Chirp_count = 6

SE0 for 2.5 us
SPEED_STATE /= 9

LINESTATE = K
chirp_count = chirp_count + 1
Chirp_count = 6
Yes

LINESTATE = J
chirp_count = chirp_count + 1
Chirp_count = 6
Yes

SPEED_STATE /= 9
SUSPEND (already in FS mode)

LINESTATE = SE0 or K

T1 = 5.6 ms

resume check

T1

LINESTATE = SE0

Yes

Reset device & PHY

OPMODE <= "10"
TXVALID <= '1'
DATA <= (others => '0')
XCRVSELECT <= '0'

T1 1.1 ms

Suspend

LINESTATE = SE0

Yes

Suspend

LINESTATE = K

No

T1 = 100 ms

LINESTATE = SE0

No

T1

LINESTATE = SE0

 awaited

SPEED_BEFORE_SUS

'0'

XCRVSELECT <= '0'
TERMSELECT <= '0'

HS_active

Yes

No

T1

Suspend

Send chirp K

 wartime

FS_active
4  HS\_active

SPEED\_BEFORE\_SUS = '0'

SE0 for > 3 ms

XCRVSELECT <= '1'
TERMSELECT <= '1'

go to FS mode

T1 0.5 ms

6  reset\_suspend

T1

5  Suspend

Reset device & PHY

OPMODE <= "10"
TXVALID <= '1'
DATA <= (others => '0')
XCRVSELECT <= '0'

T1 1.1 ms

7  Send chirp K
Valid SETUP
- PID valid
- CRC5 valid

Await DATA0

CRC16 valid?

Data1, Data0
- toggle datax flag on rx
- accept data

Send ACK

await ACK transmit

Send ACK

setup complete
data x flag = 0
Valid OUT

Valid CRC16

sequence matches flag?

Valid OUT

space in pipe?

toggle datax flag
accept data

Valid OUT

Valid OUT

Send ACK

Send NAK

NYET sent

Send NYET

next space in pipe

Valid OUT

Valid OUT

Valid OUT

Wait ACK transmit

Wait NAK transmit

Wait NYET transmit
State Previously in

Valid PING

PID valid

CRC5 valid

Space in the pipe?

Yes

Send ACK

5

ACK sent

No

Send NAK

6

NAK sent
Valid IN

if data available to send

Valid ACK
toggle data x flag
accept data

Send NAK

Send Data x
await data x transmit

No data to transmit

Data x tx done

Valid IN

data x flag = 0

0

2

NACK DONE

1

3

data transmitted
wait for ack

No
**ComBlock Compatibility List**

<table>
<thead>
<tr>
<th>FPGA development platform</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>COM-1600</strong> FPGA + ARM + DDR2 + USB2 + NAND development platform</td>
<td></td>
</tr>
<tr>
<td><strong>COM-1500</strong> FPGA + DDR2 SODIMM socket + ARM development platform</td>
<td></td>
</tr>
</tbody>
</table>

**ComBlock Ordering Information**

USB2-SOFT  USB2.0 DEVICE SIE, VHDL SOURCE CODE

MSS • 845-N Quince Orchard Boulevard•
Gaithersburg, Maryland 20878-1676 • U.S.A.
Telephone: (240) 631-1111
Facsimile: (240) 631-1676
E-mail: sales@comblock.com