USB FS (Full Speed) Test Bench - Documentation

Martin Neumann 15th of April 2013

An USB FS Host simulation environment (test bench) in VHDL

This USB FS test bench has been used with the Model Sim VHDL Simulator, however any other ‘event driven’ VHDL simulator should work as well.

This test bench contains a ‘Command Engine’ that supports all ’low level’ USB FS commands as

* Out Token Command

* In Token Command

* SOF Token Command

* Setup Token Command

* Data0 Command

* Data1 Command

* ACK Handshake Command

* NAK Handshake Command

* STALL Handshake Command

* USB Reset

Since all USB HS devices must be downward compatible, this FS simulation environment is also useable for USB 2.0 designs. A true USB 2.0 implenentation needs some more work - few USB 2.0 commands as Data2, MData, NYET and PING are already implemented, however the CHIRP logic is missing and a complete new clock logic will be required.

All commands are implemented as procedure calls, this procedures add the synchronization preamble, PIP, its complement, correct bit-stuffing and CRC-5 respective CRC-16 bits in all this cases.

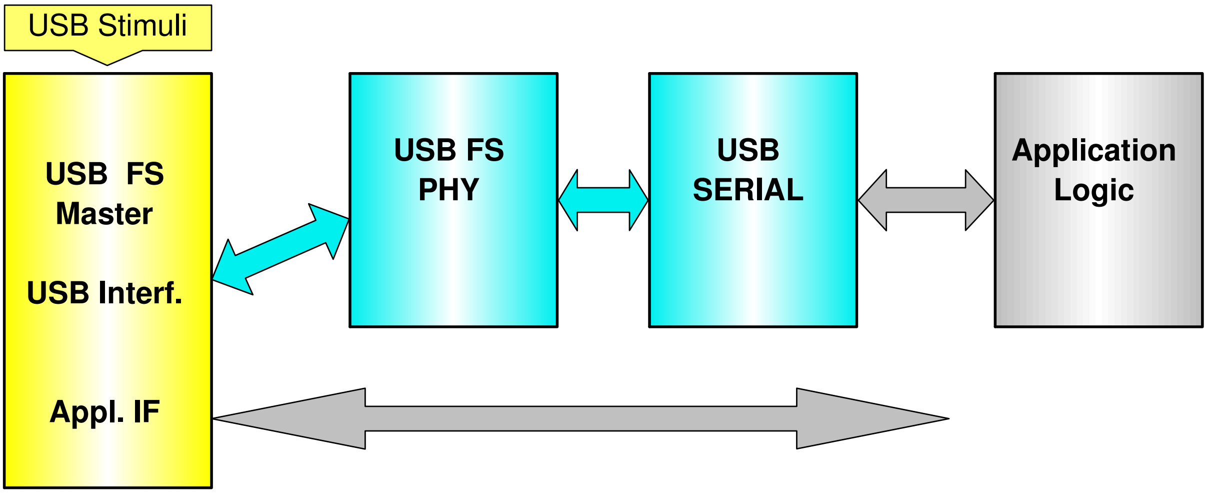
An independent USB Monitor monitors all bus activities and logs the result on the screen and in a Result.out file.

This monitor detects all USB FS Token, Data and Handshake commands It also adds direction information to distinguish if the commands are initialized from the USB host or the USB device under test.

USB FS (Full Speed) Test Bench - Documentation

Martin Neumann 15th of April 2013

When developing an USB application device, a test scenario should be able to simulate the USB host and of course the private developed application as well. The concept of the USB test environment is shown in the following figure:



The above block diagram symbolizes a test bench usb_tb.vhd consisting out of the following parts:

USB FS Master

• usb_commands.vhd	- the USB command package
• usb_stimuli.vhd	- a renamed test case file
• usb_fs_monitor.vhd	- logs all usb activities in files Transcript and Result.out
• usb_fs_master.vhd	- the usb master top entity

USB FS PHY

Open Cores USB Phy, designed by Rudolf Usselmanns according to the USB 2.0 UTMI interface specification.

This Verilog design has been translated to VHDL. From the RX portion are two versions derived, the original version that operates with a 48 MHz clock and a modified version for a 60 MHz clock.

- usb_rx_phy_60MHz.vhdl
- usb_tx_phy.vhdl
- usb_phy.vhdl

USB Serial

USB Serial, designed by Joris van Rantwijk, an USB to RS232 converter (available at http://www.xs4all.nl/~rjoris/fpga/usb.html). This VHDL design operates either at FS or HS, here tied to FS.

- usb_pkg.vhdl
- usb_init.vhdl
- usb_control.vhdl
- usb_transact.vhdl
- usb_packet.vhdl
- usb_serial.vhdl

Others

The top entity **usb_tb.vhdl** contains the USB simulation master **usb_fs_master**, the USB serial port **usb_fs_port** and a sample application (procdess **simple_application**). The simple_application process swapes high and low order nibbles of all received bytes and stores them in the ransmit buffer of the usb_fs_port entity. The usb_fs_port.vhdl file wraps the USB slave top entities (**usb_phy.vhdl** and **usb_serial.vhdl**), the usb_fs_master.vhdl file contains the actual test case file usb_stimuli.vhd, the usb_fs_monitor.vhd and some processes, that translate the procedure calls of the test case file into usb line activities.

The usb_fs_master entity provides only three signals - a negative active reset and the USB Dn and Dp signals - that's it. The top entity ties this USB signals together with those from the usb_fs_port and the usb_fs_monitor. According to the privious picture the application interface (Appl IF) consists here just out of the reset signal, however this can be expanded if more application specific control signals are required.

In the following we concentrate on the yellow portion of the above figure. To have a stable test bench environment even with various test cases, all test case files must have an identical Entity structure. For this reason it is recommended to copy and rename the actual test case to usb_stimuli.vhd prior to a new simulation run. In a Windows environment this could be simplified with the TC_Copy.bat (after adopting it's directory structure to your environment). Place a link to this file on the desktop and then just drop a new tc_xx.vhd on this link - and voilà, a new usb_stimuli.vhd is created.

When looking at a sample usb_tc file, we see that the USB access is done as procedure calls from within a normal process without sensitivity list. This allows us to insert WAIT statements at any time, we may insert LOOP constructs and any other legal sequential statements.

Before we go into more details, all usb and the two list commands log their output at two places :

- First in the Model Sim specific Transcript file (with other simulators - wherever they place their comments and warnings) and
- Second in a simulator independent file named Result.out, created in the simulator home directory (I force this to be the top level of an actual test structure with the Model Sim 'change directory' command).

After a successful simulation run it is a good idee to rename the test log file Result.out so that its file name matches that of the original test case file, e.g. **usb_tc03.out** for later reference.

Now to the first procedure calls provided:

- **list(T_No, positive);**
- **list("String");**

List() procedure calls are just for better orientation within a larger test case, the first statement loads an otherwise not used test case signal **T_No** to the specified value and logs this in the Transcript and Result.out file , the

list (”Reset completed”);

procedure writes just a comment, in this case „Reset completed”, at this two places.

The next following procedure calls carry all usb as first parameter. This out-parameter 'usb' is only internally used and controls the proper timing sequence. The USB Token Commands OUT, IN, SOF and SETUP are realized with the following very similar are procedure calls

- **out_token(usb, device_addr, endp_addr);**
- **in_token(usb, device_addr, endp_addr);**
- **sof_token(usb, frame_no);**
- **setup(usb, device_addr, endp_addr);**

To use the handy hexadecimal notation, the device_addr must be specified with 8 instead 7 bits, and similarly frame_no with 12 instead 11 bits. This MSB bit is dropped in all this cases.

```
setup(usb, X"00", X"0");
```

is then a setup command , in this case to usb-device address 0, endpoint 0;
The procedures adds internally the synchronization preamble, PIP, its complement, correct bit-stuffing and CRC-5 in all this cases.

The data procedure calls

- **send_D0(usb, wr_data);**
- **send_D1(usb, wr_data);**

require as wr_data parameter a byte array in the following notation :

```
send_D0(usb,(X"80",X"06",X"00",X"01",X"00",X"00",X"12",X"00"));
```

After transfer of the last byte the procedure will add automatically the internally computed CRC-16 value. The length of the array is freely selectable (also no data is allowed), however in case of a single byte the notation gets clumsy, we have to specyfy wr_data with index 0 as:

```
Send_D0(usb,wr_data(0) => X"12");
```

or aternative as

```
Send_D0(usb,(0 => X"12"));
```

The handshake procedures are very simple and self explaining:

- **send_ACK(usb);**
- **send_NAK(usb);**
- **send_STALL(usb);**

We covered now the Token, Data and Handshake Commands. A specialty is the USB reset - the USB reset condition is met if the USB Master pulls down both data lines to low levels (SE0) for at least 10 ms, the USB Slave may recognize the reset condition already after 2.5 µs. To keep simulation time to a minimum, the

- **send_RES(usb);**

command forces the SE0 condition for only 5 µs, both design units, the USB PHY and USB Serial will detect the reset condition within this time frame.

Whenever we expect a response from the USB slave we must issue

- **wait_slv(usb);**

This command waits for any slave response, is it either a handshake command or a stream of data.

The details of this as of all other transfers are logged by the USB_monitor entity - this entity monitors all bus activities and writes the result into the Transcript (Model Sim specific) and in the Result.out files. The monitor detects all Token, Data and Handshake commands In case the commands are initialized from the USB Master, the commands are preceded by **'Send'**, otherwise by **'Recv'**.
Data transfer results will start a new line every 16 bytes. The last two bytes are the CRC-16 bytes witch may be ignored.

The following listing shows a typical Windows 7 configuration sequence and some communication to our 'simple_application' process (all sof_tokens exept for the very first three are stripped) and the corresponding Result.out file.

```
--==========================================================================--
--                                                                          --
--  Copyright (C) 2011  by  Martin Neumann martin@neumanns-mail.de          --
--                                                                          --
--  File name   : usb_tc03.vhd                                              --
--  Author      : Martin Neumann  martin@neumanns-mail.de                   --
--  Description : Copy and rename this file to usb_stimuli.vhd              --
--                before running a new simulation!                          --
--                                                                          --
--==========================================================================--
--                                                                          --
-- Change history                                                           --
--                                                                          --
-- Version / date        Description                                        --
--                                                                          --
-- 01  15 Mar 2013 MN    Initial version                                    --
--                                                                          --
-- End change history                                                       --
--==========================================================================--

LIBRARY work, IEEE;
  USE IEEE.std_logic_1164.ALL;
  USE IEEE.std_logic_arith.ALL;
  USE work.usb_commands.ALL;

ENTITY USB_Stimuli IS PORT(
  -- Test Control Interface --
  USB             : OUT usb_action;
  rst_neg_ext     : OUT STD_LOGIC;
  t_no            : OUT NATURAL
);
END USB_Stimuli;

ARCHITECTURE sim OF usb_stimuli IS

BEGIN
--==========================================================================--
-- All outcommented procedure calls reflect the expected USB Slave response --
--==========================================================================--
  p_stimuli_data : PROCESS
  variable top : NATURAL;
  BEGIN
    list("**********************************");
    list("*                                *");
    list("*       Test USB FS SLAVE        *");
    list("* Init according to Win 7 driver *");
    list("*                                *");
    list("**********************************");
    rst_neg_ext <= '0';
    WAIT FOR 301 ns;
    rst_neg_ext <= '1';
    WAIT FOR 400 ns;
    --***************************************--
    list(T_No, 01);
    send_res(usb);
    sof_token(usb, X"55D");
    sof_token(usb, X"55E");
    sof_token(usb, X"55F");
    setup(usb, X"00",X"0"); -- GET_DESCRIPTOR
    send_D0   (usb, (X"80",X"06",X"00",X"01",X"00",X"00",X"40",X"00"));
    wait_slv  (usb);
--  recv_ACK  (usb);
```

```
    in_token(usb, X"00",X"0");
    wait_slv  (usb);
--  recv_D1   (usb, (X"12",X"01",X"10",X"01",X"02",X"00",X"00",X"40",
--                   X"9A",X"FB",X"9A",X"FB",X"20",X"00",X"00",X"00",
--                   X"00",X"01"));
    send_ACK  (usb);
    out_token(usb, X"00",X"0");
    send_D1   (usb);
    wait_slv  (usb);
--  recv_ACK  (usb);
    send_res(usb);
    list(T_No, 02);
    --***************************************--
    setup(usb, X"00",X"0"); -- SET_ADDRESS
    send_D0   (usb, (X"00",X"05",X"02",X"00",X"00",X"00",X"00",X"00"));
    wait_slv  (usb);
--  recv_ACK  (usb);
    in_token(usb, X"00",X"0");
    wait_slv  (usb);
--  recv_D1   (usb);
    send_ACK  (usb);
    list(T_No, 03);
    --***************************************--
    setup(usb, X"02",X"0"); -- GET_DESCRIPTOR 1
    send_D0   (usb, (X"80",X"06",X"00",X"01",X"00",X"00",X"12",X"00"));
    wait_slv  (usb);
--  recv_ACK  (usb);
    in_token(usb, X"02",X"0");
    wait_slv  (usb);
--  recv_D1   (usb, (X"12",X"01",X"10",X"01",X"02",X"00",X"00",X"40",
--                   X"9A",X"FB",X"9A",X"FB",X"20",X"00",X"00",X"00",
--                   X"00",X"01"));
    send_ACK  (usb);
    out_token(usb, X"02",X"0");
    send_D1   (usb);
    wait_slv  (usb);
--  recv_ACK  (usb);
    list(T_No, 04);
    --***************************************--
    setup(usb, X"02",X"0"); -- GET_DESCRIPTOR 2
    send_D0   (usb, (X"80",X"06",X"00",X"02",X"00",X"00",X"FF",X"00"));
    wait_slv  (usb);
--  recv_ACK  (usb);
    in_token(usb, X"02",X"0");
    wait_slv  (usb);
--  recv_D1   (usb, (X"09",X"02",X"43",X"00",X"02",X"01",X"00",X"80",
--                   X"FA",X"09",X"04",X"00",X"00",X"01",X"02",X"02",
--                   X"01",X"00",X"05",X"24",X"00",X"10",X"01",X"04",
--                   X"24",X"02",X"00",X"05",X"24",X"06",X"00",X"01",
--                   X"05",X"24",X"01",X"00",X"01",X"07",X"05",X"82",
--                   X"03",X"08",X"00",X"FF",X"09",X"04",X"01",X"00",
--                   X"02",X"0A",X"00",X"00",X"00",X"07",X"05",X"81",
--                   X"02",X"40",X"00",X"00",X"07",X"05",X"01",X"02"));
    send_ACK  (usb);
    in_token(usb, X"02",X"0");
    wait_slv  (usb);
--  recv_D0   (usb, (X"40",X"00",X"00"));
    send_ACK  (usb);
    out_token(usb, X"02",X"0");
    send_D1   (usb);
    wait_slv  (usb);
--  recv_ACK  (usb);
```

```
    list(T_No, 05);
    --***************************************--
    setup(usb, X"02",X"0"); -- GET_DESCRIPTOR 1
    send_D0   (usb, (X"80",X"06",X"00",X"01",X"00",X"00",X"12",X"00"));
    wait_slv  (usb);
--  recv_ACK  (usb);
    in_token(usb, X"02",X"0");
    wait_slv  (usb);
--  recv_D1   (usb, (X"12",X"01",X"10",X"01",X"02",X"00",X"00",X"40",
--                   X"9A",X"FB",X"9A",X"FB",X"20",X"00",X"00",X"00",
--                   X"00",X"01"));
    send_ACK  (usb);
    out_token(usb, X"02",X"0");
    send_D1   (usb);
    wait_slv  (usb);
--  recv_ACK  (usb);
    list(T_No, 06);
    --***************************************--
    setup(usb, X"02",X"0"); -- GET_DESCRIPTOR 2
    send_D0   (usb, (X"80",X"06",X"00",X"02",X"00",X"00",X"09",X"01"));
    wait_slv  (usb);
--  recv_ACK  (usb);
    in_token(usb, X"02",X"0");
    wait_slv  (usb);
--  recv_D1   (usb, (X"09",X"02",X"43",X"00",X"02",X"01",X"00",X"80",
--                   X"FA",X"09",X"04",X"00",X"00",X"01",X"02",X"02",
--                   X"01",X"00",X"05",X"24",X"00",X"10",X"01",X"04",
--                   X"24",X"02",X"00",X"05",X"24",X"06",X"00",X"01",
--                   X"05",X"24",X"01",X"00",X"01",X"07",X"05",X"82",
--                   X"03",X"08",X"00",X"FF",X"09",X"04",X"01",X"00",
--                   X"02",X"0A",X"00",X"00",X"00",X"07",X"05",X"81",
--                   X"02",X"40",X"00",X"00",X"07",X"05",X"01",X"02"));
    send_ACK  (usb);
    in_token(usb, X"02",X"0");
    wait_slv  (usb);
--  recv_D0   (usb, (X"40",X"00",X"00"));
    send_ACK  (usb);
    out_token(usb, X"02",X"0");
    send_D1   (usb);
    wait_slv  (usb);
--  recv_ACK  (usb);
    list(T_No, 07);
    --***************************************--
    setup(usb, X"02",X"0"); -- SET_CONFIGURATION
    send_D0   (usb, (X"00",X"09",X"01",X"00",X"00",X"00",X"00",X"00"));
    wait_slv  (usb);
--  recv_ACK  (usb);
    in_token(usb, X"02",X"0");
    wait_slv  (usb);
--  recv_D1   (usb);
    send_ACK  (usb);
    list(T_No, 08);
    --***************************************--
    setup(usb, X"02",X"0");
    send_D0   (usb, (X"A1",X"21",X"00",X"00",X"00",X"00",X"07",X"00"));
    wait_slv  (usb);
--  recv_ACK  (usb);
    in_token(usb, X"02",X"0");
    wait_slv  (usb);
--  recv_D1   (usb);
    send_ACK  (usb);
    out_token(usb, X"02",X"0");
    send_D1   (usb);
    wait_slv  (usb);
--  recv_ACK  (usb);
    list(T_No, 09);
    --***************************************--
    setup(usb, X"02",X"0");
    send_D0   (usb, (X"21",X"22",X"00",X"00",X"00",X"00",X"00",X"00"));
    wait_slv  (usb);
--  recv_ACK  (usb);
    in_token(usb, X"02",X"0");
    wait_slv  (usb);
--  recv_D1   (usb);
    send_ACK  (usb);
    list("write and read 3x 64 bytes to - from engine 1");
--===========================================================================--
--  Win 7 configuration sequence has been completed - applicatioon starting --
--  First engine 1 transfer after setup -> data toggle bit starts with 0 !!  --
--===========================================================================--
    list(T_No, 10);
    out_token(usb, X"02",X"1");
    send_D0   (usb, (X"00",X"01",X"02",X"03",X"04",X"05",X"06",X"07",
                     X"08",X"09",X"0A",X"0B",X"0C",X"0D",X"0E",X"0F",
                     X"10",X"11",X"12",X"13",X"14",X"15",X"16",X"17",
                     X"18",X"19",X"1A",X"1B",X"1C",X"1D",X"1E",X"1F",
                     X"20",X"21",X"22",X"23",X"24",X"25",X"26",X"27",
                     X"28",X"29",X"2A",X"2B",X"2C",X"2D",X"2E",X"2F",
                     X"30",X"31",X"32",X"33",X"34",X"35",X"36",X"37",
                     X"38",X"39",X"3A",X"3B",X"3C",X"3D",X"3E",X"3F"));
    wait_slv  (usb);
--  recv_ACK  (usb);
    out_token(usb, X"02",X"1");
    send_D1   (usb, (X"40",X"41",X"42",X"43",X"44",X"45",X"46",X"47",
                     X"48",X"49",X"4A",X"4B",X"4C",X"4D",X"4E",X"4F",
                     X"50",X"51",X"52",X"53",X"54",X"55",X"56",X"57",
                     X"58",X"59",X"5A",X"5B",X"5C",X"5D",X"5E",X"5F",
                     X"60",X"61",X"62",X"63",X"64",X"65",X"66",X"67",
                     X"68",X"69",X"6A",X"6B",X"6C",X"6D",X"6E",X"6F",
                     X"70",X"71",X"72",X"73",X"74",X"75",X"76",X"77",
                     X"78",X"79",X"7A",X"7B",X"7C",X"7D",X"7E",X"7F"));
    wait_slv  (usb);
--  recv_ACK  (usb);
    out_token(usb, X"02",X"1");
    send_D0   (usb, (X"80",X"81",X"82",X"83",X"84",X"85",X"86",X"87",
                     X"88",X"89",X"8A",X"8B",X"8C",X"8D",X"8E",X"8F",
                     X"90",X"91",X"92",X"93",X"94",X"95",X"96",X"97",
                     X"98",X"99",X"9A",X"9B",X"9C",X"9D",X"9E",X"9F",
                     X"A0",X"A1",X"A2",X"A3",X"A4",X"A5",X"A6",X"A7",
                     X"A8",X"A9",X"AA",X"AB",X"AC",X"AD",X"AE",X"AF",
                     X"B0",X"B1",X"B2",X"B3",X"B4",X"B5",X"B6",X"B7",
                     X"B8",X"B9",X"BA",X"BB",X"BC",X"BD",X"BE",X"BF"));
    wait_slv  (usb);
--  recv_ACK  (usb);
    list(T_No, 11);
    list("read 1st 64 bytes");
    in_token(usb, X"02",X"1");
    wait_slv  (usb);
 -- recv_D0   (usb, (X"00",X"10",X"20",X"30",X"40",X"50",X"60",X"70",
 --                  X"80",X"90",X"A0",X"B0",X"C0",X"D0",X"E0",X"F0",
 --                  X"01",X"11",X"21",X"31",X"41",X"51",X"61",X"71",
 --                  X"81",X"91",X"A1",X"B1",X"C1",X"D1",X"E1",X"F1",
 --                  X"02",X"12",X"22",X"32",X"42",X"52",X"62",X"72",
 --                  X"82",X"92",X"A2",X"B2",X"C2",X"D2",X"E2",X"F2",
 --                  X"03",X"13",X"23",X"33",X"43",X"53",X"63",X"73",
 --                  X"83",X"93",X"A3",X"B3",X"C3",X"D3",X"E3",X"F3"));
    send_ACK  (usb);
    list("read 2nd 64 bytes");
    in_token(usb, X"02",X"1");
    wait_slv  (usb);
 -- recv_D1   (usb, (X"04",X"14",X"24",X"34",X"44",X"54",X"64",X"74",
 --                  X"84",X"94",X"A4",X"B4",X"C4",X"D4",X"E4",X"F4",
 --                  X"05",X"15",X"25",X"35",X"45",X"55",X"65",X"75",
 --                  X"85",X"95",X"A5",X"B5",X"C5",X"D5",X"E5",X"F5",
 --                  X"06",X"16",X"26",X"36",X"46",X"56",X"66",X"76",
 --                  X"86",X"96",X"A6",X"B6",X"C6",X"D6",X"E6",X"F6",
 --                  X"07",X"17",X"27",X"37",X"47",X"57",X"67",X"77",
 --                  X"87",X"97",X"A7",X"B7",X"C7",X"D7",X"E7",X"F7"));
    send_ACK  (usb);
    in_token(usb, X"02",X"1");
    wait_slv  (usb);
 -- recv_D0   (usb, (X"08",X"18",X"28",X"38",X"48",X"58",X"68",X"78",
 --                  X"88",X"98",X"A8",X"B8",X"C8",X"D8",X"E8",X"F8",
 --                  X"09",X"19",X"29",X"39",X"49",X"59",X"69",X"79",
 --                  X"89",X"99",X"A9",X"B9",X"C9",X"D9",X"E9",X"F9",
 --                  X"0A",X"1A",X"2A",X"3A",X"4A",X"5A",X"6A",X"7A",
 --                  X"8A",X"9A",X"AA",X"BA",X"CA",X"DA",X"EA",X"FA",
 --                  X"0B",X"1B",X"2B",X"3B",X"4B",X"5B",X"6B",X"7B",
 --                  X"8B",X"9B",X"AB",X"BB",X"CB",X"DB",X"EB",X"FB"));
    send_ACK  (usb);
    list("write and read 1x 64 bytes to - from engine 1");
    --**************************************************--
    list(T_No, 32);
    out_token(usb, X"02",X"1");
    send_D1   (usb, (X"C0",X"C1",X"C2",X"C3",X"C4",X"C5",X"C6",X"C7",
                     X"C8",X"C9",X"CA",X"CB",X"CC",X"CD",X"CE",X"CF",
                     X"D0",X"D1",X"D2",X"D3",X"D4",X"D5",X"D6",X"D7",
                     X"D8",X"D9",X"DA",X"DB",X"DC",X"DD",X"DE",X"DF",
                     X"E0",X"E1",X"E2",X"E3",X"E4",X"E5",X"E6",X"E7",
                     X"E8",X"E9",X"EA",X"EB",X"EC",X"ED",X"EE",X"EF",
                     X"F0",X"F1",X"F2",X"F3",X"F4",X"F5",X"F6",X"F7",
                     X"F8",X"F9",X"FA",X"FB",X"FC",X"FD",X"FE",X"FF"));
    wait_slv  (usb);
    list(T_No, 13);
    in_token(usb, X"02",X"1");
    wait_slv  (usb);
 -- recv_D1   (usb, (X"0C",X"1C",X"2C",X"3C",X"4C",X"5C",X"6C",X"7C",
 --                  X"8C",X"9C",X"AC",X"BC",X"CC",X"DC",X"EC",X"FC",
 --                  X"0D",X"1D",X"2D",X"3D",X"4D",X"5D",X"6D",X"7D",
 --                  X"8D",X"9D",X"AD",X"BD",X"CD",X"DD",X"ED",X"FD",
 --                  X"0E",X"1E",X"2E",X"3E",X"4E",X"5E",X"6E",X"7E",
 --                  X"8E",X"9E",X"AE",X"BE",X"CE",X"DE",X"EE",X"FE",
 --                  X"0F",X"1F",X"2F",X"3F",X"4F",X"5F",X"6F",X"7F",
 --                  X"8F",X"9F",X"AF",X"BF",X"CF",X"DF",X"EF",X"FF"));
    send_ACK  (usb);
    list(T_No, 14);
    list("test for more data - nothing");
    in_token(usb, X"02",X"1");
    wait_slv  (usb);
 -- recv_D0   (usb);
    send_ACK  (usb);

    ASSERT FALSE REPORT"End of Test" SEVERITY FAILURE;
  END PROCESS;

END sim;
```

Listing 1**:** Test case usb_tc03.vhd renamed to usb_stimuli.vhd

When running this test case, the USB Monitor will produce the following Result.out file:

```
                  **********************************
                  *                                *
                  *       Test USB FS SLAVE        *
                  * Init according to Win 7 driver *
                  *                                *
                  **********************************
   508.343 ns  USB lines at SE0 for      283.339 ns
               Test_No 1
   6025.12 ns  USB Reset detected for     5083.435 ns
   7358.48 ns  Send SOF-Token: Frame No 0x55D, CRC5 0x12
 10441.875 ns  Send SOF-Token: Frame No 0x55E, CRC5 0x1A
  13525.27 ns  Send SOF-Token: Frame No 0x55F, CRC5 0x05
     16692 ns  Send Setup: Address 0x00, Endpoint 0x0, CRC5 0x02
 19775.395 ns  Send Data0 0x80 0x06 0x00 0x01 0x00 0x00 0x40 0x00 0xDD 0x94
 28542.237 ns  Recv ACK
 30442.275 ns  Send IN-Token: Address 0x00, Endpoint 0x0, CRC5 0x02
 33875.677 ns  Recv Data1 0x12 0x01 0x10 0x01 0x02 0x00 0x00 0x40 0x9A 0xFB 0x9A 0xFB 0x20 0x00 0x00 0x00
 45209.237 ns       ..... 0x00 0x01 0xB4 0x2C
 49109.315 ns  Send ACK
  50859.35 ns  Send OUT-Token: Address 0x00, Endpoint 0x0, CRC5 0x02
 53942.745 ns  Send Data1 0x00 0x00
 57376.147 ns  Recv ACK
               Test_No 2
  63026.26 ns  USB Reset detected for     5083.435 ns
  64359.62 ns  Send Setup: Address 0x00, Endpoint 0x0, CRC5 0x02
 67443.015 ns  Send Data0 0x00 0x05 0x02 0x00 0x00 0x00 0x00 0x00 0xEB 0x16
 76209.857 ns  Recv ACK
 78109.895 ns  Send IN-Token: Address 0x00, Endpoint 0x0, CRC5 0x02
 81543.297 ns  Recv Data1 0x00 0x00
 84776.695 ns  Send ACK
               Test_No 3
  86526.73 ns  Send Setup: Address 0x02, Endpoint 0x0, CRC5 0x15
 89610.125 ns  Send Data0 0x80 0x06 0x00 0x01 0x00 0x00 0x12 0x00 0xE0 0xF4
 98376.967 ns  Recv ACK
100277.005 ns  Send IN-Token: Address 0x02, Endpoint 0x0, CRC5 0x15
103710.407 ns  Recv Data1 0x12 0x01 0x10 0x01 0x02 0x00 0x00 0x40 0x9A 0xFB 0x9A 0xFB 0x20 0x00 0x00 0x00
115043.967 ns       ..... 0x00 0x01 0xB4 0x2C
118944.045 ns  Send ACK
 120694.08 ns  Send OUT-Token: Address 0x02, Endpoint 0x0, CRC5 0x15
123777.475 ns  Send Data1 0x00 0x00
127210.877 ns  Recv ACK
               Test_No 4
129110.915 ns  Send Setup: Address 0x02, Endpoint 0x0, CRC5 0x15
 132194.31 ns  Send Data0 0x80 0x06 0x00 0x02 0x00 0x00 0xFF 0x00 0xE9 0xA4
141044.487 ns  Recv ACK
142944.525 ns  Send IN-Token: Address 0x02, Endpoint 0x0, CRC5 0x15
146377.927 ns  Recv Data1 0x09 0x02 0x43 0x00 0x02 0x01 0x00 0x80 0xFA 0x09 0x04 0x00 0x00 0x01 0x02 0x02
157794.822 ns       ..... 0x01 0x00 0x05 0x24 0x00 0x10 0x01 0x04 0x24 0x02 0x00 0x05 0x24 0x06 0x00 0x01
168461.702 ns       ..... 0x05 0x24 0x01 0x00 0x01 0x07 0x05 0x82 0x03 0x08 0x00 0xFF 0x09 0x04 0x01 0x00
179211.917 ns       ..... 0x02 0x0A 0x00 0x00 0x00 0x07 0x05 0x81 0x02 0x40 0x00 0x00 0x07 0x05 0x01 0x02
189878.797 ns       ..... 0x38 0x89
192445.515 ns  Send ACK
 194195.55 ns  Send IN-Token: Address 0x02, Endpoint 0x0, CRC5 0x15
197628.952 ns  Recv Data0 0x40 0x00 0x00 0x8F 0xEB
 202862.39 ns  Send ACK
204612.425 ns  Send OUT-Token: Address 0x02, Endpoint 0x0, CRC5 0x15
 207695.82 ns  Send Data1 0x00 0x00
211129.222 ns  Recv ACK
               Test_No 5
 213029.26 ns  Send Setup: Address 0x02, Endpoint 0x0, CRC5 0x15
216112.655 ns  Send Data0 0x80 0x06 0x00 0x01 0x00 0x00 0x12 0x00 0xE0 0xF4
224879.497 ns  Recv ACK
226779.535 ns  Send IN-Token: Address 0x02, Endpoint 0x0, CRC5 0x15
230212.937 ns  Recv Data1 0x12 0x01 0x10 0x01 0x02 0x00 0x00 0x40 0x9A 0xFB 0x9A 0xFB 0x20 0x00 0x00 0x00
241546.497 ns       ..... 0x00 0x01 0xB4 0x2C
245446.575 ns  Send ACK
 247196.61 ns  Send OUT-Token: Address 0x02, Endpoint 0x0, CRC5 0x15
250280.005 ns  Send Data1 0x00 0x00
253713.407 ns  Recv ACK
```

```
               Test_No 6
255613.445 ns  Send Setup: Address 0x02, Endpoint 0x0, CRC5 0x15
 258696.84 ns  Send Data0 0x80 0x06 0x00 0x02 0x00 0x00 0x09 0x01 0x6F 0xC4
267463.682 ns  Recv ACK
 269363.72 ns  Send IN-Token: Address 0x02, Endpoint 0x0, CRC5 0x15
272797.122 ns  Recv Data1 0x09 0x02 0x43 0x00 0x02 0x01 0x00 0x80 0xFA 0x09 0x04 0x00 0x00 0x01 0x02 0x02
284214.017 ns       ..... 0x01 0x00 0x05 0x24 0x00 0x10 0x01 0x04 0x24 0x02 0x00 0x05 0x24 0x06 0x00 0x01
294880.897 ns       ..... 0x05 0x24 0x01 0x00 0x01 0x07 0x05 0x82 0x03 0x08 0x00 0xFF 0x09 0x04 0x01 0x00
305631.112 ns       ..... 0x02 0x0A 0x00 0x00 0x00 0x07 0x05 0x81 0x02 0x40 0x00 0x00 0x07 0x05 0x01 0x02
316297.992 ns       ..... 0x38 0x89
 318864.71 ns  Send ACK
320614.745 ns  Send IN-Token: Address 0x02, Endpoint 0x0, CRC5 0x15
324048.147 ns  Recv Data0 0x40 0x00 0x00 0x8F 0xEB
329281.585 ns  Send ACK
 331031.62 ns  Send OUT-Token: Address 0x02, Endpoint 0x0, CRC5 0x15
334115.015 ns  Send Data1 0x00 0x00
337548.417 ns  Recv ACK
               Test_No 7
339448.455 ns  Send Setup: Address 0x02, Endpoint 0x0, CRC5 0x15
 342531.85 ns  Send Data0 0x00 0x09 0x01 0x00 0x00 0x00 0x00 0x00 0x27 0x25
351298.692 ns  Recv ACK
 353198.73 ns  Send IN-Token: Address 0x02, Endpoint 0x0, CRC5 0x15
356632.132 ns  Recv Data1 0x00 0x00
 359865.53 ns  Send ACK
               Test_No 8
361615.565 ns  Send Setup: Address 0x02, Endpoint 0x0, CRC5 0x15
 364698.96 ns  Send Data0 0xA1 0x21 0x00 0x00 0x00 0x00 0x07 0x00 0x47 0x72
373465.802 ns  Recv ACK
 375365.84 ns  Send IN-Token: Address 0x02, Endpoint 0x0, CRC5 0x15
378799.242 ns  Recv Data1 0x00 0x00
 382032.64 ns  Send ACK
383782.675 ns  Send OUT-Token: Address 0x02, Endpoint 0x0, CRC5 0x15
 386866.07 ns  Send Data1 0x00 0x00
390299.472 ns  Recv ACK
               Test_No 9
 392199.51 ns  Send Setup: Address 0x02, Endpoint 0x0, CRC5 0x15
395282.905 ns  Send Data0 0x21 0x22 0x00 0x00 0x00 0x00 0x00 0x00 0x7E 0x22
404133.082 ns  Recv ACK
 406033.12 ns  Send IN-Token: Address 0x02, Endpoint 0x0, CRC5 0x15
409466.522 ns  Recv Data1 0x00 0x00
 412699.92 ns  Send ACK
               write and read 3x 64 bytes to - from engine 1
               Test_No 10
414449.955 ns  Send OUT-Token: Address 0x02, Endpoint 0x1, CRC5 0x03
 417533.35 ns  Send Data0 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A 0x0B 0x0C 0x0D 0x0E 0x0F
 428866.91 ns       ..... 0x10 0x11 0x12 0x13 0x14 0x15 0x16 0x17 0x18 0x19 0x1A 0x1B 0x1C 0x1D 0x1E 0x1F
 439533.79 ns       ..... 0x20 0x21 0x22 0x23 0x24 0x25 0x26 0x27 0x28 0x29 0x2A 0x2B 0x2C 0x2D 0x2E 0x2F
 450200.67 ns       ..... 0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39 0x3A 0x3B 0x3C 0x3D 0x3E 0x3F
460950.885 ns       ..... 0x26 0xF7
 463700.94 ns  Recv ACK
465617.645 ns  Send OUT-Token: Address 0x02, Endpoint 0x1, CRC5 0x03
 468701.04 ns  Send Data1 0x40 0x41 0x42 0x43 0x44 0x45 0x46 0x47 0x48 0x49 0x4A 0x4B 0x4C 0x4D 0x4E 0x4F
  480034.6 ns       ..... 0x50 0x51 0x52 0x53 0x54 0x55 0x56 0x57 0x58 0x59 0x5A 0x5B 0x5C 0x5D 0x5E 0x5F
 490701.48 ns       ..... 0x60 0x61 0x62 0x63 0x64 0x65 0x66 0x67 0x68 0x69 0x6A 0x6B 0x6C 0x6D 0x6E 0x6F
 501368.36 ns       ..... 0x70 0x71 0x72 0x73 0x74 0x75 0x76 0x77 0x78 0x79 0x7A 0x7B 0x7C 0x7D 0x7E 0x7F
 512201.91 ns       ..... 0x9B 0xBA
514951.965 ns  Recv ACK
 516868.67 ns  Send OUT-Token: Address 0x02, Endpoint 0x1, CRC5 0x03
519952.065 ns  Send Data0 0x80 0x81 0x82 0x83 0x84 0x85 0x86 0x87 0x88 0x89 0x8A 0x8B 0x8C 0x8D 0x8E 0x8F
531268.958 ns       ..... 0x90 0x91 0x92 0x93 0x94 0x95 0x96 0x97 0x98 0x99 0x9A 0x9B 0x9C 0x9D 0x9E 0x9F
542019.173 ns       ..... 0xA0 0xA1 0xA2 0xA3 0xA4 0xA5 0xA6 0xA7 0xA8 0xA9 0xAA 0xAB 0xAC 0xAD 0xAE 0xAF
552686.053 ns       ..... 0xB0 0xB1 0xB2 0xB3 0xB4 0xB5 0xB6 0xB7 0xB8 0xB9 0xBA 0xBB 0xBC 0xBD 0xBE 0xBF
563436.268 ns       ..... 0x5C 0x6C
 566202.99 ns  Recv ACK
               Test_No 11
               read 1st 64 bytes
568103.028 ns  Send IN-Token: Address 0x02, Endpoint 0x1, CRC5 0x03
 571536.43 ns  Recv Data0 0x00 0x10 0x20 0x30 0x40 0x50 0x60 0x70 0x80 0x90 0xA0 0xB0 0xC0 0xD0 0xE0 0xF0
 582869.99 ns       ..... 0x01 0x11 0x21 0x31 0x41 0x51 0x61 0x71 0x81 0x91 0xA1 0xB1 0xC1 0xD1 0xE1 0xF1
 593536.87 ns       ..... 0x02 0x12 0x22 0x32 0x42 0x52 0x62 0x72 0x82 0x92 0xA2 0xB2 0xC2 0xD2 0xE2 0xF2
604287.085 ns       ..... 0x03 0x13 0x23 0x33 0x43 0x53 0x63 0x73 0x83 0x93 0xA3 0xB3 0xC3 0xD3 0xE3 0xF3
614953.965 ns       ..... 0x86 0xE0
617520.683 ns  Send ACK
               read 2nd 64 bytes
619270.718 ns  Send IN-Token: Address 0x02, Endpoint 0x1, CRC5 0x03
 622704.12 ns  Recv Data1 0x04 0x14 0x24 0x34 0x44 0x54 0x64 0x74 0x84 0x94 0xA4 0xB4 0xC4 0xD4 0xE4 0xF4
 634037.68 ns       ..... 0x05 0x15 0x25 0x35 0x45 0x55 0x65 0x75 0x85 0x95 0xA5 0xB5 0xC5 0xD5 0xE5 0xF5
 644704.56 ns       ..... 0x06 0x16 0x26 0x36 0x46 0x56 0x66 0x76 0x86 0x96 0xA6 0xB6 0xC6 0xD6 0xE6 0xF6
```

```
655454.775 ns       ..... 0x07 0x17 0x27 0x37 0x47 0x57 0x67 0x77 0x87 0x97 0xA7 0xB7 0xC7 0xD7 0xE7 0xF7
 666204.99 ns       ..... 0x5C 0x78
668771.708 ns  Send ACK
670521.743 ns  Send IN-Token: Address 0x02, Endpoint 0x1, CRC5 0x03
673955.145 ns  Recv Data0 0x08 0x18 0x28 0x38 0x48 0x58 0x68 0x78 0x88 0x98 0xA8 0xB8 0xC8 0xD8 0xE8 0xF8
 685372.04 ns       ..... 0x09 0x19 0x29 0x39 0x49 0x59 0x69 0x79 0x89 0x99 0xA9 0xB9 0xC9 0xD9 0xE9 0xF9
 696038.92 ns       ..... 0x0A 0x1A 0x2A 0x3A 0x4A 0x5A 0x6A 0x7A 0x8A 0x9A 0xAA 0xBA 0xCA 0xDA 0xEA 0xFA
706789.135 ns       ..... 0x0B 0x1B 0x2B 0x3B 0x4B 0x5B 0x6B 0x7B 0x8B 0x9B 0xAB 0xBB 0xCB 0xDB 0xEB 0xFB
 717539.35 ns       ..... 0x31 0x91
720106.068 ns  Send ACK
               write and read 1x 64 bytes to - from engine 1
               Test_No 32
721856.103 ns  Send OUT-Token: Address 0x02, Endpoint 0x1, CRC5 0x03
724939.498 ns  Send Data1 0xC0 0xC1 0xC2 0xC3 0xC4 0xC5 0xC6 0xC7 0xC8 0xC9 0xCA 0xCB 0xCC 0xCD 0xCE 0xCF
736356.393 ns       ..... 0xD0 0xD1 0xD2 0xD3 0xD4 0xD5 0xD6 0xD7 0xD8 0xD9 0xDA 0xDB 0xDC 0xDD 0xDE 0xDF
747106.608 ns       ..... 0xE0 0xE1 0xE2 0xE3 0xE4 0xE5 0xE6 0xE7 0xE8 0xE9 0xEA 0xEB 0xEC 0xED 0xEE 0xEF
757940.158 ns       ..... 0xF0 0xF1 0xF2 0xF3 0xF4 0xF5 0xF6 0xF7 0xF8 0xF9 0xFA 0xFB 0xFC 0xFD 0xFE 0xFF
769273.718 ns       ..... 0xE1 0x21
 772040.44 ns  Recv ACK
               Test_No 13
773940.478 ns  Send IN-Token: Address 0x02, Endpoint 0x1, CRC5 0x03
 777373.88 ns  Recv Data1 0x0C 0x1C 0x2C 0x3C 0x4C 0x5C 0x6C 0x7C 0x8C 0x9C 0xAC 0xBC 0xCC 0xDC 0xEC 0xFC
788790.775 ns       ..... 0x0D 0x1D 0x2D 0x3D 0x4D 0x5D 0x6D 0x7D 0x8D 0x9D 0xAD 0xBD 0xCD 0xDD 0xED 0xFD
 799540.99 ns       ..... 0x0E 0x1E 0x2E 0x3E 0x4E 0x5E 0x6E 0x7E 0x8E 0x9E 0xAE 0xBE 0xCE 0xDE 0xEE 0xFE
 810374.54 ns       ..... 0x0F 0x1F 0x2F 0x3F 0x4F 0x5F 0x6F 0x7F 0x8F 0x9F 0xAF 0xBF 0xCF 0xDF 0xEF 0xFF
  821708.1 ns       ..... 0xEB 0x09
824274.818 ns  Send ACK
               Test_No 14
               test for more data - nothing
826024.853 ns  Send IN-Token: Address 0x02, Endpoint 0x1, CRC5 0x03
829458.255 ns  Recv Data0 0x00 0x00
832691.653 ns  Send ACK
```

Listing 2**:** Result.out result file from test case usb_tc03.vhd

All lines without a time stamp are user comments, created via a list() command. All other lines are written by the USB Monitor. It permanently monitors the two USB lines and whenever it detects a sync signal, it evaluates the command, regardless if it is caused by the USB Master or the USB Slave. Only the direction signal is obtained from the USB Master. The USB reset detector checks the USB lines for SE0 events over 200 ns, a reset is signaled as soon as SE0 exceeds 2,5 µs.