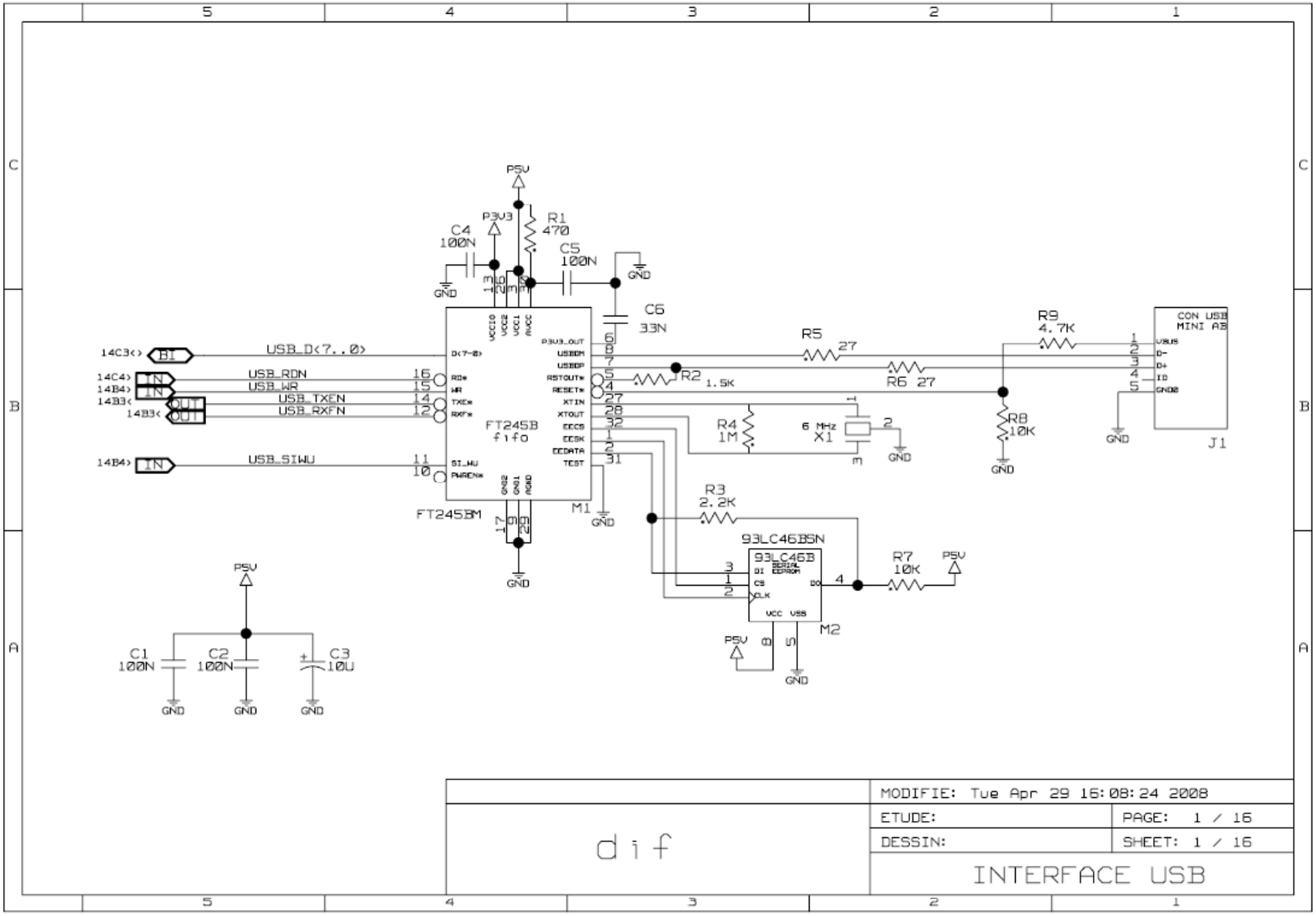
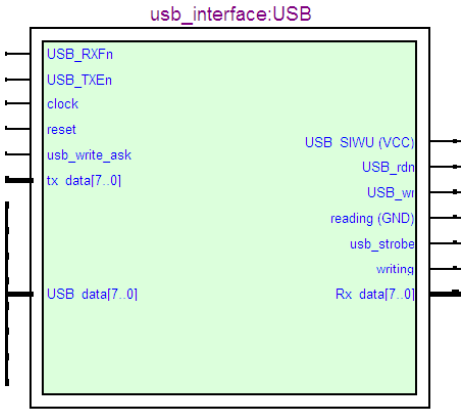
USB Interface
Used for the DHCAL DIF

Inspired from Clement's code


5
4
3
2
1
C
B
A
P3V3
R1
470
C4
100N
C5
100N
C6
33N
GND
FT245B
fifo
FT245BM
M1
USB_D<7..0>
USB_RDN
USB_WR
USB_TXEN
USB_RXFN
USB_SIWU
R2
1.5K
R5
27
R6 27
R9
4.7K
R4
1M
6 MHz
X1
R8
10K
CON USB
MINI AB
J1
R3
2.2K
93LC46BSN
93LC46B
R7
10K
M2
C1
100N
C2
100N
C3
10U
MODIFIE: Tue Apr 29 16:08:24 2008
ETUDE:
PAGE: 1 / 16
DESSIN:
SHEET: 1 / 16
INTERFACE USB
dif

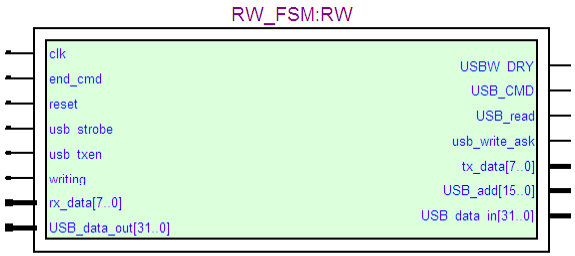
Firmware Schematics View

USB_interface



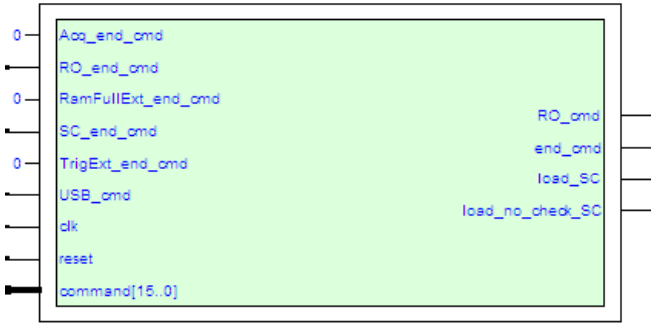
USB interface
- 1 SM for USB read (FPGA to PC)
- 1 SM for USB write (PC to FPGA)

RW_FSM



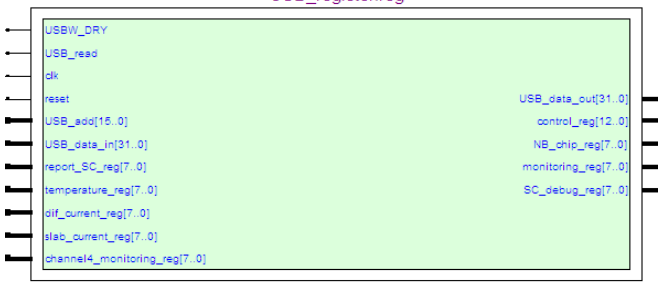
SM which determines if USB accesses are registers access (16 bit address, 32 bit data) or commands

Cmd_decoder



Handles all commands

USB_register



Handles all the FPGA registers (R/W)

USB Interface

```
1   --julie 8 february 2008 from clement's code
2   --This entity contains the USB_interface of the FPGA
3   --It is made of 2 components, one for USB writes (PC to FPGA)
4   --and one component for USB reads (FPGA to PC).
5
6   library ieee;
7   use ieee.std_logic_1164.all;
8   use ieee.std_logic_arith.all;
9
10  entity usb_interface is
11  Port(
12      clock       : in std_logic; --40 MHz external clock
13      reset       : in std_logic; --global reset
14      USB_TXEn    : in std_logic; --external USB fifo signals
15      USB_RXFn    : in std_logic; --external USB fifo signals
16      usb_write_ask   : in std_logic; --RW_FSM asks for sending data to the PC
17      tx_data     : in std_logic_vector(7 downto 0); --data to be transmitted to the PC
18
19      USB_data    : inout std_logic_vector (7 downto 0);--external
20      Rx_data     : out std_logic_vector(7 downto 0); -- data from the PC
21      USB_rdn     : out std_logic;--external USB fifo signals
22      USB_wr      : out std_logic;--external USB fifo signals
23      USB_SIWU    : out std_logic;--external USB fifo signals
24      usb_strobe  : out std_logic;--data from PC strobe
25      writing     : out std_logic; --usb interface is writing
26      reading     : out std_logic --usb interface is reading
27      --to be completed
28    );
29  end usb_interface;
```

RW_FSM

```
1  --Julie 5 MArch 2008 from Clement's code
2  --state machine which determines :
3      --if USB accesses are comands or register accesses
4          --add(15) = 1 Mode command
5          --add(15) = 0 mode register
6      --if register access, treat read and write (32 bits access)
7          --add(14) = 1 : read access : the PC sends 2 bytes for address and waits 4 bytes for data from the PC
8          --add(14) = 0 : write access : the PC sends 2 bytes for address and 4 bytes for data
9  --All accesses are MSB first
10
11
12 library ieee;
13 use ieee.std_logic_1164.all;
14 use ieee.std_logic_arith.all;
15
16 entity RW_FSM is
17 port(
18     clk     : in std_logic;     --40 MHz clock
19     reset       : in std_logic;     --general reset
20     rx_data     : in  std_logic_vector (7 downto 0);    --8 bits data from the USB interface
21     usb_strobe  : in std_logic;     --Input USB byte is available (for entity USB_reg)
22     writing     : in std_logic;     --USB is writing
23     usb_txen    : in std_logic;     --external USB fifo signals
24     tx_data     : out std_logic_vector (7 downto 0);    --8 bits data from the USB interface
25     usb_write_ask   : out std_logic;    --RW_FSM asks for sending data to the PC
26     USB_add      : out std_logic_vector (15 downto 0);  --16 bit address register
27     USB_data_in  : out std_logic_vector (31 downto 0);  --32 bit data for register read (for entity USB_reg)
28     USB_data_out     : in std_logic_vector (31 downto 0);   --32 bit data for register update (WR) (from entity U
29     USBW_DRY     : out std_logic;   --usb 32 bit data ready to be written
30     USB_read     : out std_logic;   --USB register read signal
31     USB_CMD     : out std_logic;    --one command is active
32     end_cmd     : in std_logic );   -- the current command is finished
33 end RW_FSM;
34
```

Cmd_decoder

```
1  --julie 5 MArch 2008 from Clement's code
2  --state machine for the USB command decoder
3  --This entity is only used in command mode (add bit 15 = 1)
4  --It decodes commands, send the corresponding order
5  --and wait for the command completion before returning the signal
6  --end_command to the RW_FSM entity.
7
8
9  library ieee;
10 use ieee.std_logic_1164.all;
11 use ieee.std_logic_arith.all;
12
13 entity cmd_decoder is
14 port(
15      clk     : in std_logic; --40 MHz oscillator clock
16      reset       : in std_logic; --general reset
17      USB_cmd     : in std_logic; --one command is active
18      command     : in  std_logic_vector (15 downto 0); --command value
19      SC_end_cmd      : in std_logic; --Slow control is ended
20      Acq_end_cmd : in std_logic; --acquisition is ended
21      RO_end_cmd  : in std_logic; --RO is ended
22      RamFullExt_end_cmd: in std_logic; --ramfullext signal is generated
23      TrigExt_end_cmd : in std_logic; --trigger ext signal is generated
24
25      end_cmd     : out std_logic; --the current command is finished
26      load_SC     : out std_logic; --load the slow control
27      load_no_check_SC : out std_logic; --load and check the slow control
28      start_acq_cmd   : out std_logic; --start the acquisition
29      RamFullExt_cmd  : out std_logic; --send ramfullext signal to the hardrocs
30      TrigExt_cmd : out std_logic; --send trigger ext signal to the hardrocs
31      RO_cmd      : out std_logic -- start the digital read out
32      );
33 end cmd_decoder;
```

USB_register

```
1  --Julie March 2008
2  --This entity contains all the FPGA mapped registers
3  --They are accessible through the USB interface
4
5  library IEEE ;
6  use IEEE.std_logic_1164.all ;
7  use IEEE.std_logic_arith.all;
8  use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10 entity USB_register is
11   port (
12     clk          : in std_logic;--40 MHz external clock
13     reset            : in std_logic;--global reset
14     USB_add          : in std_logic_vector (15 downto 0);--register address
15     USB_data_in      : in std_logic_vector (31 downto 0);--register data in (from PC)
16     USB_data_out         : out std_logic_vector (31 downto 0); --register data out (to the PC)
17     USBW_DRY         : in std_logic; --usb 32 bit data ready to be written
18     USB_read         : in std_logic;--from RW_FSM
19
20     report_SC_reg        : in std_logic_vector (7 downto 0); --SC report register
21     control_reg      : out std_logic_vector (12 downto 0);
22     NB_chip_reg      : out std_logic_vector (7 downto 0); --number of hardroc chips
23     temperature_reg      : in std_logic_vector (7 downto 0); --current temperature
24     dif_current_reg      : in std_logic_vector (7 downto 0); --DIF power consumption
25     slab_current_reg     : in std_logic_vector (7 downto 0); --slab power consumption
26     channel4_monitoring_reg : in std_logic_vector (7 downto 0); --free ADC 4th channel
27     monitoring_reg       : out std_logic_vector (7 downto 0); --monitoring control register
28     SC_debug_reg         : out std_logic_vector (7 downto 0)--Slow control debug register
29
30 );
31 end USB_register;
```

False read out
(Example of FPGA to PC data transfer)

Entity declaration

```
entity faux_ro is
Port(
    clock       : in std_logic; --40 MHz external clock
    reset       : in std_logic; --global reset
    USB_TXEn    : in std_logic; --external USB fifo signals
    RO_cmd      : in std_logic;
    writing     : in std_logic;
    RO_end_cmd  : out std_logic;
    usb_write_ask   : out std_logic; --asks for sending data to the
    tx_data     : out std_logic_vector(7 downto 0) --data to be tran
  );
end faux_ro;
```

Port Map

```
  FauxRO :  faux_ro
Port map(
    clock       => clock,
    reset       => reset,
    USB_TXEn    => USB_TXEn,
    RO_cmd      => RO_cmd,
    writing     => writing,
    RO_end_cmd  => RO_end_cmd,
    usb_write_ask   => usb_write_ask_RO,
    tx_data     => tx_data_RO
  );
```

MUX in top level entity

```
P2 : process(clock, reset)
begin
    If reset ='1' then
        readout <= '0';
    elsif rising_edge (clock) then
        if RO_cmd ='1' then
            readout <= '1';
        elsif RO_end_cmd ='1' then
            readout <= '0';
        end if;
    end if;
end process P2;

P3 : process(clock, reset)
begin
    if reset ='1' then
        usb_write_ask <= '0';
        tx_data <= (others => '0');
    elsif rising_edge(clock) then
        if readout ='1' then
            usb_write_ask <= usb_write_ask_RO;
            tx_data <= tx_data_RO;
        else
            usb_write_ask <= usb_write_ask_FSM;
            tx_data <= tx_data_FSM;
        end if;
    end if;
end process P3;
```

USB command list

command	fonction
x01	load and check slow control
x11	load slow control without checking
x02	start acquisition
x21	ramfullext
x22	Trig ext
x03	Digital readout

USB register list

adresse	registre	valeur par defaut	R/W	
0	test	98765432	read	
1	ID_register [31,,0]	BABACAFE	read	
2	test_register [31,,0]	1234ABCD	R/W	
3	control_register [12,,0]	1F3F	R/W	
	bit 0 : reset FPGA	1		active at low level
	bit 1 : resetn : reset hardroc	1		active at low level
	bit 2 : BCID_resetn : BCID reset	1		active at low level
	bit 3 : SC_resetn : slow control reset	1		active at low level
	bit 4 : SR resetn : Shift register reset	1		active at low level
	bit 5 : SC report register resetn	1		active at low level
	bit 8 : Pwr_analog	1		
	bit 9 : pwr_dac	1		
	bit 10 : pwr_ss	1		
	bit 11 : pwr_digital	1		
	bit 12 : pwr_adc	1		
4	status_register [31,,0]	22222222	read	
5	nb HR[7..0]	1	R/W	memoire allouee pour 7 au max pour l'instant
6	Slow control report register [7,,0]		R	
	bit 0 : CRC OK			
	bit 1 : CRC error			
	bit 2 : Load test OK			
	bit 3 : load test error			
	if all SC_load ok : x05			
10	monitoring_register [7,,0]	F9	R/W	
11	temperature_reg [7,,0]		read	
12	dif_current_reg [7,,0]		read	
13	slab_current_reg [7,,0]		read	
14	channel4_monitoring_reg [7,,0]		read	
19	SC debug register [7,,0] (N° of frame t	0	R/W	

Wiki web site

- Firmware available on

https://lyosvn.in2p3.fr/ilc/wiki/VHDL

Or

http://lappweb.in2p3.fr/~prast/GrosFichiers/VHDL_ILC/VHDL_ILC.htm