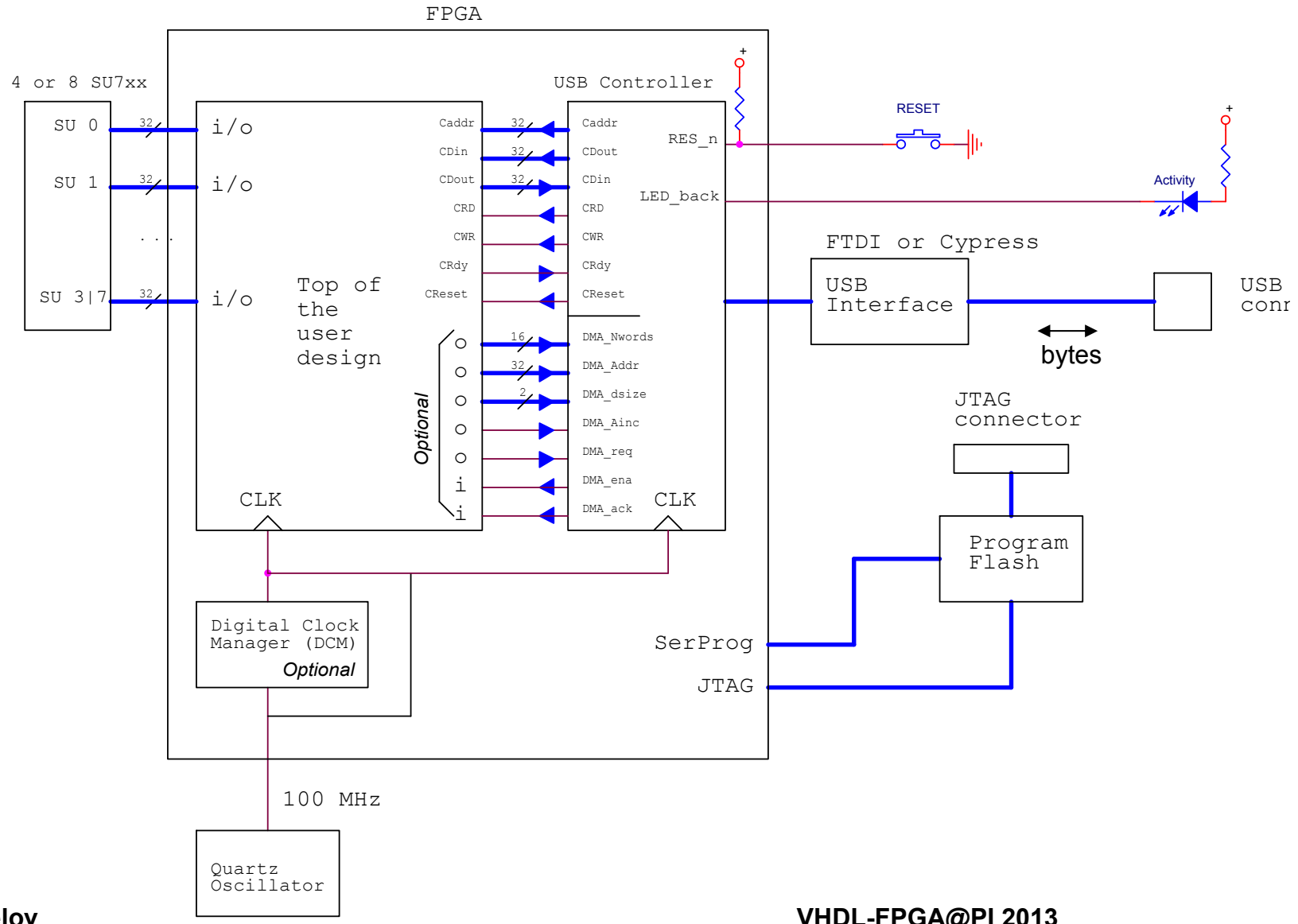
Logic Box

Logic Box

- Block Diagram
- USB Interface
  - Commands
  - Write/Read
  - DMA mode
- User Design
- Top & Contraints Generator
- Design Flow

Block Diagram of DL701/6/9



Commands

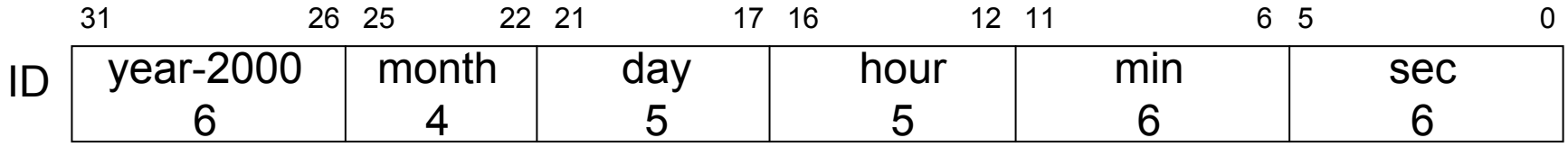
The PC sends and receives bytes over the USB interface. The first byte received by the DL7xx is a command code. Depending on it, additional bytes are expected (or not) and as response some (or no) bytes are sent back to the PC.

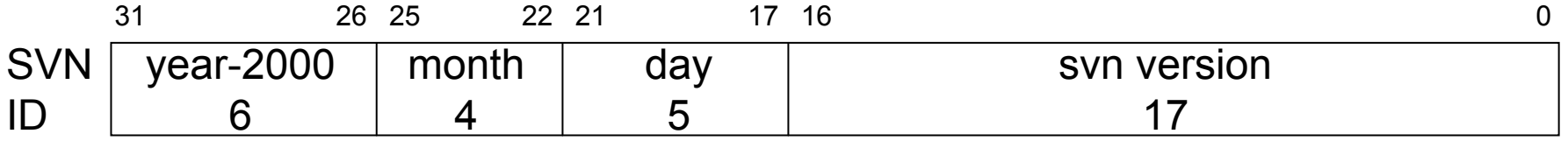
Commands without arguments or return value

ASCII code	Description
R	Activate the CReset
Q	Enable DMA
q	Disable DMA
+	Increment the address
-	Decrement the address

Read commands

ASCII code	Send back	Description
#	4 bytes	Send the ID
V	4 bytes	Send the SVN ID
a	4 bytes	Send the current address
l	4 bytes	Send back read data[31..0]
t	3 bytes	Send back read data[23..0]
w	2 bytes	Send back read data[15..0]
b	1 byte	Send back read data[7..0]





Note: MSByte sent/received first, LSByte last

Write commands

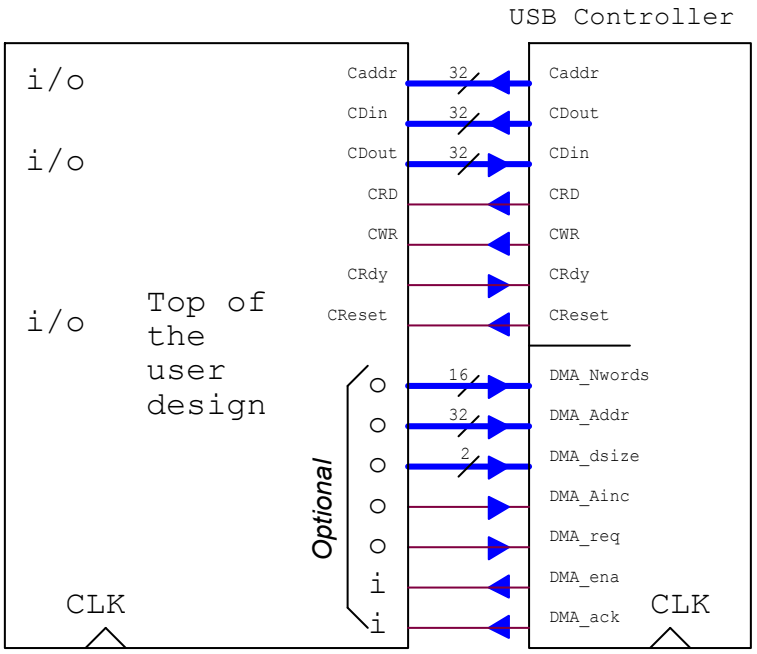
ASCII code	Followed by	Description
A	MSbyte..LSByte	Set the address[31..0]
E	MSbyte..LSByte	Set the address[23..0]
M	MSbyte, LSByte	Set the address[15..0]
S	LSByte	Set the address[7..0]
N	MSByte, LSByte	Set counter for burst RW
F	MSByte, LSByte	Set counter for FIFO burst RW
L	MSByte..LSByte	Write data[31..0]
T	MSByte..LSByte	Write data[23..0]
W	MSByte, LSByte	Write data[15..0]
B	LSByte	Write data[7..0]

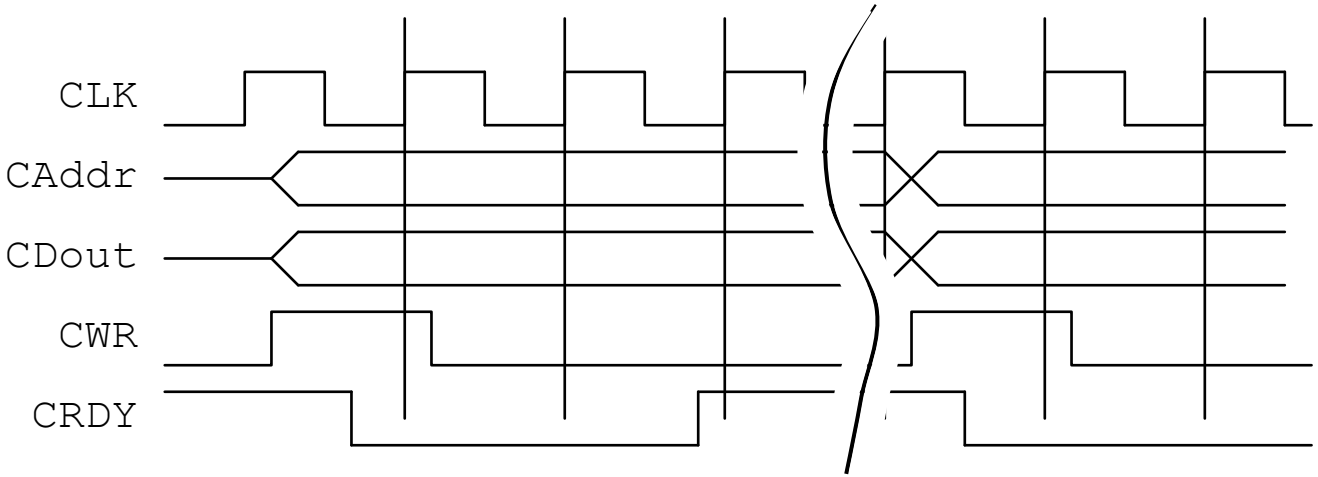
Note: 1) MSByte sent/received first, LSByte last; 2) do not use E, M, S in the software, we plan to remove them; 3) F command doesn't change the address, N increments the address.

Example

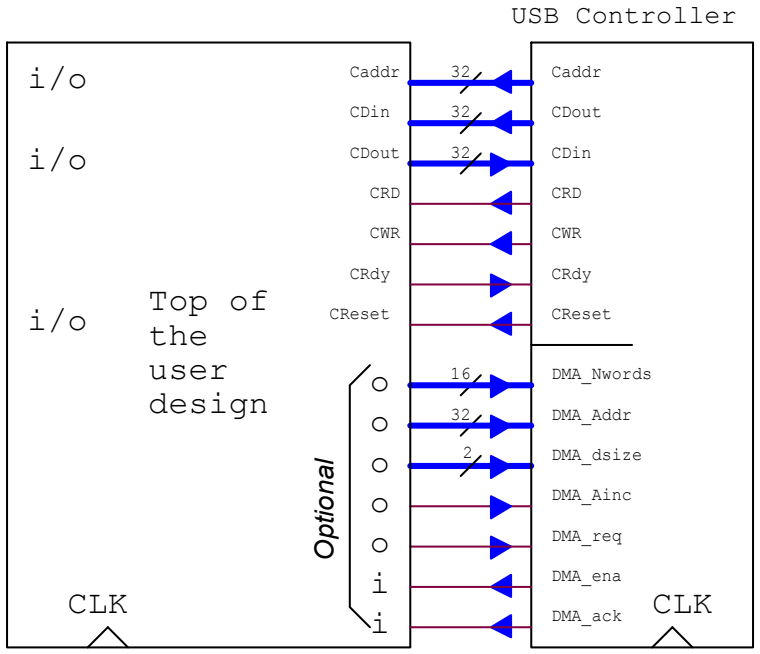
Command ASCII code	PC to DL hex	DL to PC hex	Comment
#	-	0x35329717	2013-04-25 09:28:23
V	-	0x35300127	2013-04-24 SVN NR 295
A	0xdeadaffe	-	CAddr=0xdeadaffe
L	0x04030201	-	Bus[CAddr]=0x04030201
A	0xfaceacad	-	CAddr=0xfaceacad
L	0x08070605	-	Bus[Caddr]=0x08070605
A	0xdeadaffe	-	CAddr=0xdeadaffe
l		0x04030201	read Bus[Caddr]
A	0xdeadaffe	-	CAddr=0xdeadaffe
N	0x004	-	Set address counter=4
B	0x01020304	-	Bus[Caddr++]=0x01
			Bus[Caddr++]=0x02
			Bus[Caddr++]=0x03
			Bus[Caddr++]=0x04
A	0xdeadaffe	-	CAddr=0xdeadaffe
N	0x004	-	Set address counter=4
b		0x01020304	Read from CAddr to CAddr+3

Write Command on the CBus



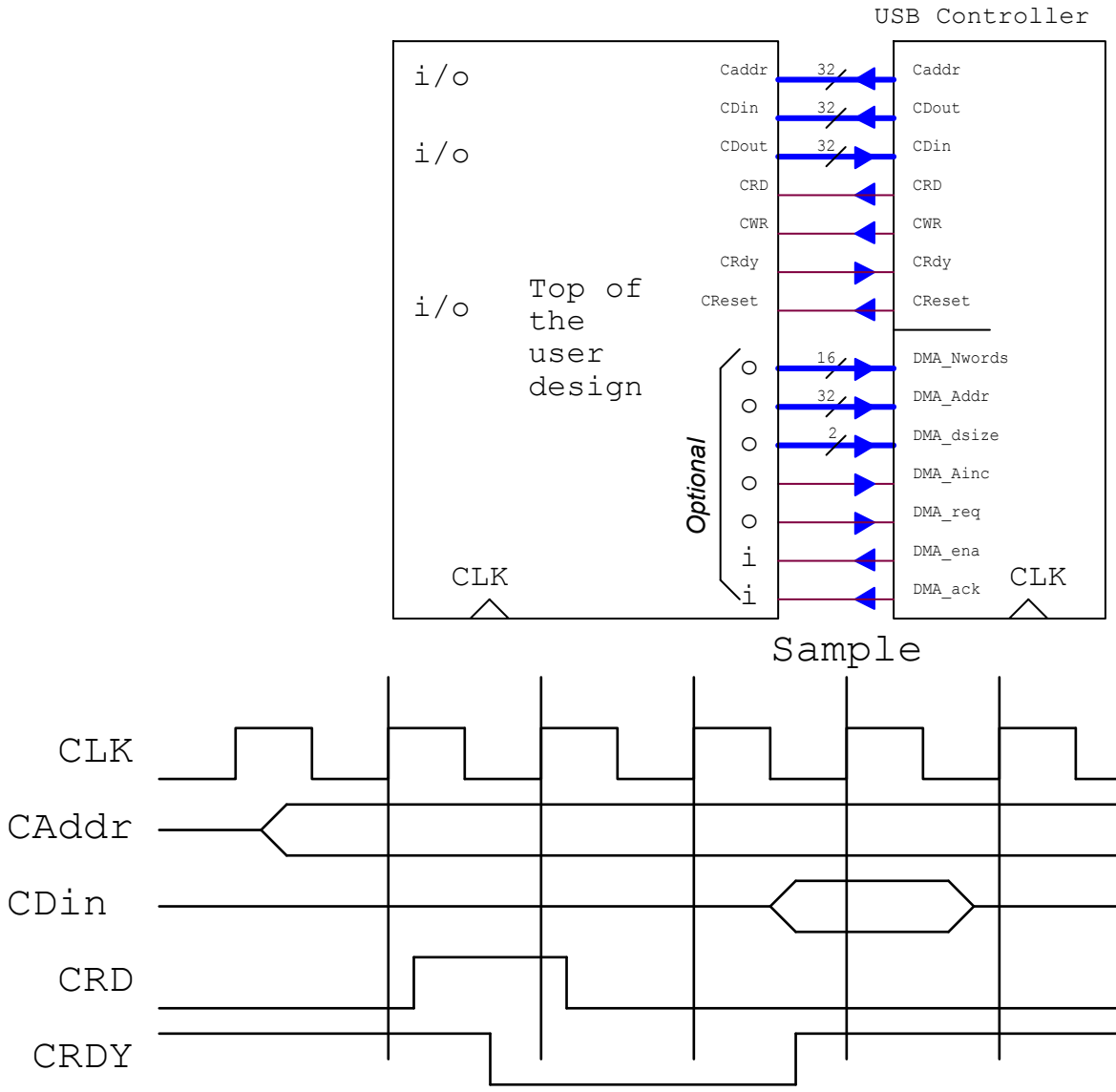


Writing to registers (example)

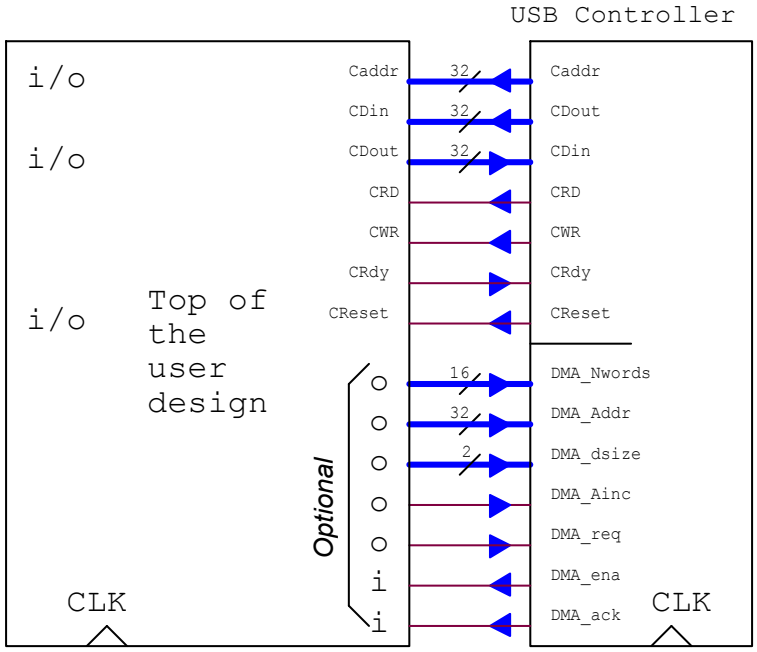



```
...
constant my_address : std_logic_vector(31 downto 4) := X"deadaff";
type my_reg_type is array(0 to 3) of std_logic_vector(31 downto 0);
signal my_reg : my_reg_type;
begin
    -- write
    process(clk)
    begin
        if rising_edge(clk) then
            if Reset='1' then
                for i in my_reg_type'range loop
                    my_reg(i) <= (others => '0');
                end loop;
            elsif CAddr(31 downto 4) = my_address and CWR='1' then
                case CAddr(1 downto 0) is
                  when "00" => my_reg(0) <= CDin;
                  when "01" => my_reg(1) <= CDin;
                  when "10" => my_reg(2) <= CDin;
                  when "11" => my_reg(3) <= CDin;
                  when others => NULL;
                end case;
            end if;
        end if;
    end process;
...
```

Read Command on the CBus



Reading from registers (example)

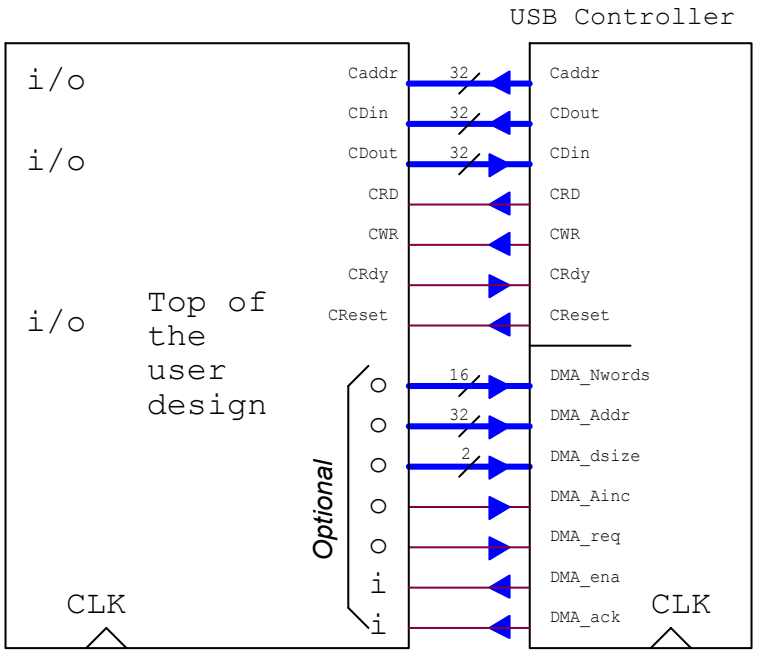



```
signal crd_p  : std_logic_vector( 3 downto 0);
...
    -- read
    process(clk)
    begin
        if rising_edge(clk) then
            crd_p <= crd_p(crd_p'high-1 downto 0) & CRD;
            CDout <= (others => '0');
            if CAddr(31 downto 4) = my_address then
                CDout <= my_reg(conv_integer(CAddr(1 downto 0)));
            end if;
        end if;
    end process;

    CRdy <= not (CRD or crd_p(0) or crd_p(1));
...
```

delay CRdy

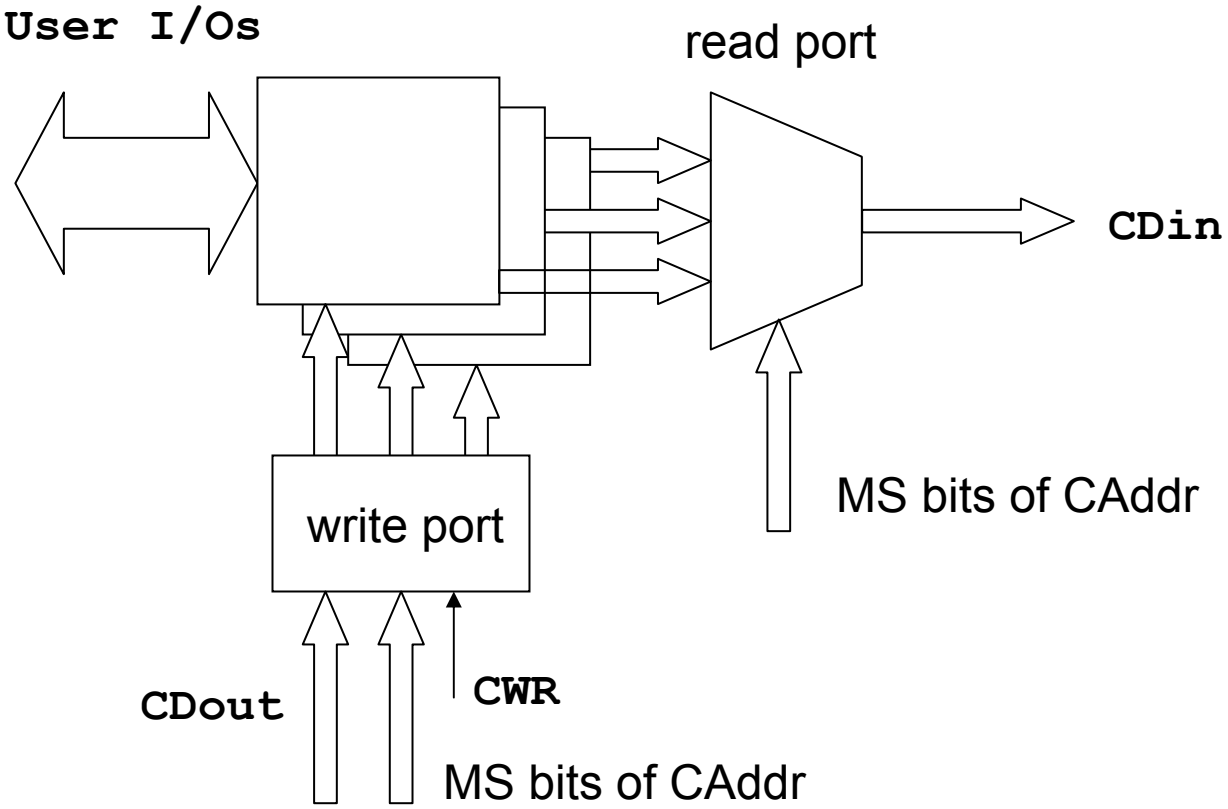
Using DMA (example)

Start a transfer after a falling edge of **`UserIn`**.
The USB controller generates the read cycles and sends the data to the PC.
The DMA should be enabled first.

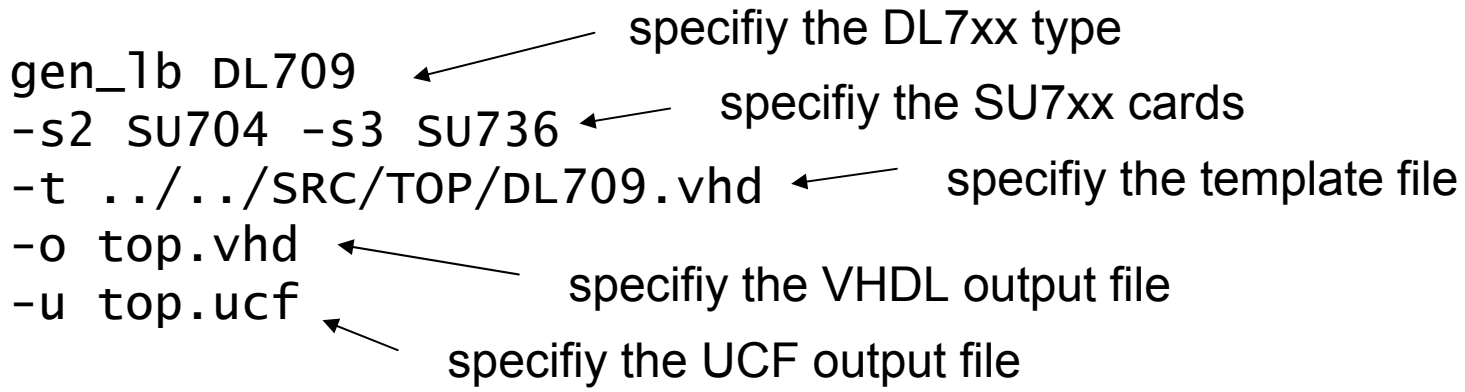



```
...
     UserIn          : in    std_logic;
...

signal userp  : std_logic_vector( 1 downto 0);
...
    -- DMA
    process(clk)
    begin
        if rising_edge(clk) then
            userp <= userp(0) & UserIn;
            -- falling edge of UserIn
            DMA_req <= userp(1) and not userp(0);
        end if;
    end process;

    DMA_Ainc   <= '1'; -- with addr autoincrement
    DMA_Nwords <= X"0004"; -- number of words
    DMA_Addr   <= my_address & "0000"; -- start address
    DMA_dsize  <= "00"; -- byte, 1-16bit,2-24bit,3-32bit
...
```

User Design


User I/Os
read port
CDin
write port
MS bits of CAddr
CDout
CWR
MS bits of CAddr

Generate Top and UCF files



This program supports now:

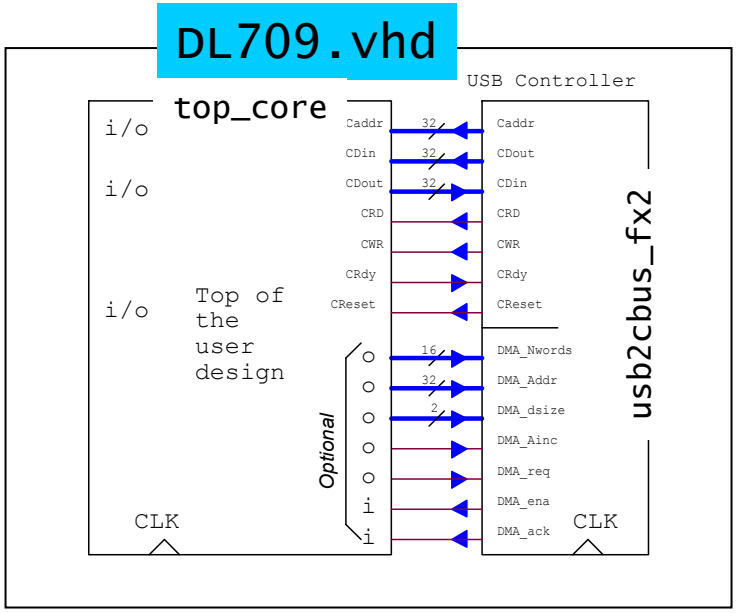
DL701, 706, 709, 710, 711

```
SU701, 702, 703, 704, 706, 707, 709, 710, 711, 712, 713,
       714, 715, 717, 720, 721, 722, 724, 725, 726, 727,
       728, 730, 731, 733, 734, 736, 737
```

Generate **Top** and UCF files

```
library IEEE;
use IEEE.std_LOGIC_1164.ALL;
use work.LogicBox_pkg.all;
entity DL709 is
Port (
    CLK                 : in    std_logic;
    RES_n               : in    std_logic;
    LED_Back            : out   std_logic;
    -- put here the signals to the SUxxx modules
-- Slot 0 is empty
-- Slot 1 is empty
-- SU704 on slot 2
    SU2_INTTL           : in    std_logic_vector( 5 downto  1);
    SU2_OUTP            : out   std_logic_vector( 5 downto  1);
    SU2_INECL           : in    std_logic_vector( 5 downto  1);
    SU2_OE_n            : out   std_logic_vector( 5 downto  1);
    SU2_LED_n           : out   std_logic_vector( 5 downto  1);
    SU2_TERM            : out   std_logic_vector( 5 downto  1);
```

Clock, Reset and LED



SU7xx boards

Generate **Top** and UCF files

```
-- SU736 on slot 3
    SU3_CMP_SE          : in     std_logic_vector( 2 downto  1);
...
    SU3_SDO_DAC         : in     std_logic;
    SU3_SCLK_DAC        : out    std_logic;
    SU3_CS_DACn         : out    std_logic;
...
    -- end of the SUxxx section
    -- FX2
    FXClk               : out    std_logic;
    FXAddr              : out    std_logic_vector( 1 downto  0);
    FXData              : inout  std_logic_vector( 7 downto  0);
    FXRD_n              : out    std_logic;
...
    FXPEnd_n            : out    std_logic);
end DL709;
```

SU7xx boards

USB Interface

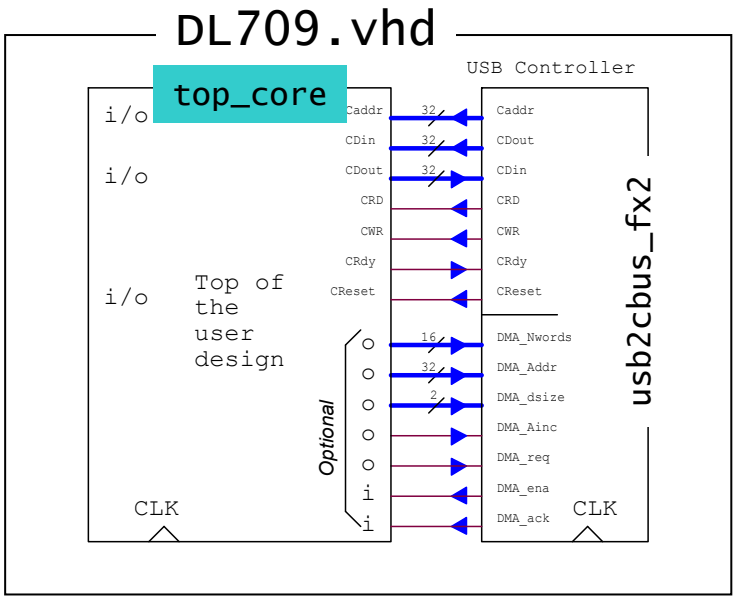
Generate **Top** and UCF files

Component declaration

```
component top_core is
Generic(ClkMHz   : integer := 100);
Port (
    CLK                 : in     std_logic;
    Reset               : in     std_logic;
    -- put here the signals to the SUxxx modules
-- Slot 0 is empty
-- Slot 1 is empty
-- SU704 on slot 2
    SU2_INTTL           : in     std_logic_vector( 5 downto  1);
...
    SU2_TERM            : out    std_logic_vector( 5 downto  1);
-- SU736 on slot 3
    SU3_CMP_SE          : in     std_logic_vector( 2 downto  1);
...
    SU3_LED_NIMn        : out    std_logic_vector( 2 downto  1);
...
-- IF
    CAddr               : in     std_logic_vector(31 downto  0);
    CWR                 : in     std_logic;
    CRD                 : in     std_logic;
    CDIn                : in     std_logic_vector(31 downto  0);
    CDOut               : out    std_logic_vector(31 downto  0);
    CRdy                : out    std_logic);
end component;
```

SU7xx signals

CBus signals



Generate Top and **UCF** files

```
# DL709
NET "CLK"                       LOC = "AF14" | IOSTANDARD = LVTTL ;
NET "LED_Back"                  LOC = "K26"  | IOSTANDARD = LVTTL ;
NET "RES_n"                     LOC = "G26"  | IOSTANDARD = LVTTL | PULLUP ;
NET "FXRD_n"                    LOC = "W26"  | IOSTANDARD = LVTTL ;
NET "FXWR_n"                    LOC = "AC14" | IOSTANDARD = LVTTL ;
NET "FXEmpty"                   LOC = "M26"  | IOSTANDARD = LVTTL ;
NET "FXFull"                    LOC = "L26"  | IOSTANDARD = LVTTL ;
NET "FXClk"                     LOC = "R26"  | IOSTANDARD = LVTTL ;
NET "FXAddr<0>"                 LOC = "Y26"  | IOSTANDARD = LVTTL ;
NET "FXAddr<1>"                 LOC = "AC25" | IOSTANDARD = LVTTL ;
NET "FXSLOE_n"                  LOC = "Y25"  | IOSTANDARD = LVTTL ;
NET "FXPEnd_n"                  LOC = "AC26" | IOSTANDARD = LVTTL ;
NET "FXData<0>"                 LOC = "Y15"  | IOSTANDARD = LVTTL ;
…
NET "FXData<7>"                 LOC = "AA12" | IOSTANDARD = LVTTL ;
# Empty slot 0
# Empty slot 1
# SU704 on slot 2
NET "SU2_INTTL<5>"              LOC = "P1"   | IOSTANDARD = LVTTL ;
NET "SU2_OUTP<5>"               LOC = "P2"   | IOSTANDARD = LVTTL ;
NET "SU2_INECL<5>"              LOC = "P3"   | IOSTANDARD = LVTTL ;
NET "SU2_OE_n<5>"               LOC = "P4"   | IOSTANDARD = LVTTL ;
…
```

Directory tree

```
|\_ C
|\_ SIM
|       \_ USB2CBUS
|                  |\_ SRC
|                   \_ DATA
|\_ SRC
|      |\_ COMMON  (led.vhd, ticker.vhd, LogicBox_pkg.vhd, svn_extract.vhd …)
|      |\_ USB2CBUS (controller_va.vhd, FX2.vhd, FTDI245n.vhd, usb2cbus_fx2.vhd
|       \_ TOP  (Templates DL7xx.vhd)          usb2cbus_ftdi.vhd, VME2CBUS.vhd)
|
 \_ PROJECTS
           |\_ DL709_2xSU704_TDC
           |\_ DL709_7xSU706
           |\_ DL709_8xSU704_MALU
            \_ DL709_SU737_GbEth (Makefile)
                                 |\_ C         (DL709.ucf|.xst, DL709_files.txt, bitgen.ut,
                                 |\_ SCRIPTS             xprog.cmd, xprog_flash.cmd)
                                 |\_ SRC (DL709.vhd …)
                                 |\_ SIM_TOP_CORE   (Makefile)
                                 |               |\_ SRC
                                 |                \_ DATA
                                 |
                                 |\_ REPORTS (cleared before compilation)
                                  \_ other temporary…
```