-------------------------------------------------------------------------

-- File: lec18.vhdl

-- Crtl: 00=Hold 01=Increment 10=Load 11=reset

-------------------------------------------------------------------------

entity lec10 is

generic( N : integer := 4);

port( clk : in STD\_LOGIC;

reset : in STD\_LOGIC;

crtl : in std\_logic\_vector(1 downto 0);

D : in unsigned (N-1 downto 0);

Q : out unsigned (N-1 downto 0));

end lec10;

architecture behavior of lec10 is

signal processQ: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ;

begin

process(clk)

begin

if (rising\_edge(clk)) then

if (reset = '0') then

processQ <= \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ;

elsif (crtl = "01") then

processQ <= \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ;

elsif (crtl = "10") then

processQ <= \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ;

elsif (crtl = "11") then

processQ <= \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ;

end if;

end if;

end process;

Q <= processQ;

end behavior;

------------------------------------------------------------------------------

-- File: user\_logic.vhd

------------------------------------------------------------------------------

use work.lec18Parts.all;

entity user\_logic is

generic (

C\_NUM\_REG : integer := 4;

C\_SLV\_DWIDTH : integer := 32);

port ( LED : out std\_logic\_vector(7 downto 0);

Bus2IP\_Clk : in std\_logic;

Bus2IP\_Resetn : in std\_logic;

Bus2IP\_Data : in std\_logic\_vector(C\_SLV\_DWIDTH-1 downto 0);

… lots of other stuff …);

architecture IMP of user\_logic is

signal \_\_\_ : unsigned (7 downto 0);

begin

counter: \_\_\_\_\_\_\_\_

generic map (\_\_\_\_)

port map( \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_,

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_,

slv\_reg1(1 downto 0),

unsigned(slv\_reg0(7 downto 0)),

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);

LED <= std\_logic\_vector(Q);

SLAVE\_REG\_READ\_PROC : process(slv\_reg\_read\_sel, slv\_reg0, slv\_reg1) is

begin

case slv\_reg\_read\_sel is

when "1000" => slv\_ip2bus\_data <= X"000000" & std\_logic\_vector(Q);

when "0100" => slv\_ip2bus\_data <= slv\_reg1;

when others => slv\_ip2bus\_data <= (others => '0');

end case;

end process SLAVE\_REG\_READ\_PROC;

end IMP;

------------------------------------------------------------------------------

-- counter.vhd Do you need to add: use work.lec18Parts.all;

------------------------------------------------------------------------------

entity counter is

port ( LED : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

S\_AXI\_ACLK : in std\_logic;

S\_AXI\_ARESETN : in std\_logic;

S\_AXI\_WDATA : in std\_logic\_vector(C\_S\_AXI\_DATA\_WIDTH-1 downto 0);

… lots of other stuff …);

architecture IMP of counter is

AXI\_LITE\_IPIF\_I : entity axi\_lite\_ipif\_v1\_01\_a.axi\_lite\_ipif

port map( S\_AXI\_ACLK => S\_AXI\_ACLK,

S\_AXI\_ARESETN => S\_AXI\_ARESETN,

S\_AXI\_WDATA => S\_AXI\_WDATA,

Bus2IP\_Clk => ipif\_Bus2IP\_Clk,

Bus2IP\_Resetn => ipif\_Bus2IP\_Resetn,

Bus2IP\_Data => ipif\_Bus2IP\_Data,

… lots of other stuff …);

USER\_LOGIC\_I : entity counter\_v1\_00\_a.user\_logic

port map (

LED => \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_,

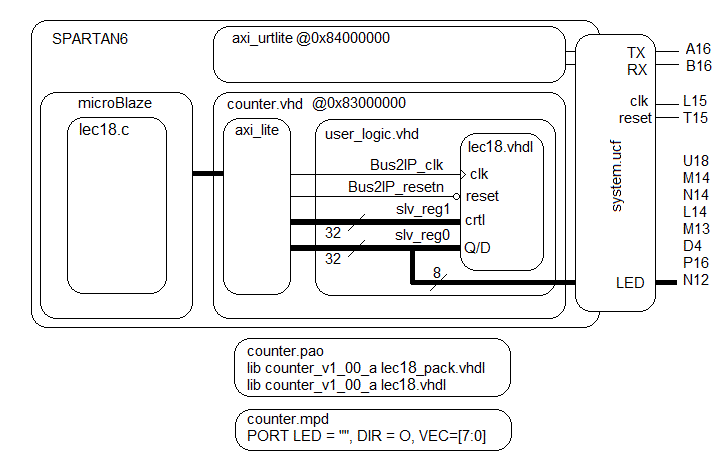
Bus2IP\_Clk => ipif\_Bus2IP\_Clk,

Bus2IP\_Resetn => ipif\_Bus2IP\_Resetn,

Bus2IP\_Data => ipif\_Bus2IP\_Data,

… lots of other stuff …);

end IMP;

****

//--------------------------------------------------------------------

//-- Name: Chris Coulston

//-- Date: Feb 25, 2015

//-- File: lec18.c

//-------------------------------------------------------------------------

#include <xuartlite\_l.h> // Contains XUartLite\_RecvByte

#include <xparameters.h>

#include <xil\_io.h> // Contains Xil\_Out8 and its variations

#include <stdio.h> // Contains xil\_printf

#define uartReadReg 0x84000000 // read <= RX, write=>TX

#define countQReg 0x83000000 // 8 LSBs of slv\_reg0 read=Q, write=D

#define countCrtlReg 0x83000004 // 2 LSBs of slv\_reg2 are control

#define count\_HOLD 0x00 // The control bits are defined in the VHDL

#define count\_COUNT 0x01 // code contained in lec18.vhdl. They are

#define count\_LOAD 0x02 // added here to centralize the bit values in

#define count\_RESET 0x03 // a single place.

int main(void) {

u8 c;

while(1) {

c=XUartLite\_RecvByte(uartReadReg);

switch(c) {

case '?':

xil\_printf("--------------------------\r\n");

xil\_printf(" count Q = %x\r\n",Xil\_In16(countQReg));

xil\_printf("--------------------------\r\n");

xil\_printf("?: help menu\r\n");

xil\_printf("c: COUNTER count up LEDs (by 9)\r\n");

xil\_printf("l: COUNTER load counter\r\n");

xil\_printf("r: COUNTER reset counter\r\n");

break;

//------- Tell the counter to count up – does not quite work as intended

case 'c':

Xil\_Out8(\_\_\_\_\_\_\_\_\_\_\_\_\_\_,\_\_\_\_\_\_\_\_\_\_\_\_\_\_);

Xil\_Out8(\_\_\_\_\_\_\_\_\_\_\_\_\_\_,\_\_\_\_\_\_\_\_\_\_\_\_\_\_);

break;

//------- Tell the counter to load a value

case 'l':

xil\_printf("Enter a 0-9 value to store in the counter: ");

c=XUartLite\_RecvByte(uartReadReg) - 0x30;

Xil\_Out8(countQReg,\_\_\_\_\_\_\_\_\_\_\_\_\_); // put value into slv\_reg1

Xil\_Out8(countCrtlReg,count\_LOAD); // load command

xil\_printf("%c\r\n",c+0x30);

break;

//---------- Reset the counter

case 'r':

Xil\_Out8(countCrtlReg,\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_); // reset command

break;

//------------- unknown character

default:

xil\_printf("unrecognized character: %c\r\n",c);

break;

} // end case

} // end while 1

return 0;

} // end main