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

-- File: lec19.vhdl

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

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

entity lec11 is

generic( N : integer := 4);

 port( clk : in STD\_LOGIC;

 reset\_n : in STD\_LOGIC;

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

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

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

end lec11;

architecture behavior of lec11 is

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

begin

 process(clk)

 begin

 if (rising\_edge(clk)) then

 if (reset\_n = '0') then

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

 elsif (ctrl = "01") then

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

 elsif (ctrl = "10") then

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

 elsif (crtl = "11") then

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

 end if;

 end if;

 end process;

 Q <= processQ;

end behavior;

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

-- File: my\_counter\_ip\_v1\_0\_S00\_AXI.vhd

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

use ieee.numeric\_std.all;

entity my\_counter\_ip\_v1\_0\_S00\_AXI is

 generic (

 -- Width of S\_AXI data bus

 C\_S\_AXI\_DATA\_WIDTH : integer := 32;

 -- Width of S\_AXI address bus

 C\_S\_AXI\_ADDR\_WIDTH : integer := 7

 );

 port (

 -- Users to add ports here

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

 -- User ports ends

 -- Do not modify the ports beyond this line

 -- Global Clock Signal

 S\_AXI\_ACLK : in std\_logic;

 -- Global Reset Signal. This Signal is Active LOW

 S\_AXI\_ARESETN : in std\_logic;

 … lots of other stuff …);

architecture arch\_imp of my\_counter\_ip\_v1\_0\_S00\_AXI is

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

 ---- Signals for user logic register space example

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

 component lec11 is

 generic (N: integer := 4);

 Port( clk: in STD\_LOGIC;

 reset\_n : in STD\_LOGIC;

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

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

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

 end component;

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

begin

 -- Address decoding for reading registers

 loc\_addr := axi\_araddr(ADDR\_LSB + OPT\_MEM\_ADDR\_BITS downto ADDR\_LSB);

 case loc\_addr is

 when b"00000" =>

 reg\_data\_out <= X"000000" & std\_logic\_vector(Q);

 when b"00001" =>

 reg\_data\_out <= slv\_reg1;

 when b"00010" =>

 reg\_data\_out <= slv\_reg2;

 … lots more stuff here

 end case;

 -- Add user logic here

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

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

 port map( clk => \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_,

 reset\_n => \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_,

 ctrl => slv\_reg1(1 downto 0),

 D => unsigned(slv\_reg0(7 downto 0)),

 Q => \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);

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

 -- User logic ends

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

-- my\_counter\_ip\_v1\_0.vhd

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

use ieee.numeric\_std.all;

entity my\_counter\_ip\_v1\_0 is

generic (

 C\_S00\_AXI\_DATA\_WIDTH : integer := 32;

 C\_S00\_AXI\_ADDR\_WIDTH : integer := 7

 );

port (

 -- Users to add ports here

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

 -- User ports ends

 -- Do not modify the ports beyond this line

 -- Ports of Axi Slave Bus Interface S00\_AXI

 s00\_axi\_aclk : in std\_logic;

 s00\_axi\_aresetn : in std\_logic;

 … lots of other stuff …);

architecture arch\_imp of my\_counter\_ip\_v1\_0 is

-- component declaration

component my\_counter\_ip\_v1\_0\_S00\_AXI is

generic (

 C\_S\_AXI\_DATA\_WIDTH : integer := 32;

 C\_S\_AXI\_ADDR\_WIDTH : integer := 7

 );

port (

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

S\_AXI\_ACLK : in std\_logic;-- Instantiation of Axi Bus Interface S00\_AXI

… lots of other stuff …);

my\_counter\_ip\_v1\_0\_S00\_AXI\_inst : my\_counter\_ip\_v1\_0\_S00\_AXI

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

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

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

 … lots of other stuff …);my\_counter\_ip\_v1\_0\_S00\_AXI\_inst : my\_counter\_ip\_v1\_0\_S00\_AXI

 generic map (

 C\_S\_AXI\_DATA\_WIDTH => C\_S00\_AXI\_DATA\_WIDTH,

 C\_S\_AXI\_ADDR\_WIDTH => C\_S00\_AXI\_ADDR\_WIDTH )

 port map (

LED => LED,

S\_AXI\_ACLK => s00\_axi\_aclk,

S\_AXI\_ARESETN => s00\_axi\_aresetn,

 … lots of other stuff …);

****

/\*--------------------------------------------------------------------

-- Name: Prof Jeff Falkinburg

-- Date: Feb 16, 2017

-- File: lec19.c

-- Event: Lecture 19

-- Crs: CSCE 436

--

-- Purp: MicroBlaze Tutorial that implements a custom IP to microBlaze.

--

-- Documentation: MicroBlaze Tutorial

--

-- Academic Integrity Statement: I certify that, while others may have

-- assisted me in brain storming, debugging and validating this program,

-- the program itself is my own work. I understand that submitting code

-- which is the work of other individuals is a violation of the honor

-- code. I also understand that if I knowingly give my original work to

-- another individual is also a violation of the honor code.

-------------------------------------------------------------------------\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Include Files \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**#include** "xparameters.h"

**#include** "stdio.h"

**#include** "xstatus.h"

**#include** "platform.h"

**#include** "xil\_printf.h" // Contains xil\_printf

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

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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Constant Definitions \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

 \* The following constants define the slave registers used for our Counter PCORE

 \*/

**#define** countQReg 0x44a00000 // 8 LSBs of slv\_reg0 read=Q, write=D

**#define** countCtrlReg 0x44a00004 // 2 LSBs of slv\_reg1 are control

**#define** countRollReg 0x44a00008 // 1 LSBs of slv\_reg2 for roll

/\*

 \* The following constants define the Counter commands

 \*/

**#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.

**#define** printf xil\_printf /\* A smaller footprint printf \*/

**#define** uartRegAddr 0x40600000 // read <= RX, write => TX

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Function Prototypes \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Variable Definitions \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

 \* The following are declared globally so they are zeroed and so they are

 \* easily accessible from a debugger

 \*/

**int** **main**()

{

 **unsigned** **char** c;

 init\_platform();

 print("Welcome to Lecture 19\n\r");

 **while**(1) {

 c=XUartLite\_RecvByte(uartRegAddr);

 **switch**(c) {

 /\*-------------------------------------------------

 \* Reply with the help menu

 \*-------------------------------------------------

 \*/

 **case** '?':

 printf("--------------------------\r\n");

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

 printf("--------------------------\r\n");

 printf("?: help menu\r\n");

 printf("o: k\r\n");

 printf("c: COUNTER count up LEDs (by x26)\r\n");

 printf("s: COUNTER start counter\r\n");

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

 printf("r: COUNTER reset counter\r\n");

 printf("f: flush terminal\r\n");

 **break**;

 /\*-------------------------------------------------

 \* Basic I/O loopback

 \*-------------------------------------------------

 \*/

 **case** 'o':

 printf("k \r\n");

 **break**;

 /\*-------------------------------------------------

 \* Tell the counter to count up

 \*-------------------------------------------------

 \*/

 **case** 'c':

 Xil\_Out8(countCtrlReg,count\_COUNT);

 Xil\_Out8(countCtrlReg,count\_HOLD);

 **break**;

 /\*-------------------------------------------------

 \* Start the counter to count up

 \*-------------------------------------------------

 \*/

 **case** 's':

 Xil\_Out8(countCtrlReg,count\_COUNT);

 **break**;

 /\*-------------------------------------------------

 \* Stop the counter from counting

 \*-------------------------------------------------

 \*/

 **case** 'S':

 Xil\_Out8(countCtrlReg,count\_HOLD);

 **break**;

 /\*-------------------------------------------------

 \* Tell the counter to load a value

 \*-------------------------------------------------

 \*/

 **case** 'l':

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

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

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

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

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

 **break**;

 /\*-------------------------------------------------

 \* Reset the counter

 \*-------------------------------------------------

 \*/

 **case** 'r':

 Xil\_Out8(countCtrlReg,count\_RESET); // reset command

 **break**;

 /\*-------------------------------------------------

 \* Clear the terminal window

 \*-------------------------------------------------

 \*/

 **case** 'f':

 **for** (c=0; c<40; c++) printf("\r\n");

 **break**;

 /\*-------------------------------------------------

 \* Unknown character was

 \*-------------------------------------------------

 \*/

 **default**:

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

 **break**;

 } // end case

 }

 cleanup\_platform();

 **return** 0;

} // end main