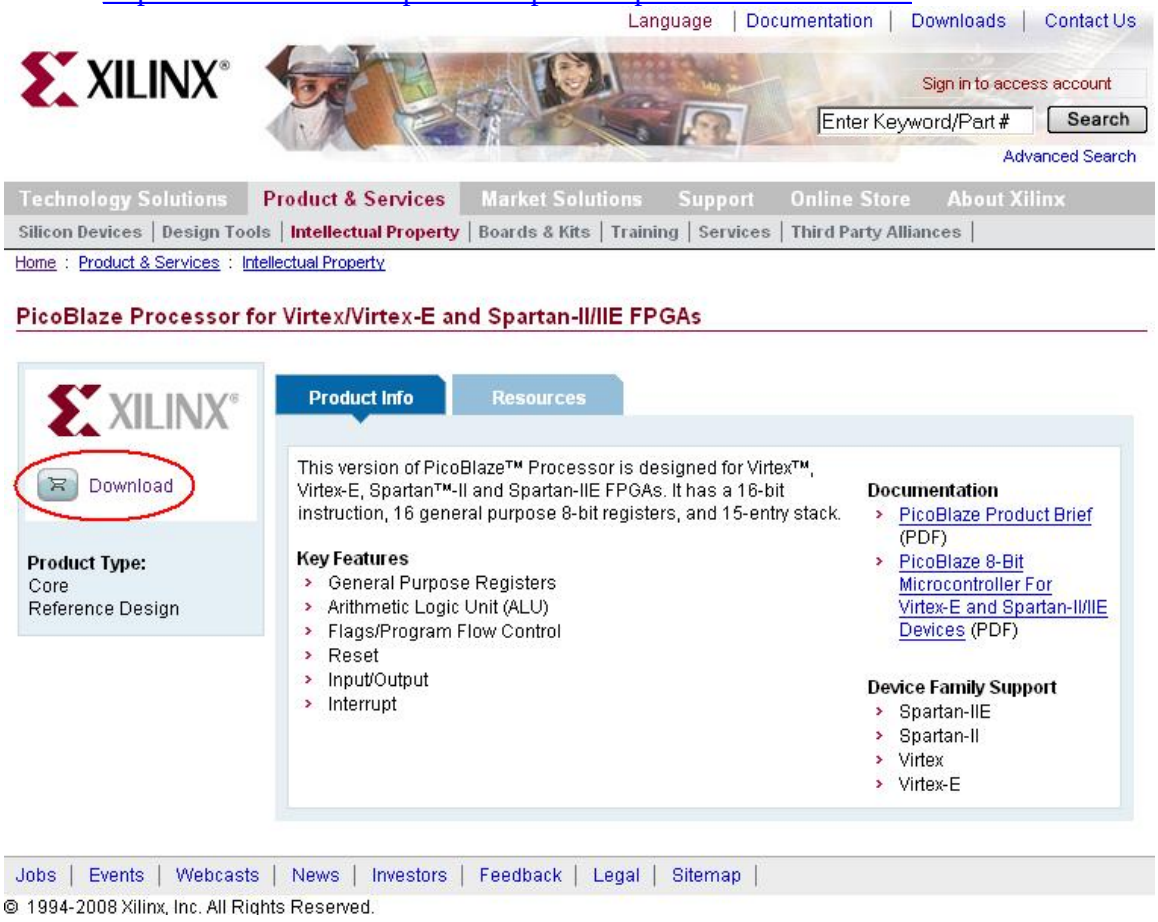
Joe Richardson
Andrew Fitzgerald

Tutorial

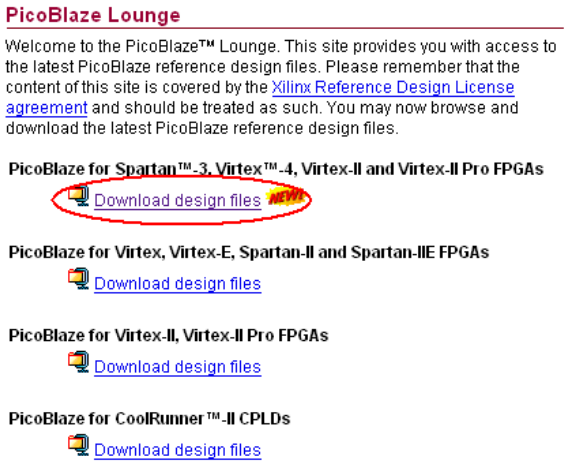
1. Follow the following hyperlink to get the version of picoblaze for the Spartan III FPGA: http://www.xilinx.com/products/ipcenter/picoblaze-VE-S2.htm



*Note: You will need a Xilinx user account. It's free to create and will allow you access to the download area. You will have to fill out a brief survey regarding your usage of picoblaze as well.

2. Proceed to the picoblaze lounge next. It will be a link provided to you after completing the above.

3. Download the resource files for the Spartan III FPGA.



4. Once you've extracted the zip file to some rewritable directory on your account, the file containing the picoblaze processor is KPCSM3.vhd specifically. This file should then be included in your Xilinx project.

5. The next step would be to create your assembly language file that will be converted into a .vhd file to be included in your Xilinx project as well.

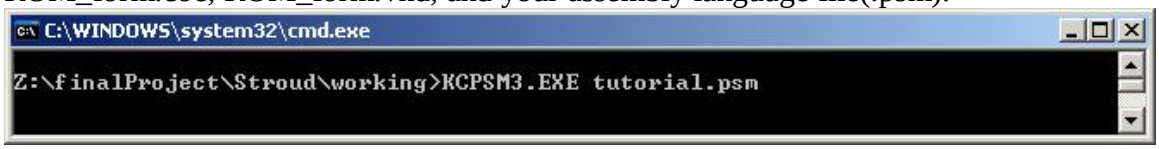
Sample Assembly language file to read from the switches and display on the LEDs.

```
; switches      DSIN   $00
; buttons       DSIN   $01
; LEDS          DSOUT        $02
; segments      DSOUT        $03

INPUT s0,00
OUTPUT s0,02
JUMP 000
```

Note: The comments (; denotes a commented line in assembly) at the beginning of the file should be uncommented when using the picoblaze IDE included with the downloaded files. Please see http://www.xilinx.com/ipcenter/processor_central/picoblaze/picoblaze_user_resources.htm for more information regarding the ISA.

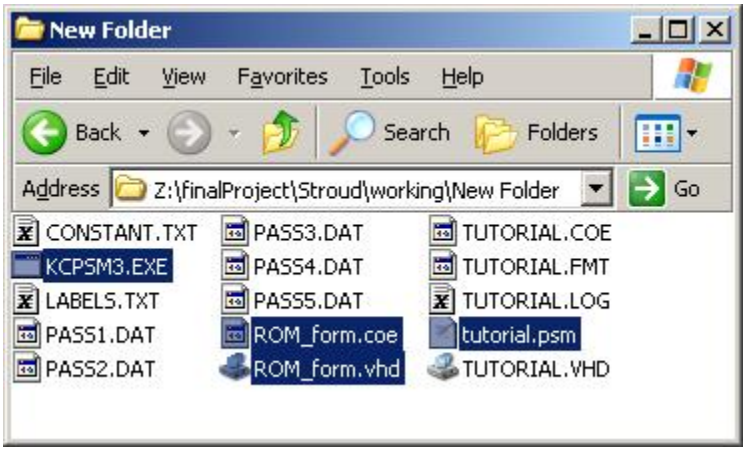
6. The next step is to transform the assembly code into a *.vhd file that can then be added into your Xilinx project to act as an instruction ROM. The KCPSM3.EXE included with the Xilinx picoblaze files is the assembler that performs this step. To run the assembler, navigate to your working directory with the command prompt, then enter %location_of_picoblaze_extraction%/KPCSM3.exe followed by the name of your assembly language file. The location extraction directory should contain KCPSM3.exe, ROM_form.coe, ROM_form.vhd, and your assembly language file(.psm).



*Note: If the temporary directory on your current machine is not writable, you will see the following error:

and your assembly language file WILL NOT be converted. This can be resolved by asking the current lab manager to make the Windows temporary directory writable.

7. After the file completes, the assembler creates many files. These files are shown below and are not highlighted.



The created file, in our case tutorial.vhd, can then be added into your Xilinx project.

8. Within your Xilinx project, you will need a new top level file to connect the picoblaze processor with your newly generated program ROM from your assembly language file. is can be done with simple port maps, as shown below:

```
**********************************************************************
-- Top level VHDL for PICOBLAZE TEST for ELEC4200
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

```
entity picotest is
    Port (       switches : in std_logic_vector(7 downto 0);
                 LEDS : out std_logic_vector(7 downto 0);
                 clk : in std_logic);
    end picotest;
architecture Behavioral of picotest is
-- declaration of KCPSM3 (always use this declaration to call up PicoBlaze
core)
  component kcpsm3
    Port (      address : out std_logic_vector(9 downto 0);
            instruction : in std_logic_vector(17 downto 0);
                port_id : out std_logic_vector(7 downto 0);
           write_strobe : out std_logic;
               out_port : out std_logic_vector(7 downto 0);
            read_strobe : out std_logic;
                in_port : in std_logic_vector(7 downto 0);
              interrupt : in std_logic;
          interrupt_ack : out std_logic;
                  reset : in std_logic;
                    clk : in std_logic);
    end component;
-- declaration of program memory (here you will specify the entity name as your
.psm prefix name)
  component tutorial
    Port (      address : in std_logic_vector(9 downto 0);
            instruction : out std_logic_vector(17 downto 0);
                    clk : in std_logic);
    end component;
-- Signals used to connect PicoBlaze core to program memory and I/O logic
signal address       : std_logic_vector(9 downto 0);
signal instruction   : std_logic_vector(17 downto 0);
signal port_id       : std_logic_vector(7 downto 0);
signal out_port      : std_logic_vector(7 downto 0);
signal in_port       : std_logic_vector(7 downto 0);
signal write_strobe  : std_logic;
signal read_strobe   : std_logic;
signal interrupt_ack : std_logic;
-- the following 2 inputs are assigend inactive values since they are unused in
this example
signal reset         : std_logic :='0';
signal interrupt     : std_logic :='0';
-- Start of circuit description
begin
  -- Instantiating the PicoBlaze core
  processor: kcpsm3
    port map(      address => address,
               instruction => instruction,
                   port_id => port_id,
              write_strobe => write_strobe,
                  out_port => out_port,
               read_strobe => read_strobe,
                   in_port => in_port,
                 interrupt => interrupt,
             interrupt_ack => interrupt_ack,
                     reset => reset,
                       clk => clk);
  -- Instantiating the program memory
  program: tutorial
    port map(      address => address,
               instruction => instruction,
                       clk => clk);

     -- Connect Input to PicoBlaze
```

```
       process( port_id, clk, read_strobe )
       begin
              if clk'event and clk='1' then
                     if read_strobe='1' then
                            case port_id is
                                   when x"00" => in_port <= switches;
                                   when x"01" => in_port <= "00000" & buttons(3
downto 1);
                                   when others => null;
                            end case;
                     end if;
              end if;
       end process;

   -- Connect Output from PicoBlaze
       process( port_id, clk, write_strobe  )
       begin
              if clk'event and clk='1' then
                     if write_strobe='1' then
                            case port_id is
                                   when x"02" => LEDs <= out_port;
                                   when x"03" => ssegVal <= out_port(3 downto 0);
                                   when others => null;
                            end case;
                     end if;
              end if;
       end process;

end Behavioral;
**********************************************************************
```

9. Add in your user constraints to map to your chosen inputs and outputs.

```
**********************************************************************
NET "clk" LOC = "T9";

NET "switches<0>"   LOC = "F12";
NET "switches<1>"   LOC = "G12";
NET "switches<2>"   LOC = "H14";
NET "switches<3>"   LOC = "H13";
NET "switches<4>"   LOC = "J14";
NET "switches<5>"   LOC = "J13";
NET "switches<6>"   LOC = "K14";
NET "switches<7>"   LOC = "K13";

NET "LEDs<0>"       LOC = "K12";
NET "LEDs<1>"       LOC = "P14";
NET "LEDs<2>"       LOC = "L12";
NET "LEDs<3>"       LOC = "N14";
NET "LEDs<4>"       LOC = "P13";
NET "LEDs<5>"       LOC = "N12";
NET "LEDs<6>"       LOC = "P12";
NET "LEDs<7>"       LOC = "P11";
```

```
NET "buttons<0>"     LOC = "M13";
NET "buttons<1>"     LOC = "M14";
NET "buttons<2>"   LOC = "L13";
NET "buttons<3>"   LOC = "L14";

NET "segments<0>"   LOC = "P16";
NET "segments<1>"   LOC = "N16";
NET "segments<2>"   LOC = "F13";
NET "segments<3>"   LOC = "R16";
NET "segments<4>"   LOC = "P15";
NET "segments<5>"   LOC = "N15";
NET "segments<6>"   LOC = "G13";
NET "segments<7>"   LOC = "E14";

NET "anodes<0>"     LOC = "D14";
NET "anodes<1>"     LOC = "G14";
NET "anodes<2>"     LOC = "F14";
NET "anodes<3>"     LOC = "E13";
**************************************************************************
```

10. Generate your programming file and you're done.

Note: You can change address for any of the inputs/outputs by changing the port_id values in the case statements.

Debugging PicoBlaze programs

Our preferred way to debug PicoBlaze programs was to make use of the FPGA LEDs and the seven segment display. So if you needed to see the value of some variable just dump it to the seven segment display. If seven digits wasn't enough we added a windowing feature to our display so one button would scroll forward and one would go backwards. The size of the buffer for the seven segment display is a generic value that can easily be changed to be larger or smaller, depending on the needs of the user. Since our project makes use of states, we decided to use the LEDs to help debug that. So each LED corresponded to a state and the one that was lit was the current state. After we were assured this was working properly, we removed the LED correspondence.

A program called ChipScope is rumored to be able to debug PicoBlaze programs, but there isn't much information out there about that just yet. On a question and answer site some one asked how to debug PicoBlaze programs with ChipScope and here was the answer. "If you dump the address, instruction, and cpu_clock signals into Chipscope and then print out the .log file, it is quite easy to get Chipscope to trigger on any instruction that you want." Here is a link to that page
http://forums.xilinx.com/xlnx/board/message?board.id=PicoBlaze&thread.id=101

Interfacing between PicoBlaze and FPGA hardware

The interfacing between PicoBlaze and FPGA hardware is very simple. The inputs and outputs to PicoBlaze must be mapped to hardware pins. Below is a simple example of this.

Section of PicoBlaze program

```
; switches      DSIN   $00
; buttons       DSIN   $01
; LEDS          DSOUT  $02
; segments      DSOUT  $03

INPUT s0,01
OUTPUT s0,03
JUMP 000
```

Section of VHDL

```
entity picotest is Port (
    switches : in std_logic_vector(7 downto 0);
    LEDS : out std_logic_vector(7 downto 0);
    clk : in std_logic);
end picotest;

-- Connect Input to PicoBlaze
process( port_id, clk, read_strobe )
begin
    if clk'event and clk='1' then
        if read_strobe='1' then
            case port_id is
                when x"00" => in_port <= switches;
                when x"01" => in_port <= "00000" & buttons(3 downto 1);
                when others => null;
            end case;
         end if;
    end if;
end process;

-- Connect Output from PicoBlaze
process( port_id, clk, write_strobe  )
begin
    if clk'event and clk='1' then
        if write_strobe='1' then
            case port_id is
                when x"02" => LEDs <= out_port;
                when x"03" => ssegVal <= out_port(3 downto 0);
                when others => null;
            end case;
        end if;
    end if;
end process;
```

Section of Constraint file

```
NET "switches<0>"   LOC = "F12";
NET "switches<1>"   LOC = "G12";
NET "switches<2>"   LOC = "H14";
```

```
NET "switches<3>"   LOC = "H13";
NET "switches<4>"   LOC = "J14";
NET "switches<5>"   LOC = "J13";
NET "switches<6>"   LOC = "K14";
NET "switches<7>"   LOC = "K13";

NET "LEDs<0>"       LOC = "K12";
NET "LEDs<1>"       LOC = "P14";
NET "LEDs<2>"       LOC = "L12";
NET "LEDs<3>"       LOC = "N14";
NET "LEDs<4>"       LOC = "P13";
NET "LEDs<5>"       LOC = "N12";
NET "LEDs<6>"       LOC = "P12";
NET "LEDs<7>"       LOC = "P11";
```