Introduction to using PicoBlaze microcontroller with the Nexys3 board

This laboratory work is a guide for the PicoBlaze first time users. The objective is to present basic information about the PicoBlaze microcontroller and to describe the basic steps that have to be made in order to implement a very simple project and download it to the Nexys3 board.

1. The PicoBlaze microcontroller

PicoBlaze is a fully embedded 8-bit RISC microcontroller soft core from Xilinx. Its reference design, VHDL source code and code assembler are freely available for Xilinx users. The most recent microcontroller of the PicoBlaze series is KCPSM6 that is optimized for Spartan-6, Virtex-6 and 7-series devices. Fig. 1 depicts the architecture of KCPSM6.

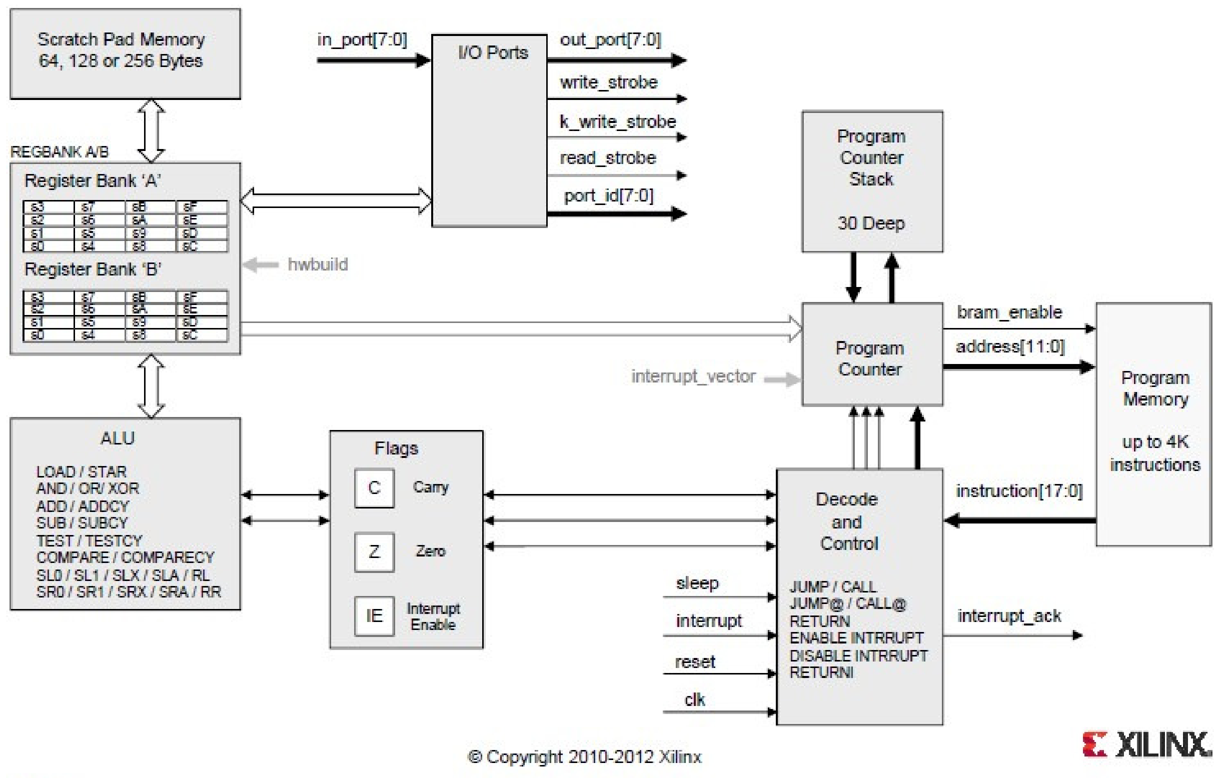

Figure 1. KCPSM6 architecture

The KCPSM6 microcontroller can execute a program of up to 4K instructions. All instructions have an 18 bit format and are executed in 2 clock cycles. The maximum clock frequency depends on the device and the design. The execution performance ranges between 52 and 119 MIPS.

The KCPSM6 provides two banks of 16 general purpose registers, and for larger data sets, it provides access to a scratch pad memory of 64 bytes, 128-bytes or 256-bytes.

The ALU implements a set of logical (AND, OR, XOR) and arithmetic (ADD, SUB) instructions, shift and rotate, TEST and COMPARE instructions. Instruction parameters are values contained in registers or constant values. The results are returned to registers. The program counter is used to fetch instructions from the program memory. In normal conditions, the program counter increments each 2 cycles. JUMP instructions can be used to implement loops and branches. The program counter stack enables the nesting of up to 30 subroutines.

For more information about the KCPSM6, please refer to [1].

To use the KCPSM6 in a design, there are two files/components that have to be included: the actual description of the microcontroller and the program memory (contains the user defined program). Fig. 2 shows how these two components are connected.

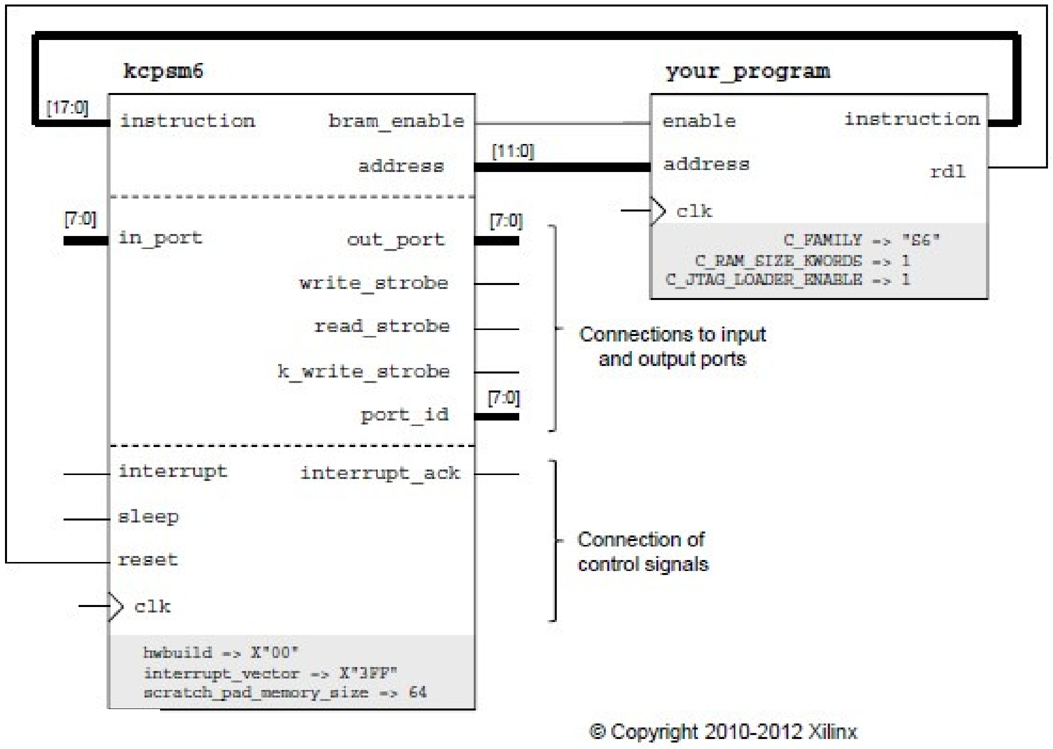

Figure 2. KCPSM6 connection to the program memory

2. Using the PicoBlaze in a design project

The steps needed to implement a basic design project that uses KCPSM6 microcontroller are presented as follows:

Step 1. Get the archive with the KCPSM6 software from Xilinx.
This archive contains VHDL/Verilog source files, the compiler and the user guide.

Step 2. Write a program and compile it to obtain the program memory component.
Write an assembly language program for the KCPSM6 microcontroller and save it as "simple.psm"
To keep it simple use the following code:

```
start: INPUT s0, 00
       OUTPUT s0, 80
       JUMP start
```

This will cause the microcontroller to read data from "port_id=00" and write the same data to "port_id=80".

Create a new folder “YourName”. In this folder, create another folder “simple_design”. Copy the compiler “KCPSM6.exe”, your program “simple.psm” and the “kcpsm6.vhd” file in it. Compile your program with the KCPSM compiler. If the compilation is successful, you will obtain the “simple.vhd” file, which is the program memory for your application.

To compile your program, write to the console:

```
➢ KCPSM simple.psm
```

Step 3. Create a new project.

Start the Xilinx ProjectNavigator and create a new project. Name the new project “simple_design” and save it in “YourName” folder. Set the following project properties:

- Family = Spartan6
- Device = XC6SLX16
- Package = CSG324
- Preferred Language = VHDL

The above parameters can be read on the Nexys3 board.

Click the Finish button.

Step 4. Add kcpsm6 and program memory components.

Click on the Project menu (the top menu) and choose Add source. Select “kcpsm6.vhd” and “simple.vhd” files and add them to the project. Double click on the two files and see how the entities *kcpsm6* and *simple* are defined.

Step 5. Link the microcontroller with the program memory.

To create your design, add a new VHDL module in the project. Name it “top_level”. The inputs for the new entity are the clock (clk) and 8 switches (switches) on the Nexys3 board. The outputs are 8 leds (LEDs) on the Nexys3 board. A led will turn on/off if the corresponding switch is on/off, on the rising edge of the clock.

Include *kcpsm6* and *simple* components in the design as follows:

```
--The PicoBlaze core
component kcpsm6
port (address : out std_logic_vector(11 downto 0);
                       instruction : in std_logic_vector(17 downto 0);
                       bram_enable : out std_logic;
                           in_port : in std_logic_vector(7 downto 0);
                          out_port : out std_logic_vector(7 downto 0);
                           port_id : out std_logic_vector(7 downto 0);
                      write_strobe : out std_logic;
                    k_write_strobe : out std_logic;
                       read_strobe : out std_logic;
                         interrupt : in std_logic;
                     interrupt_ack : out std_logic;
                             sleep : in std_logic;
                             reset : in std_logic;
                               clk : in std_logic);
end component;

--The program memory
component simple
```

```
port (address : in std_logic_vector(11 downto 0);
          instruction : out std_logic_vector(17 downto 0);
               enable : in std_logic;
                  rdl : out std_logic;
                  clk : in std_logic);
end component;
```

Link the two components as in Fig. 2. Set Sleep and Interrupt signals to zero.
Define input and output ports as follows:

```
-- The inputs connect via a pipelined multiplexer
input_ports: process(clk)
begin
if clk'event and clk='1' then
case port_id(1 downto 0) is
 -- read simple toggle switches and buttons at address 00 hex
when "00" =>
 in_port <= switches;
 -- Don't care used for all other addresses to ensure minimum
 -- logic implementation
when others =>
 in_port <= "XXXXXXXX";
end case;
 end if;
end process input_ports;

-- adding the output registers to the processor at address 80 hex
output_ports: process(clk)
begin
 if clk'event and clk='1' then
 if port_id(7)='1' then
 LED S <= out_port;
 end if;
end if;
end process output_ports;
```

Save the file after you finish writing the code.

Step.6. Check the syntax.
Double click on the “Synthesize-XST > Check Syntax” option. If there are any errors, fix them and re-check the syntax.

Step7. Assign board pins to signals.
The *top_level* entity has the following code:

```
entity top_level is
Port ( switches : in STD_LOGIC_VECTOR (7 downto 0);
clk : in STD_LOGIC;
LEDs : out STD_LOGIC_VECTOR (7 downto 0));
end top_level;
```

The input and output signals have to be assigned to the clock, switches and leds on the Nexys3 board. For this, expand the User Constraints (Design section in the right of the Project Navigator), then double click on "I/O Pin Planning (PlanAhead) - Pre-Synthesis". You will be asked if you want to create the.UCF file. Click OK. In the PlanAhead window, complete the Site column as follows:

Name	**Site**
clk	V10
LEDs[7]	T11
LEDs[6]	R11
LEDs[5]	N11
LEDs[4]	M11
LEDs[3]	V15
LEDs[2]	U15
LEDs[1]	V16
LEDs[0]	U16
switches[7]	T5
switches[6]	V8
switches[5]	U8
switches[4]	N8
switches[3]	M8
switches[2]	V9
switches[1]	T9
switches[0]	T10

For switches and leds set the I/O Std to LVTTL, for the clock set it to LVCMOS33.
Save you changes and close the PlanAhead window.

Step 8. Implement design.
Double click on Implement Design (Design section in the right of the Project Navigator). This will cause Synthesize-XST to run first and next the stages of Implement Design. If each stage is completes with success, a green tick will appear for it. If there are any errors or warnings, resolve them and redo this step.

Step 9. Download the program on the board.
First double click on Generate Programming File. If this stage succeeds, connect the board and turn it on, then double click on Manage Configuration Project (iMPACT).
In the iMPACT window, click on Boundary Scan (on the right). Right click on the text "Right click to Add Device or Initialize JTAG Chain", and select Initialize Chain. Click YES on the dialog box that asks you if you wish to select a configuration file. Choose the top-level.bit file from the open file dialog. Click OK on the Device Programming Properties dialog. Then right click on the device and choose Program.

Step 10. Test the program on the board.
Change the switches states and see if the leds respond to these changes.

3. Exercise

Use the PicoBlaze and Nexys3 to add two 4-bit numbers. Use the switches to enter the values and the 7 segment display to display the result.

Bibliography

[1] K. Chapman, “PicoBlaze for Spartan-6, Virtex-6 and 7-Series (KCPSM6)”
[2] Digilent, “Nexys3 Board Reference Manual”
[3] J. Banks, “The Spartan-3E Tutorial 2: Introduction to using the PicoBlaze Microcontroller”