ELEC6016 Digital System Design

SystemVerilog Design of an Embedded Processor

Introduction

This exercise is done individually and the assessment is:

- By formal report describing the final design, its development, implementation and testing.
- By a laboratory demonstration of the final design on an Altera FPGA development system

Specification

The objective of this exercise is to design an n-bit implementation of picoMIPS.

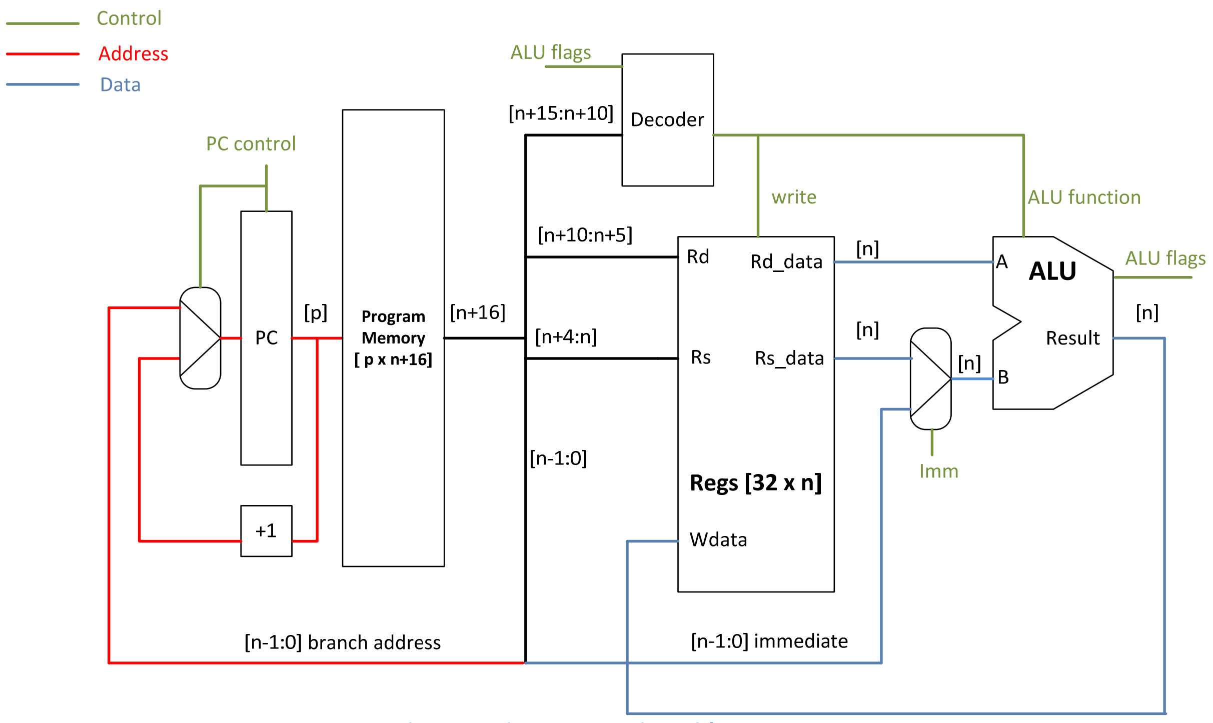

Figure 1. picoMIPS, version without RAM.

The design should be as small as possible in terms of FPGA resources but sufficient to implement the affine transformation algorithm described below. The size cost function of the design is defined as follows:

Cost = number of Logic Elements used + 30 x Kbits of RAM used

Each Logic Element has a flip-flop hence flip-flops are included in the above cost figure. The cost figure should be calculated for Altera Cyclone IV EP4CE115 and should be as low as possible. Altera Cyclone IV has 266 18x18 bit embedded hardware multipliers. If embedded multipliers are used in your implementation, they are 'free', i.e. they do not contribute to the size cost. To demonstrate the cost figure of your design show in your report Altera Quartus synthesis statistics for Cyclone IV EP4CE115.

ELEC6016 lecture slides describe the picoMIPS architecture in detail and provide SystemVerilog coding suggestions for the picoMIPS blocks. An additional functionality to input 8-bit data and to output 8-bit results will be required as described below. You may design your own instruction set and modify the instruction format in any way you wish. You may also modify the architecture if it helps to reduce the cost figure.

Affine transformation algorithm

An affine transformation is a geometrical transformation that preserves collinearity, i.e. all points lying on a line will also lie on a line after the transformation and distance ratios are preserved. For two-dimensional images, the general affine transformation can be expressed as:

$$\begin{vmatrix} x_2 \\ y_2 \end{vmatrix} = A \times \begin{vmatrix} x_1 \\ y_1 \end{vmatrix} + B$$

Where [x1,y1] are the coordinates of a pixel before the transformation and [x2,y2] – after the transformation. The 2x2 matrix **A** and the two-element vector **B** provide the transformation coefficients. For example, a pure translation of pixels occurs if:

$$A = \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix}, B = \begin{vmatrix} b_1 \\ b_2 \end{vmatrix}$$

Similarly, the following coefficients implement pure scaling:

$$A = \begin{vmatrix} a_{11} & 0 \\ 0 & a_{22} \end{vmatrix}; B = \begin{vmatrix} 0 \\ 0 \end{vmatrix}$$

In practice different affine transformations are combined to produce a complex transformation.

Implementation of affine transformation in picoMIPS

You are required to develop both a smallest possible picoMIPS architecture and a machine-level program for the general affine transformation. The 6 constants that define the transformation must be included as immediate literals in your program. Several sets of transformation constants will be provided separately for you to choose from. Pixel coordinates must be read from the switches SW0-SW7 on the FPGA development system and the resulting pixel coordinates after the transformation displayed on the LEDS LED0-LED7. Switch SW8 provides handshaking functionality as described in the pseudocode below. Switch SW9 should act as an active low reset.

Pseudocode

```
1. Wait for coordinate x1 by polling switch SW8. Wait while SW8=0. When SW8
   becomes 1 (SW8=1) read the coordinate x1 from SW0-SW7.
2. Wait for switch SW8 to become 0
3. Wait for coordinate y1 as specified in step 1.
4. Wait for SW8 to become 0
5. Execute the affine transformation algorithm and display coordinate x2 on LED0-
   LED7.
6. Wait until SW8 becomes 1
7. Display coordinate y2 on LED0-LED7.
8. Wait until SW8 becomes 0
9. Go to step 1.
```

Design strategy

Develop SystemVerilog code and a separate testbench for each module in your design. Simulate each module in Modelsim. Synthesise each module in Altera Quartus and carefully analyse the synthesis warnings, statistics and RTL diagrams. When you are satisfied that all your picoMIPS modules are correct, write a testbench for the whole design and simulate. Synthesise the whole design and again, carefully analyse the warnings, statistics and RTL diagrams. You will be able to take an FPGA Development System on loan for a few days in Week 6 or Week 7. Test your design either at home or in the laboratory. In Week 8 or 9 (after the Easter Break) you will be asked to demonstrate your design.

The input/output functionality can be implemented in several different ways. For example, you can design your own IN and OUT instructions for reading/writing data using external ports. To use fewer hardware resources, you can consider connecting the ALU output, or a register output directly to the LEDs. You could consider dedicating a specific register number, e.g. register 1 to the input port. In this way, the ADD instruction can be used to read data, e.g. ADD %5, %0,%1 would move the input data to register %5. Be creative and use your imagination!

Formal report

Submit an electronic copy of your report through C-BASS, the electronic handin system, and a printed copy to the ECS front office. The report should not exceed 3000 words in length. It must contain a full discussion of your design, including the final circuit diagrams, your instruction set and your program. A Word template will be provided with suggested structure of the report. source files must be packaged in a zip file and submitted electronically as a separate file at the same time. The reports will normally be marked and returned to you before the end of term. , so that any feedback s can be incorporated into later reports. In this exercise, 40% of the marks are allocated to the report, its style and organisation, with the remaining 60% for the technical content. As always, bonus marks are awarded for implementation of novel concepts.

tjk, 6 Feb’13, rev. 17 Apr’13, 4 Feb’14