Soft Multiprocessor on FPGA

Sun Wei

1. Introduction

Soft Multiprocessor on FPGA is becoming more attractive as the design cost and NRE soaring up in deep-submicron age. However, it becomes time consuming and error prone to design multiprocessor when the number of processors grows quickly. To make it easier, I designed a tool, BlazeCluster, to generate multiprocessor architecture on FPGA consisting of Xilinx microblaze and PowerPC cores from a simple, top-level script.

The tool is written in Perl. On most of Linux installations, the Perl interpreter is already there. For Windows XP you can install activePerl. The generated MHS, MSS and UCF file can be synthesized by Xilinx EDK7.1 and ISE7.1. The simulator is ModelSim 6.1 starter version.

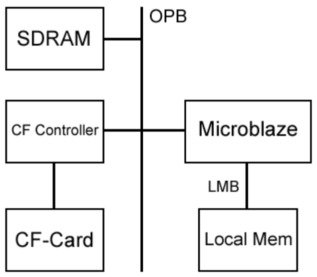
It can also be used as a fast prototype tool for computing intensive applications on FPGA. Followings are two cases studies to show the usage of BlazeCluster.

2. Case Study I: Use BlazeCluster to Design and Optimize JPEG Encoder on FPGA

BlazeCluster is a handy tool in designing and optimizing multiprocessor on FPGA because you can avoid writing detailed description for MHS files. The grammar is similar to natural language and can be easily understood from a few examples. Here is a JPEG example to show how it works. The reference code is from http://www.opencores.org/projects.cgi/web/mb-jpeg/overview by Sun Wei, Joris van Emden and Marcel Lauwerijssen.

Step1: One-microblaze system.

The one-processor architecture can be generated by BlazeCluster with the following description.



microblaze_0, microblaze, opb-master, 64k on-chip ram, 256m on-board opb-ddr sdram address 0x30000000, jtag, barrel-shifter, opb-uartlite baudrate 9600 stdio, opb-cf-card readwrite

Save the description into a file called *system.js*, put it into the same directory as BlazeCluster scripts and run *jena.pl*. If it runs successfully, there will be system.MHS, system.MSS and system.UCF created.

Currently BlazeCluster can't generate all EDK project files. You can generate an empty project by EDK Base System Builder Wizard. Then close EDK and replace system.MHS, system.MSS, data/system.UCF with generated file and open EDK again. Now you get the architecture you want.

The reference code is already based on microblaze processor. You can also use BlazeCluster to generate architecture to replace the existing one. Compared to reference code, I also added a barrel shifter onto the processor and did some code cleanup. The subsampling is removed to make the code easy.

The system can be profiled by Xilinx EDK tools. The following is the profiling result for major functions. It's measured based on a 1600x1200 24bit BMP file input. Please note that it's intrusive profiling so the result may be not very accurate.

Function	**Iterations**	**Time (s)**	**PS**
dct()	90000	2.26	DCT
RGB2YUV_matrix()	30000	1.49	Color Conversion
zzq_encode()	30000	1.29	VLC
EncodeDataUnit()	30000	0.79	VLC
get_MB()	30000	1.24	Preparation

Step 2 : Two-Microblaze System

From the previous profiling result, it's clear that dct() take most of time. It's logical to take it out and put on another processor. Two-processor architecture can be generated from following:

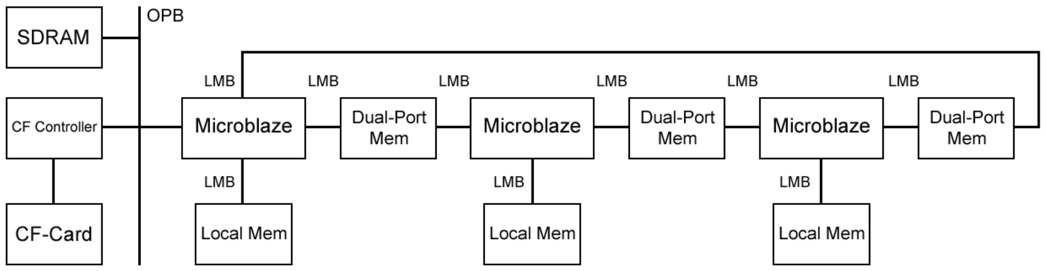
microblaze_0, microblaze, opb-master, 64k on-chip ram, jtag, 256m on-board opb-ddr sdram address 0x30000000, opb-uartlite baudrate 9600 stdio, opb-cf-card readwrite
microblaze_1, microblaze, 8k on-chip ram
dp01, dpram, 8k, left microblaze_0 address 0x20000000, right microblaze_1 address 0x20000000

dp01 is the communication channel between two processors. It's 8kbyte on-chip dual port memory. The left side is connected to microblaze_0 and the right side is connected to microblaze_1 for dct().

For efficient communication, the result of color conversion can be put onto dual port memory directly and the same to the result of dct. It avoids data copy. Link buffer list can be used to avoid waiting time. Finally the profiling result shows that the execution time is almost the previous one minus dct time. The communication overhead is little.

Step 3: Three-Microblaze System

VLC is also a time consuming part. It can be unloaded with one more processor. The description file is as following.

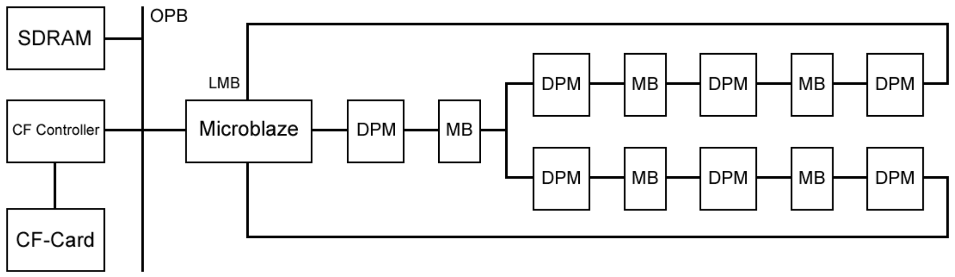


microblaze_0, microblaze, opb-master, 64k on-chip ram, jtag, 256m on-board opb-ddr sdram address 0x30000000, opb-uartlite baudrate 9600 stdio, opb-cf-card readwrite
microblaze_1, microblaze, 8k on-chip ram
microblaze_2, microblaze, 8k on-chip ram
dp01, dpram, 8k, left microblaze_0 address 0x20000000, right microblaze_1 address 0x20000000
dp12, dpram, 8k, left microblaze_1 address 0x21000000, right microblaze_2 address 0x21000000
dp20, dpram, 8k, left microblaze_2 address 0x22000000, right microblaze_0 address 0x22000000

There are two more communication channels. One is between dct processor and vlc processor. The other one is between vlc processor and master processor. The profiling result shows that vlc part is unloaded from the master processor with little communication overhead. Compared to original one-microblaze system we already get 2.5x performance gain. The reason that we didn't get 3x gain is because the load is not balanced between different processors.

Step 4: Six-Microblaze System

There is still plenty of space left on an XC2VP30 chip. Why not duplicate last design and get double speed. The description file is as following.



Simplified diagram (MB-Microblaze, DPM-Dual Port Mem, Unmarked bus are all LMB bus)

microblaze_m, microblaze, opb-master, 64k on-chip ram, jtag, 256m on-board opb-ddr sdram address 0x30000000, opb-uartlite baudrate 9600 stdio, opb-cf-card readwrite
microblaze_cc, microblaze, 8k on-chip ram
microblaze_dct0, microblaze, 8k on-chip ram
microblaze_dct1, microblaze, 8k on-chip ram
microblaze_vlc0, microblaze, 8k on-chip ram
microblaze_vlc1, microblaze, 8k on-chip ram
dp_m_cc, dpram, 8k, left microblaze_m address 0x20000000, right microblaze_cc
dp_cc_dct0, dpram, 8k, left microblaze_cc address 0x21000000, right microblaze_dct0
dp_dct0_vlc0, dpram, 8k, left microblaze_dct0 address 0x22000000, right microblaze_vlc0
dp_vlc0_m, dpram, 8k, left microblaze_vlc0 address 0x23000000, right microblaze_m
dp_cc_dct1, dpram, 8k, left microblaze_cc address 0x24000000, right microblaze_dct1
dp_dct1_vlc1, dpram, 8k, left microblaze_dct1 address 0x25000000, right microblaze_vlc1
dp_vlc1_m, dpram, 8k, left microblaze_vlc1 address 0x26000000, right microblaze_m

However, these six processors are not arranged as two instances of last design. Because the preparation and color conversion time consuming is around half of time consuming for dct and vlc, I use one processor for preparation of two channels and one processor for color conversion for two channels. For dct and vlc, each channel has its own instance.

Profiling result shows that it takes the same time as last design while it encode two 1600x1200 BMP pictures simultaneously.

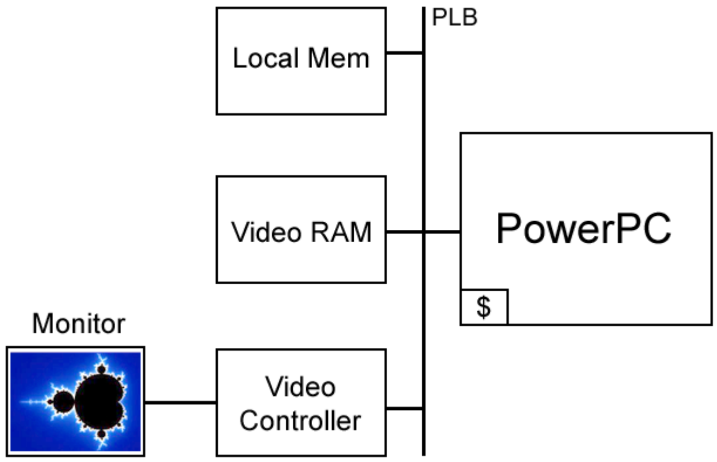
3. Cast Study II: Use BlazeCluster to Design and Optimize Mandelbrot Set Generator on FPGA

Mandelbrot set is a set of points in the complex plane that forms a fractal. It's popular outside of mathematics because of its aesthetic appeal. It's also a typical parallel computing application implemented on a variety of platforms. Here I try to implement it on soft multiprocessor on FPGA.

The reference code is Mandelbrot Generator project by antikons.and QuickMAN project by Paul Gentieu. Both of them are on sf.net. For FPGA part, the reference code is slideshow example from Xilinx.

Step 1: One PowerPC system

This is the baseline of the design. I simply combine the reference code together. It can show a Mandelbrot Set image via VGA output. There is no FPU on PowerPC so the floating-point operation is emulated. The description file is as following.



ppc0, powerpc, 100mhz, 32k on-chip ram, jtag, opb-uartlite baudrate 9600 stdio, plb-dcr-vgacontroller, 256m on-board plb-ddr sdram address 0x30000000

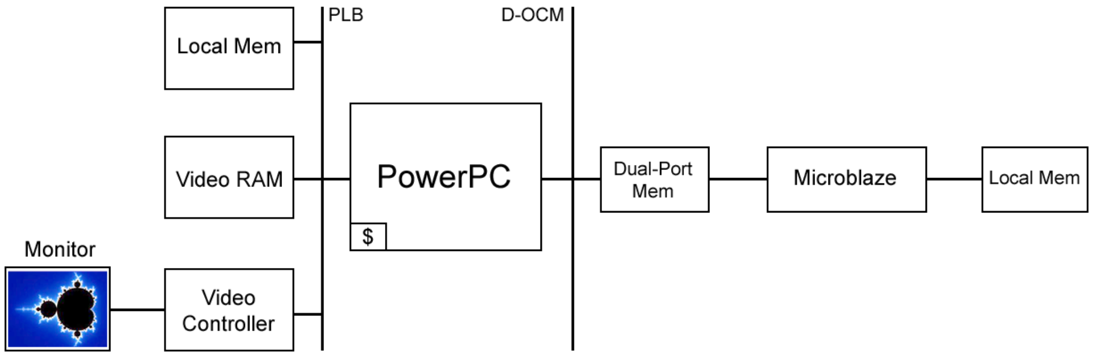
With this script, BlazeBluster instantiate a PowerPC core with 32kbyte on-chip memory with PLB bus, a PLB-OPB bridge, a UART controller with OPB bus, an OPB-DCR bridge, a VGA controller with PLB bus and DCR bus and a DDR SDRAM controller with PLB bus. The clock controller for processor and SDRAM is also instantiated.

PowerPC core does Mandelbrot set computation. The result is mapped to a palette and painted in the video buffer, which is on-board SDRAM. The video controller read from the video buffer to refresh the display.

It takes a while to paint a single figure. It’s mainly due to lack of FPU on PowerPC. Xilinx offers FPU for PowerPC but it requires additional cost.

Step 2: One PowerPC plus One Microblaze with FPU system

The design can be accelerated by adding a Microblaze with FPU. The communication between two processors is still dual port memory because it's efficient for bulk of data transfer. The description is as following:



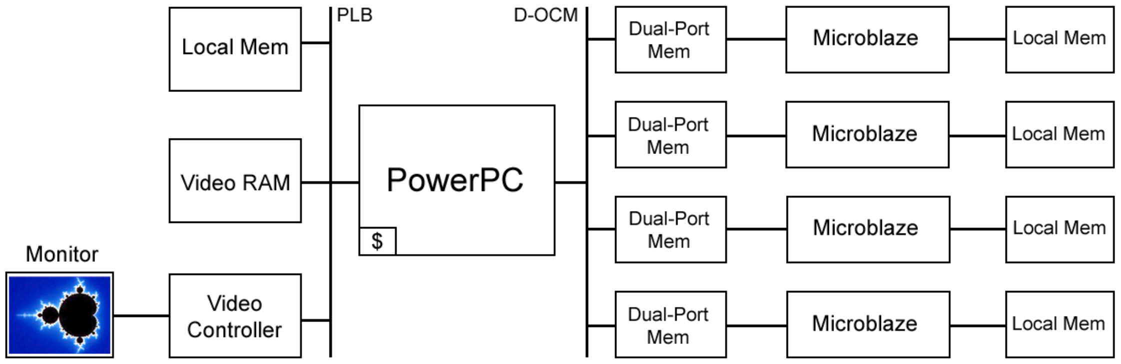
ppc0, powerpc, 100mhz , 32k on-chip ram, jtag, opb-uartlite baudrate 9600 stdio, plb-dcr-vgacontroller, 256m on-board plb-ddr sdram address 0x30000000
mb0, microblaze, 8k on-chip ram, barrel-shifter, fpu
dp_ppc0_mb0, dpram, 8k, left ppc0 address 0x20000000, right mb0

It's similar to the script in the previous case. However, the bus controller for dual port memory on PowerPC side is DOCM bus. You can't use PLB bus or LMB bus in this case. The bus controller on Microblaze side is still LMB bus. You don't need to specify them as BlazeCluster instantiate them automatically.

There is a significant improvement in speed due to FPU on Microblaze.

Step 3: One PowerPC plus Four Microblaze with FPU system

To achieve higher performance, more co-processor will help. I finally add four microblaze with FPU. The result is almost 4X faster. The description is as following.



ppc0, powerpc, 32k on-chip ram, jtag, opb-uartlite baudrate 9600 stdio, plb-dcr-vga-controller, 256m on-board plb-ddr sdram address 0x30000000
mb0, microblaze, 8k on-chip ram, barrel-shifter, fpu
mb1, microblaze, 8k on-chip ram, barrel-shifter, fpu
mb2, microblaze, 8k on-chip ram, barrel-shifter, fpu
mb3, microblaze, 8k on-chip ram, barrel-shifter, fpu
dp_ppc0_mb0, dpram, 8k, left ppc0 address 0x20000000, right mb0
dp_ppc0_mb1, dpram, 8k, left ppc0 address 0x20100000, right mb1
dp_ppc0_mb2, dpram, 8k, left ppc0 address 0x20200000, right mb2
dp_ppc0_mb3, dpram, 8k, left ppc0 address 0x20300000, right mb3

4. For Further Study

There are still lots of area left on 2VP30P FPGA and lots of interesting combination can be explored. For instance, special hardware accelerator can be added because most processors do a specific job and the topology need to update to accommodate this change.

There is some problem in profiling code on PowerPC. That's the reason that there is no quantitative result for the second case study. However it is worthy to explore on.

For more information:
Web (BlazeCluster) http://www.opencores.org/projects.cgi/web/mpdma/overview
Web (Mandelbrot) http://quickwayne.googlepages.com/

Blog http://mp-fpga.blogspot.com/

I am happy to discuss with you for more detail and more ideas. You can reach me at quickwayne@gmail.com

Acknowledgement

Thanks for the inspiration and support from Prof. H. Corporaal and Prof. J. van Meerbergen of Technical University Eindhoven, the Netherlands. That's essentially important to me. I also thank Joris van Emden, Marcel Lauwerijssen, Antikons, Paul Gentieu for their reference code.

Reference

1. JPEG codec library based on microblaze, by Sun Wei, Joris van Emden and Marcel Lauwerijssen. http://www.opencores.org/projects.cgi/web/mb-jpeg/overview

2. Embedded JPEG codec libarary by Sun Wei, Joris van Emden and Marcel Lauwerijssen. http://sourceforge.net/projects/mb-jpeg

3. An FPGA-based Soft Multiprocessor System for IPv4 Packet Forwarding by Yujia Jin, Kaushik Ravindran, Nadathur Satish and Kurt Keutzer. http://www.gigascale.org/pubs/688/fpl05_softmp.pdf

4. Mandelbrot Generator by antikons. http://sourceforge.net/projects/mandel/

5. QuickMAN-Fast Mandelbrot Generator by Paul Gentieu. http://sourceforge.net/projects/quickman/

6. Wikipedia http://en.wikipedia.org/wiki/Mandelbrot_set

Appendix A Release Notes

05/05 v0.1

1. Generate MHS, MSS and UCF file from description file for Xilinx EDK 7.1. Note: you need to generate an empty project by BSB in EDK and replace them by generated files.
2. Only for XUPV2P board in this version.
3. Dual port memory communication channel is supported.
4. Support Microblaze processor only.

06/02 v0.11

1. Add support for PowerPC processor.
2. Combine all clock generation to DCM, remove vector_util.
3. Migrate to object-oriented Perl.