
Application Note

# Combining Impulse C™ with uClinux™ for MicroBlaze™-based FPGAs

**David Pellerin, CTO**
Impulse Accelerated Technologies, Inc.
**Scott Thibault, President**
Green Mountain Computing Systems, Inc.

## Overview

Using a small-footprint, open-source operating system such as uClinux can dramatically increase the power and flexibility of FPGA embedded processors. An operating system on an embedded “soft” processor can provide access to standard hardware devices provided in the FPGA-based platform (including network interfaces and other devices) as well as providing multi-tasking capabilities.

By combining software running under the control of an operating system with custom-designed hardware accelerators residing in the FPGA it’s possible to create extreme high-performance applications in which critical algorithms reside as dedicated hardware in the FPGA, while non-critical software components (or software test benches) reside in the embedded processor.

One problem with such an approach has been the relative difficulty of designing the low-level hardware required to implement custom accelerators and their related hardware/software interfaces.

This application note describes how Impulse C and the CoDeveloper tools have been used to solve this problem, allowing mixed software/hardware applications to be described and implemented on FPGAs in an embedded Linux environment, without the need to write low-level hardware descriptions.

## The uClinux operating system

uClinux is a port of the open-source Linux operating system. The uClinux kernel represents a compact operating system appropriate for a wide variety of 32-bit, non-MMU processor cores, with the tradeoff being that multi-tasking must be carefully managed to avoid memory conflicts.

The lack of memory management on uClinux target processors means that there is no protection against badly-behaved processes, which can write anywhere in memory. There is also no support for virtual memory, which also requires memory management. For most embedded applications, however, these restrictions are not critical and in fact the lack of memory management can provide performance advantages and provide opportunities for more direct control over hardware through memory-mapped I/O interfaces.

The kernel has been ported to a number of embedded processors including the Xilinx MicroBlaze. The MicroBlaze port was performed by Dr. John Williams, Research Fellow at the University of Queensland (www.itee.uq.edu.au/~jwilliams/mblaze-uclinux/) using the Memec V2MB1000 prototyping board. This is coincidentally the same board used in tutorials provided in the CoDeveloper for Xilinx MicroBlaze Platform Support Package documentation.

## Combining uClinux and Impulse C

The lack of memory management in uClinux greatly simplifies the use of the MicroBlaze processor as an embedded test bench, or as the primary processing element in a mixed software/hardware application. Because there is no limitation in uClinux on writing to specific memory-mapped addresses, the same code that implements an Impulse C process on MicroBlaze without an operating system can be compiled with little or no change to operate in a multi-tasking environment under uClinux. The lack of a driver layer results in very little impact on the potential throughput rate of data from the processor to the FPGA. (If memory management exists on the processor, such as the case in PowerPC-based Virtex II Pro and Virtex IV platforms, it would be necessary to create a device driver or other abstraction layer between the Impulse C software process and the FPGA-resident processes, resulting in substantial performance degradation.)

The benefits of using uClinux in this way are many. Because the operating system includes many common peripheral interfaces “in the box”, it is relatively easy to create applications in which test data (in the form of actual files and other interfaces) may be read in using standard C calls (such as calls to **fopen**, **fread** and **getc**), then subsequently streamed to the FPGA using the software-to-hardware streaming macros provided with Impulse C. Using a board such as the Memec V2MB1000 it is possible (even trivial) to create a network-enabled device in which the FPGA board becomes a computing node connected to other such nodes, and to desktop computers via FTP or other protocols.

Because Impulse C allows hardware processes to be expressed entirely in the C language, and the CoDeveloper tools automatically generate the needed hardware-to-software interfaces, software engineers using uClinux and Impulse C have everything needed to generate and test complex, hardware-accelerated algorithms.

## A demonstration project

To demonstrate the feasibility of combining Impulse C with uClinux, we chose one of our standard Impulse C sample applications, an image filter. This image filter accepts pixel data on a data stream representing all pixels in a 512X512 image. The filter uses includes pixel packing and unpacking processes that operate in parallel with the actual filter to assemble windows of pixels which are then analyzed to perform a convolution for each pixel in the image. This method is implemented in Impulse C using multiple hardware processes that communicate with a software test bench (also written using Impulse C library calls) residing on the embedded processor.

During processing, pixels are sent by the software test bench (via the FSL interconnect provided with MicroBlaze and via a stream unpacking process written in Impulse C) into the input stream of the first hardware process. In this process the pixels are cached and packaged into three streams (representing three streaming scan lines of the image) which are fed in turn to the image filter process in a parallel pipeline. After filtering, the pixels are re-packed to create 32-bit streams and passed back to the software test bench. The result of this is that an effective maximum throughput of two clock cycles per pixel is achievable. (The actual throughput for this hardware/software test is limited by the speed of the embedded processor, however.)

The filter (which is described in more detail in the *CoDeveloper User's Guide*) is summarized in Figure 1.

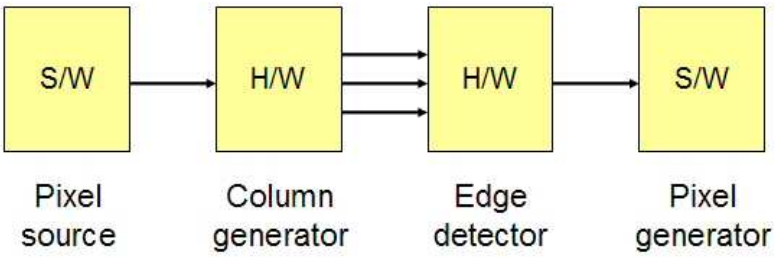


***Figure 1.*** *Edge detection image filter (simplified diagram)*

In order to test this image filter with a variety of images, a software test bench was written that reads image data from an input file (in TIFF format) and streams this data across the FSL interconnect (which is abstracting using Impulse C software macros) to the hardware column generator. After filtering, the processed pixels are read back in by the software test bench using an Impulse C non-blocking stream read macro (**HW_STREAM_READ_NB**). The software test bench then writes the resulting file out to a new TIFF format file containing the filtered image. uClinux provides the required file I/O operations, using a RAM disk provided in the uClinux platform.

To complete the test and allow images to be easily moved from a host PC to the Memec V2MB1000 board, an FTP client is used (via the uClinux console) which communicates with an FTP server running on the PC. The complete demonstration is diagrammed in Figure 2.

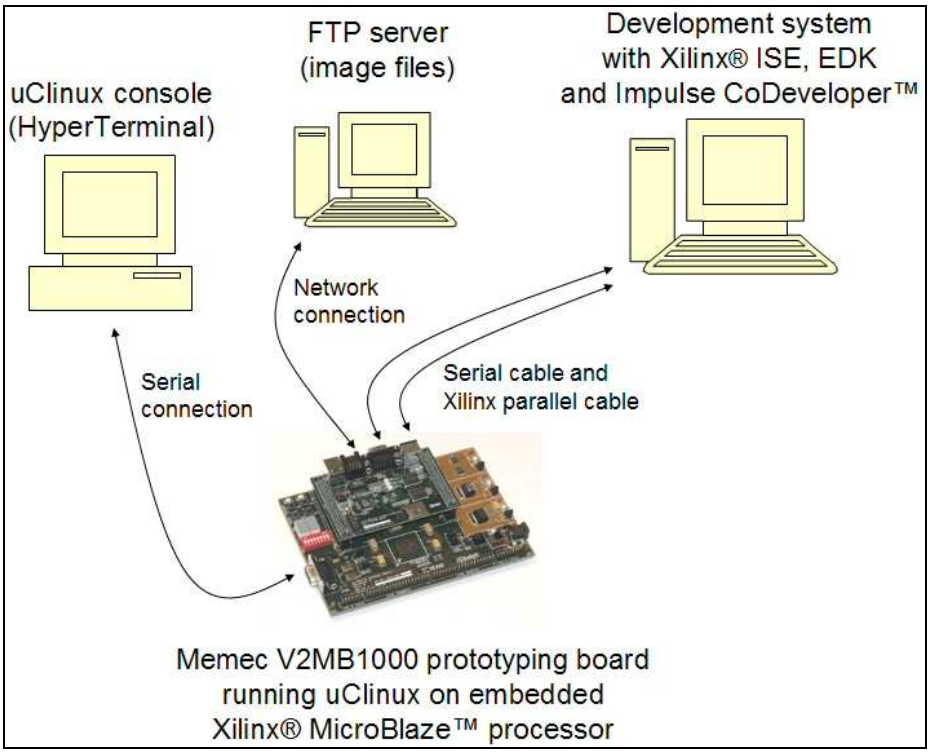


***Figure 2.*** *Demonstration project hardware/software setup*

To run the demonstration on the V2MB1000 (which is available for downloading at www.ImpulseC.com), users follow these steps:

*PREREQUISITES*

To run the demonstration, you must have a Memec V2MB1000 Development board with the P160 Communications Module installed. You must also have a Windows PC with 2 COM ports (or two PCs, one of which simply acts as a console terminal) and have Xilinx EDK 6.2 or 6.3 installed.

Your demonstration machine must have an Ethernet LAN with address 192.168.0.44 available (this will be the board's address on the network).

*PC SETUP*

1. Connect the COM port of the P160 Communications Module to COM1 of the PC.

2. Connect the COM port of the main V2MB1000 board to COM2 of the PC (or to the secondary machine).

3. Connect a LAN cable to the Ethernet port of the P160 Communications Module.

4. Launch **tftpd\tftpd32.exe** on the PC and under settings, set the Base directory to the demonstration directory.

5. Launch a terminal program (such as Hyperterminal) and connect it via COM2 (or COM1 of the secondary machine, if used) at 57600 baud, 8 bits, no parity.

*BOARD SETUP*

1. Open an Xygwin shell and change directory to the demonstration directory.

2. Download the MicroBlaze design with the Impulse C edge detect module:

   ```
   ./configure_fpga.sh
   ```

   (ignore any output in the terminal window)

3. Download the uClinux system file system to the on-board RAM:

   ```
   ./download_image.sh
   ```

   (when complete, the kernel will boot and display messages in terminal)

*RUN THE BENCHMARK ON UCLINUX*

1. Change to /**tmp** directory on uClinux using terminal shell (/**tmp** is a RAM disk):

   ```
   cd tmp
   ```

2. Load the image from the PC using **tftp** from the uClinux console:

   `tftp -g -r peppers.tiff` <IP address of your PC>

3. View the contents of /**tmp**:

   ```
   ls -l
   ```

4. Run the image copy program (performance overhead baseline):

   ```
   copy_img peppers.tiff copy.tiff
   ```

5. Observe the reported time (~985ms)

6. Run the software-only edge detect program:

   ```
   soft_edge peppers.tiff soft.tiff
   ```

7. Observe the reported time (~1962ms = baseline + 977ms)

8. Run the FPGA-accelerated edge detect program:

   ```
   fpga_edge peppers.tiff fpga.tiff
   ```

7. Observe the reported time (~1179ms = baseline + 194ms)

8. Upload the resulting image to the PC:

   `tftp -p -r fpga.tiff` <IP address of your PC>

9. Open fpga.tiff on PC to view the filtered results.

That's all there is to it. What you will have seen when going through this demonstration is an actual comparison of software and hardware implementations of the same image filter algorithm. In the software implementation of the image filter all processing occurs on the MicroBlaze, while in the hardware-accelerated version the two filter processes are implemented in hardware as parallel pipelined processes. The result (when the approximate baseline overhead of file reading and writing is subtracted) is a 5X increase in performance. Note, however, that this performance increase is still throttled by the need to drive the test from a software test bench running on the MicroBlaze processor.

In an actual imaging application (in which pixel data is streaming directly from a hardware interface such as a camera) the potential performance gains are dramatically higher, with two-cycle pixel throughput rates achievable due to automatic parallelizing by the CoDeveloper compiler of the core image filter processes. If there was no embedded processor in the system, the actual time to process one image at 50MHz would be just 11ms as opposed to the approximately 194ms time observed in this test. (512 X 512 pixels at two cycles per pixel equals 524288 clock cycles, which equates to 10485.76 us @ 50MHz.) Using the embedded processor as a test bench, however, makes the testing of such applications much simpler.

## Summary

This application note has described how the uClinux operating system can be an effective way to generate highly capable software test benches for custom hardware modules developed using Impulse C and CoDeveloper. There are many other potential applications for uClinux and Impulse C, including the ability to described multiple (threaded) Impulse C software processes and to create highly reconfigurable systems consisting of multiple MicroBlaze cores, each running uClinux in combination with Impulse C hardware processes. Research is continuing into such multi-processor, mixed software/hardware applications, which appear to be limited only by our imaginations.