uC/OS II vs uCLinux

Hala Ayoub
Humber College
2 PUBLICATIONS **1** CITATION

uC/OS II vs uCLinux

Peter McLean
University of Ottawa
Ottawa, Ontario
Email: peter.cd.mclean@gmail.com

Hala Ayoub
University of Ottawa
Ottawa, Ontario
Email: hayoub@uottawa.ca

Dr. Miodrag Bolic
University of Ottawa
Ottawa, Ontario, Canada
Email: mbolic@site.uottawa.ca

December 4, 2009

***Abstract*—The uClinux port is a derivative of Linux kernel intended for microcontrollers without memory management units (MMU) [1]. It provides a single shared address space for all processes. Whereas, uC/OS-II is a portable, ROMable, scalable, preemptive, real-time deterministic multitasking kernel for microprocessors, microcontrollers and DSPs [2]. In this paper, we implemented uCos and uClinux kernels on the same NIOS-II platform and compared the performance.**

***Index Terms*—Embedded OS, uClinux, uCos, benchmarks, performance,**

Requirements

Hardware and Software requirements: This project requires the following hardware and software components:

- Windows XP Pro
- Quartus II Development Software version 9.0
- NIOS-II IDE 9.0SP2
- SOPC Builder
- Ubuntu Linux
- Altera Stratix II EP1S10 FPGA
- Altera USB-Blaster.

Motivation

Embedded operating systems are becoming increasingly popular in devices such as mobile phones, GPS, house-hold appliances, and medical devices. Its important to choose the right operating system to fulfill the requirements. When choosing the operating system its important to take into account the tradeoff performance vs. features. This paper illustrates the basic performance differences between uC/OS II and uClinux using simple MiBench benchmarks.

I. Introduction

A. Operating Systems

Real Time Operating System (RTOS) is a multitasking operating system intended for real time applications. It facilitates the creation of real time system. Some key features of RTOS are the minimization of interrupt latency and thread switching latency. uC/OS II is a RTOS developed by Micrium. It supports the NIOS II processor which uses the Integrated Development Environment (IDE) based on Eclipse. It uses pre-emption to run tasks and interrupt service routines. It provides semaphores, mutexs, message mailboxes, message queues, timers, event flags, and memory partition management. [2]

Linux is a UNIX-like operating system that is open source and free to use under the GNU General Public License. uClinux is an port of the Linux operating system targeting embedded devices without an Memory Management Unit (MMU). It originally ported Linux 2.0, but has ports based on Linux 2.2 and Linux 2.6 now. There is support for many different processors from ARM to microBlaze and of course NIOS II. [1]

B. Benchmarks - Mibench

MiBench is a set of benchmarks intended to evaluate performance of embedded devices. [3] The benchmarks are intended to be directly related to industry applications such as Automotive, Telecomm, and Consumer. Using MiBench we observed the effects of the operating system on regular non-threaded applications.
Telecomm - FFT: is considered a telecomm benchmark that performs fast fourier transform on an array of data. Fourier transforms are used in digital signal processing to find the frequencies contained in a given input signal. The input data is a polynomial function with pseudorandom amplitude and frequency sinusoidal components.
Automotive - Basicmath: is an automotive industrial benchmark that performs simple mathematical operations such as round, cubic function solving using constant inputs that do not have dedicated hardware support in embedded processors.

C. Benchmarks - Thread Metric

uClinux can be an RTOS. uCLinux supports the POSIX pthread API. Pthreads are functionally equivalent to tasks within uC/OS II, with a loss of some of the finer control mechanisms. There is support for mutexs, semaphores, priority-based pre-emption and round robin scheduling for threads of equal priority.

There are some subtle difference between Linux pthreads and uC/OS II tasks. uC/OS II does not support tasks of equal priority, while pthreads and uC/OS III do. Pthreads does not support thread suspension. Sleeping within a pthread sleeps the whole process. When uC/OS II is started using OSStart() that function call never returns while in pthreads the main process continues after thread creation, but you can block it using join calls. These differences add up quickly and make it very difficult to compare both with a standard benchmark.

Thread-Metric is an RTOS benchmark developed by Express Logic [4]. By implementing the uC/OS II and uClinux pthreads we can compare the performace of the linux pthreads with uC/OS II tasks, but we expect that uC/OS II will outperform the pthreads.

II. Procedures

Obtain the pre-built EP1S10 .sof hardware image file and .ptf from nioswiki ”TryOutUCLinux” [5]

```
NiosII_stratix_1s10_full_featured.sof
and
NiosII_stratix_1s10_full_featured.ptf
```

Install the sof to the board by using the Quartus II programmer

A. *uC/OS II*

Procedure for compiling and running a uC/OS II program [6]:

1) Create a NIOS-II IDE C/C++ application project
2) Use the MicroC/OS-II RTOS Hello World example as a template
3) Choose the .ptf file and follow the instructions to create a new microC/OS-II project
4) Create the system library of your project with MicroC/OS II selected as the RTOS, and the rest as default
5) The RTOS is configured using a wizard launched from the System Library properties window. The default configuration was used
6) Compile your project
7) Create a run configuration and download your program to the hardware
8) Now you have configured, built and run the microC/OS-II program
9) Run the benchmarks 5 times

B. *uCLinux*

Procedure for compiling and booting uCLinux: [5]

1) You must have a Linux (Ubuntu) PC with software development packages.
2) Install these packages on Ubuntu by running the following commands:
   ```
   sudo apt-get update
   sudo apt-get install git-core git-gui make gcc ncurses-dev bison flex gawk gettext ccache zlib1g-dev libx11-dev texinfo liblzo2-dev pax-utils uboot-mkimage corkscrew
   ```
3) check if the default shell is bash by running:
   ```
   ls -l /bin/sh
   log out and in
   check if you have ”cc” which is a symlink to ”gcc” by running:
   which gcc
   gcc -v
   which cc
   cc -v
   ```
4) Download the tar file nios2-linux-20090730.tar by running:
   ```
   wget http://www.niosftp.com/pub/uclinux/nios2-linux-20090730.tar
   ```
5) Obtain the pre-built byinary tool chain by downloading:
   ```
   wget http://www.niosftp.com/pub/gnutools/nios2gcc-20080203.tar.bz2
   ```
6) Verify that the tools are in the path by calling:
   ```
   nios2-linux-uclibc-gcc -v
   ```
7) Build the kernel and apps using uCLinux-dist sourses
8) Run:
   ```
   cd uClinux-dist
   make menuconfig
   ```
9) Compile the benchmarks by:
   ```
   nios2-linux-uclibc-gcc .c -o xxx.c -elf2flt
   ```
10) This will change the format of the compiled program to Binary FLAT format.
11) Copy the file to romfs/bin dir using:
    ```
    cp xxx  /nios2-linux/uClinux-dist/romfs/bin
    ```
12) Rebuild the kernel image using:
    ```
    cd  /nios2-linux/uClinux-dist
    make
    ```
    Using the nios2 command shell download the kernel image to the board.
    ```
    nios2-download -g zImage nios2-terminal
    ```

III. Results

A. *Benchmarks - MiBench*

TABLE I
MiBench Results

	Basicmath_large (cycles)		Fast Fourier Transform (cycles)	
Test #	uC/OS II	uCLinux	uC/OS II	uCLinux
1	5673109015	6428276654	40739174	45358819
2	5673109015	6425169087	40739174	45275550
3	5673109015	6428180338	40739174	45275880
4	5673109015	6425264271	40739174	45219619
5	5673109015	6428190538	40739174	45238118

uC/OS II acheives an average speed-up over uClinux of 1.1328398622 in the basicmath_large test and 1.1113038 in the fft test.

B. *Benchmarks - Thread Metric*

Out of 8 tests in the Thread Metric benchmark only 1 was actually possible to implement due to difference in implementation between pthreads and uC/OS II. However, upon attempting to perform the benchmark on the board the pthread version of the benchmark would not run. The issue is that the Thread Metric tests use a sleep to stall the reporting task until the working task has finished. Pthreads does not supply an individual thread sleep since usleep() sleeps the entire process. A working hack sleeps the thread in cygwin during testing using a timed wait on a mutex, but in practice it does not work in uClinux.

IV. Discussion

In both benchmarks uC/OS II outperformed uClinux. This is because uC/OS II will not pre-empt the working process since it is the only process. In uClinux, however, the operating system services will run when needed or run while the process blocks on I/O. This interference adds the overhead seen while comparing these two.

The Thread Metric testing was a dissapointment. Pthreads provides functionality very similar to the uC/OS II task system with many of the same supporting features. However, due to the nature of the Thread Metric tests and the differences that do exist between the two no tests were successfully executed.

Conclusion

In this paper, we compared the perfomance between uCLinux and uC/OS running on the same hardware platform using MiBench benchmarks. As well, comparisons were made between POSIX pthreads functionality and uC/OS II tasks. uC/OS II outperformed uCLinux in the basicmath_large and fft tests because it does not pre-empt a single thread.

uC/OS II is an apple and uClinux is an orange. They are intended to be used under different requirements. If you require very fast interrupt disabling and task switching then you should use a RTOS such as uC/OS II. If you require feature rich file system, TCP/IP, video, and other peripherals then you should use uClinux.

References

[1] G. Ungerer, D. J. Dionne, and M. Durrant. What is uclinux? http://www.uclinux.org/description/. Arcturus Networks Inc.

[2] *uC/OS-II TM Real-Time Operating System*, http://micrium.com/newmicrium/uploads/file/datasheets/ucosii_datasheet.pdf, Micrium, May. 2009.

[3] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in *Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on*, Dec. 2001, pp. 3–14.

[4] W. Lamie and J. Carbone. (2007, Oct.) Measuring real-time performance of an rtos. http://www.rtos.com/PDFs/MeasuringRTOSPerformance.pdf. Express Logic Inc. 11423 West Bernardo Court, San Diego, CA. 92127. Benchmark source available at ftp://ftp.embedded.com/pub/2007/05Carbone_ThreadMetric/TM_CMP.zip.

[5] M. Eltoddo, mschnell, hippo, Admin, and ... (2009, Oct.) uclinux. http://www.nioswiki.com/OperatingSystems/UClinux. See also: TryOutuClinux, BinaryToolchain, UClinuxDist, and CompileHello.

[6] *Using MicroC/OS-II RTOS with the Nios II Processor Tutorial 2007.*, http://www.altera.com/literature/tt/tt_nios2_MicroC_OSII_tutorial.pdf, Altera Corporation, 101 Innovation Drive, San Jose, CA 95134, jan. 2007.