PERFORMANCE COMPARISON OF XILKERNEL AND µC/OS-II ON MICROBLAZE

[1]MUTHUKUMAR N, [2]JAISHANTHI SAMMBANDAM

[1]P.G. Scholar, [2]M.E., Assistant Professor, Department of ECE, Easwari Engineering College, Chennai
E-mail: [1]muthukumar_nm@yahoo.co.uk, [2]shanthisindia@yahoo.com

Abstract- Soft core processors are increasingly used in embedded system designs based on Field Programmable Gate Array (FPGA) devices. Real time operating systems (RTOS) play an important role in design of embedded systems using microcontroller or microprocessors. MicroBlaze is a soft processor core designed for use with Xilinx FPGA device families. This paper compares the performance of RTOSs Xilkernel and µC/OS-II on a MicroBlaze soft core processor. Performance comparison is based on time taken for Context Switching Time between tasks, Message passing between task, Allocation and Release of memory blocks and Memory Footprint, which are measured through simple software based technique.

Index Terms- MicroBlaze, RTOS Performance Benchmarks, µC/OS-II, Xilkernel.

I. INTRODUCTION

System-on-Chip (SoC) based designs are preferred in embedded system design than a design based on several chips. SoC using FPGA offers many advantages over the conventional approach using Application Specific Integrated Circuits (ASIC), like Faster time to market, No Non-Recurring Engineering (NRE) Cost, Simple Design Cycle, Field programmability and Volume flexibility. Generally FPGAs are used for lower speed, lower complexity and lower volume designs. But today's FPGAs even run at 500 MHz and by making use of inherent parallelism and flexibility concepts of FPGAs, such an approach can yield superior performance. With unprecedented logic density increase and a host of other features, such as embedded processors, DSP blocks, clocking, and high-speed serial at ever lower price, FPGAs are suitable for almost any type of design.

The embedded processors available within a FPGA device can be implemented as a soft or hard core. A hard processor core has dedicated silicon area on the FPGA. A soft-core processor solution is one that is implemented entirely in the logic primitives of an FPGA. A soft core implementation may not operate at the same clock frequencies or have the same performance of a discrete hard core solution. But, in many embedded applications, performance can be traded for expanded functionality, reduced cost, and flexibility. MicroBlaze and Nios II are leading soft core processors available with popular family of FPGA devices.

Using a real time operating system (RTOS) is becoming indispensible in real time embedded system design owing to RTOSs' deterministic behavior and efficient resource management characteristics. Ability to create tasks to handle and distribute huge sized codes, existence of scheduling algorithms to manage the tasks and efficient interrupt handling and faster memory allocation are some of the many advantages of using an RTOS. An RTOS needs extra memory, additional computational resources, but normally the advantages outweigh

Comparing the performance of RTOSs according to different benchmark parameters will be helpful in choosing the RTOS for the system design. For a designer using MicroBlaze soft processor on a Xilinx FPGA platform, Xilinx provides Xilkernel as a choice for RTOS. µC/OS-II is one of the popular RTOSs from 3rd party vendors that support MicroBlaze soft processor. Many of research publications that compare RTOS performance suggest an approach based on frequently used system services and their execution time measurement. [1] evaluates many RTOSs for use with small microcontrollers, four RTOSs are shortlisted and compared on Renesas M16C/62P 16-bit microcontroller platform. In [2], metrics based on duration of context switching, message transfer and task synchronization are proposed and evaluated for QNX distributed RTOS platform. Real time performance is emphasized in [3] for RTOS comparison, the methodology involves specific test set up and procedure to determine work case response time for latency, latency jitter and response time.

The current work presents a simple set of software based benchmark tests to compare performance of Xilkernel and µC/OS-II. The following sections explain the benchmark environment used, benchmark parameters, their measurement methodology and the results obtained.

II. BENCHMARK ENVIRONMENT & RTOSS

The MicroBlaze embedded processor soft core is a

reduced instruction set computer (RISC) optimized for implementation in Xilinx FPGA family of devices. The architecture supports 70 configuration options [4], helping designers to create either a very small footprint embedded microcontroller or a high performance embedded computer running various RTOSs. It has rich instruction set of around 120 instructions divided into two types based on 32-bit or 16-bit operands and using two addressing modes.

The low-cost Spartan-6 FPGA LX9 Microboard is used as target platform for benchmarking and its salient features are listed below.

- Spartan-6 XC6SLX9-2CSG324C FPGA
- 64 MB LPDDR SDRAM
- 128 Mb Multi-I/O SPI Flash
- 10/100 Ethernet PHY
- USB-to-UART port
- On-board USB JTAG circuitry
- Programmable clock chip
- Four LEDs, 4-bit DIP switch, Reset and PROG push-buttons

One major advantages of choosing LX9 Microboard for benchmarking MicroBlaze is many pre-built hardware designs available with it, of which AVS6LX9MBHP011 has been used for benchmarking. This reduces the effort on hardware platform design for this bench marking exercise. The bitstream files representing processor design are made available by Xilinx, which can be exported to SDK (Software Development Kit) to set-up the hardware platform on Spartan 6 FPGA. The block diagram of chosen processor [5] is given below.

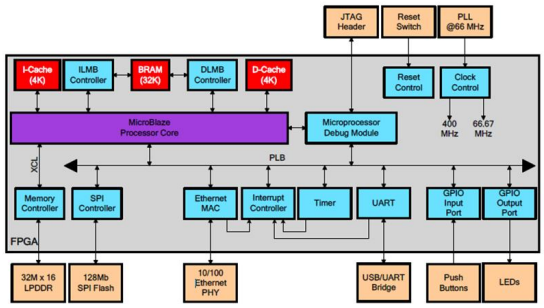


Fig 1. AVS6LX9MBHP011

Xilkernel is native RTOS for embedded processors from Xilinx. It is highly integrated with Xilinx family of design tools. It allows the users to customize the kernel both in terms of size and functionality to suit their requirements. It provides POSIX compliant APIs covering core kernel and other features like Thread management, Semaphores, message queues, memory management and software timers. Xilkernel for MicroBlaze is available as software library with the Xilinx SDK, the tool chain used for benchmarking. MicroCOS-II, commonly termed as µC/OS-II or uC/OS-II, is the acronym for Micro-Controller Operating Systems Version 2. It is a priority-based pre-emptive real-time multitasking operating system kernel for microprocessors, written mainly in C programming language. Its features and API set are similar to the one supported by Xilkernel, but the APIs are not POSIX compliant. It has been ported to over 100 microcontrollers/processors and is freely available for research and academic work. A port of µC/OS-II for MicroBlaze on LX9 Microboard is readily available from Micrium, the vendors for µC/OS-II.

III. BENCHMARK PARAMETERS & METHODOLOGY

The current work follows approach of comparison of RTOS performance based on the time taken by them to perform specific services. The resulting time values will be a measure of RTOS overhead. The selected services must be commonly found in many RTOSs and represent typically used services in building any embedded system using RTOS. The benchmark parameters chosen are Context Switching Time between tasks, Message passing between task, Allocation and Release of memory blocks and Memory Footprint.

The following aspects characterize the software written for the benchmark tests.

- The tests are iteration based, count of iterations reported for fixed duration of 1 second.
- Execute the service and its inverse in pairs; for instance send message and get message
- Variables keep track of execution counts, initialized after reporting for next cycle of test
- Simple printf function is used to report the counts
- No special or extra Hardware required
- Coded in C, easy to extend for other RTOS and adapt to different platform

For each benchmark parameter the execution count as well as memory footprint (Code and Data memory usage) will be collected.

The target processor has 32 KB BRAM as local memory, 64 MB External RAM, 16 MB Flash, Instruction and Data cache of 4 KB each. The LX9 microboard platform uses a clock of 66.67 MHZ. The OS tick value for both the RTOSs is set as 10 milliseconds and also task stack size is set to the same value.

A. Context Switching Time between tasks

Reporting Task (Highest Priority)	Task 1 (High priority)	Task 2 (Low Priority)

initialize count; sleep for 1 second; report count;	get semaphore; increment count;	increment count; release semaphore;

Fig 2. Context Switching Time between tasks

This parameter represents the time taken by the RTOS to transfer the execution context from one task to another. µC/OS-II has APIs to suspend and resume tasks, OSTaskSuspend() and OSTaskResume(), but Xilkernel does not have equivalent APIs. Hence semaphore related APIs are used to switch execution between two tasks in a repeated manner. The benchmark test is illustrated in fig 2.

TABLE I. APIS FOR CONTEXT SWITCHING TIME

Action	Xilkernel	µC/OS-II
Get Semaphore	sem_wait()	OSSemPend()
Release Semaphore	sem_post()	OSSemPost()

Two test tasks Task 1 and Task 2 can execute only when Reporting task is sleeping. Task 1 starts executing, performs a get semaphore action. Semaphore used is created to act as signal and Task 1 gets suspended until semaphore is released by Task 2. Task 2 starts executing, releases semaphore, but gets suspended immediately as high priority Task 1 is released. This processing repeats until reporting task wakes up to report count, initialize count and sleeps again. Table I give the respective APIs used for Xilkernel and µC/OS-II

B. Message passing Time between tasks

Message passing is important Inter Process communication (IPC) mechanism used for synchronization between tasks. Fig 3 explains the method used to measure message passing time between two tasks. Both the RTOSs use message passing mechanism involving queues that use memory pointers to copy messages into message queue areas globally accessible among tasks.

Reporting Task (Highest Priority)	Task 1 (High priority)	Task 2 (Low Priority)
initialize count; sleep for 1 second; report count;	retrieve message from queue; increment count;	put message onto queue;

Fig 3. Message passing time between tasks

Again, Two test tasks Task 1 and Task 2 can execute only when Reporting task is sleeping. When high priority Task 1 starts executing, it performs message retrieval action from queue and gets suspended as there is no message in queue yet. This results in Task 2 executing, which puts a message in the queue. When message become available in queue, the waiting high priority Task 1 is activated immediately and Task 2 gets suspended. The cyclic processing goes on until Reporting task intercepts to report count and initialize count. The table II gives the RTOS APIs involved. The message passing time includes the processing time to copy message from and to queue as well as context switching time between the message producer and consumer tasks.

TABLE II. APIS FOR MESSAGE PASSING TIME

Action	Xilkernel	µC/OS-II
Retrieve message	msgget()	OSQPend()
Put message	msgsnd()	OSQPost()

C. Allocation/Release of memory blocks

When embedded system involves dynamic memory requirements, fixed size memory blocks can be allocated, used and released after their use. These blocks are pre-created as a structure comprising of equal sized contiguous memory chunks connected by pointers similar to a linked list. Such a structure is often referred as buffer pool. The fixed size memory block allocation eliminates non-determinism associated with variable sized dynamic memory allocation. Fig 4 gives the method used to measure Allocation/Release time of memory blocks.

Reporting Task (High Priority)	Task 1 (Low Priority)
initialize count; sleep for 1 second; report count;	allocate memory block; release memory block; increment count

Fig 4. Allocation/Release time of memory blocks

Task 1 allocates a memory block of size 128 bytes and releases it immediately. This process gets repeated with the reporting task intercepting to report count once every second. Table III gives the RTOS APIs involved in the benchmark test.

TABLE III. APIS FOR ALLOCATION/RELEASE OF MEMORY BLOCKS

Action	Xilkernel	µC/OS-II
Allocate memory block	bufmalloc()	OSMemGet()
Release memory block	buffree()	OSMemPut()

D. Memory Footprint

As each of the task sets for benchmark tests are compiled and linked, the associated messages from the

SDK, give details on code size and data memory size for the executable images. Both RTOS can be configured to include only necessary features, excluding unwanted features, for example mutex locks. Also it is ensured that the stack size for each task is the same under both RTOSs.

IV. BENCHMARK RESULTS

The hardware specifications in the form of bitstream files for MicroBlaze processor are downloaded onto Spartan 6 LX9 microboard via JTAG connector provided with the board. The benchmarking code is written in C under Xilinx SDK, organized as separate project for each parameter. Both RTOS libraries are organized as Board Support Package projects to be compiled and linked together with benchmark software projects. To run the benchmark tests, executable image from the respective software projects are downloaded to RAM, leaving the hardware project and processor same for different the test runs.

TABLE IV. EXECUTION COUNTS FOR BENCHMARK PARAMETERS

	Execution count per second	
Parameter	Xilkernel	µC/OS-II
Context switching time between tasks	48424	11770
Message passing between tasks	11570	5787
Allocation/release of memory blocks	175610	35266

TABLE V. MEMORY FOOTPRINT DATA

Context Switching time between tasks					
	text	data	bss	dec	Hex
Xilkernel	27846	1480	29618	58944	e640
µC/OS-II	22218	420	33182	55820	da0c
Message passing between tasks					
	text	data	bss	dec	Hex
Xilkernel	27834	1484	29630	58948	e644
µC/OS-II	21442	436	33222	55100	d73c
Allocation/Release of memory blocks					
	text	data	bss	dec	Hex
Xilkernel	27858	1496	30526	59880	e9e8
µC/OS-II	21270	424	25514	47208	b868

The benchmark tests produce report of execution counts per second, the summary of results is given in Table 4. Table 5 gives the memory footprint data collected during compilation of benchmark code. The results show the context switching time overhead of µC/OS-II is approximately four times that of Xilkernel; its message passing time and memory block allocation/release time are respectively twice and five times that of Xilkernel. The memory footprint data of both the RTOS are comparable, the code size of µC/OS-II is slightly (20-25%) lower than that of Xilkernel. The RTOS configuration had only equivalent features enabled and the memory footprint results do not favor decisively any one RTOS.

CONCLUSION

The comparison of benchmark results with respect to the parameters considered show Xilkernel in better light compared to µC/OS-II. Also Xilkernel has easy integration with Xilinx tools. On the other hand, µC/OS-II has easy portability to other processors and wide spread design experience. Another advantage with µC/OS-II is availability of its complete source code. Choice of RTOS needs to consider many factors like language support, tools compatibility & availability, technical support and licensing scheme along with performance to meet the design goals. In future, the results out of the current work need to be augmented to include more parameters, for example, Interrupt latency to enhance its usefulness to designers. Also, the comparison must include other RTOSs that support MicroBlaze.

REFERENCES

[1] T. Anh and S. Tan, "Real-Time Operating Systems For Small Microcontrollers", IEEE Micro, vol. 29, no.5, pp. 30-45, Sept.-Oct, 2009.

[2] K. M. Sacha, “Measuring the real-time operating system performance”, Seventh Euromicro workshop on Real-time systems proceedings, Odense, Denmark, pp. 34–40, 1995.

[3] Rafael V. Aroca and Glauco Caurin, “A Real Time Operating Systems (RTOS) Comparison”. Workshop de Sistemas Operacionais (WSO) – Congresso da Sociedade Brasileira de Computação (CSBC), 2009.

[4] Gokhan Ugurel and Cuneyt F. Bazlamacci, "Context switching time and memory footprint comparison of Xilkernel and µC/OS-II on MicroBlaze", 7th International Conference on Electrical and Electronics Engineering (ELECO), no. 2, pp. 62-65, 2011.

[5] Avnet, "MicroBlaze Hardware Platform Overview", https://www.em.avnet.com/J900_AvtS6LX9MicroBoard_MicroBlaze_Hardware_Platforms_04.pdf

[6] Xilinx, "Xikernel v5.01.a Documentation", http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_2/oslib_rm.pdf

[7] Jean J. Labraosse, "MicroC/OS-II The Real-Time Kernel", CMP Books, 2002.

★★★