Real-Time Operating Systems for Systems on a Chip

About Bob Zeidman

- Founder of Zeidman Technologies, Zeidman Consulting
- Engineering consultant since 1987
- Clients include Apple Computer, Cisco Systems, Mentor Graphics, and Texas Instruments
- Author of *Verilog Designer's Library, Introduction to Verilog, Designing with FPGAs and CPLDs,* and articles on engineering and consulting
- Patents on software synthesis, hardware synthesis, emulation
- Degrees from Cornell and Stanford

Introduction

- This seminar examines different options for putting a real-time operating system (RTOS) on a system on a chip (SOC)
  - Purchase an off-the-shelf RTOS
  - Write your own RTOS
  - Synthesize an RTOS

Introduction

- Intended Audience
  - Software engineers who need to design a multitasking embedded system for an SOC and who are concerned about cost, development time, efficiency, and reliability
- Results
  - You will understand the requirements, limitations, tradeoffs, and tools available for implementing an RTOS for an SOC
- Prerequisites
  - Basic understanding of FPGA design or ASIC design
  - Understanding of systems on a chip (SOCs)
  - Knowledge of programming
  - Knowledge of real-time operating systems (RTOSes) is helpful but not required

What is a System on a Chip?

- For our purposes, a SOC is one that includes a microprocessor.

What is an RTOS?

- "A program that schedules execution *within specified time constraints*, manages system resources, and provides a consistent foundation for developing application code."

  Real-Time Concepts for Embedded Systems

What is a “Hard RTOS?”

- Has time constraints that must be met under any and all conditions for certain tasks.

What is an OS?

- Only kernel – the core supervisory software that provides minimal logic, scheduling, and resource management algorithms?
- A combination of various modules, including the kernel, a file system, networking protocol stacks, and other components required for a particular application?
- For our purposes, it is the kernel (task manager). The drivers and applications will be considered separately.

Do You Need An RTOS?

- If your SOC will execute more than one task
- Communication with other processors is a task

Purchasing an RTOS

- “Off-The-Shelf”
- Object Code vs. Source Code
- Royalties vs. Royalty Free

Purchasing an RTOS

- Debugged and tested
- Integrated with existing tools
- Comes with support
- Requires little in-house specialized knowledge

Writing Your Own RTOS

- About half of all embedded systems projects still use a proprietary, "home-grown" RTOS (finally changing)
- Protects intellectual property
- Maintains control over code
- Reduces complexity and size
- Requires significant in-house expertise

Synthesizing an RTOS

- New technology
- Source code in, source code out
- Integrated with existing tools
- Requires no in-house specialized knowledge
- Protects intellectual property
- Maintains control over code
- Reduces complexity and size

RTOS Considerations

- Run Time Issues
- Development Time Issues
- SOC Issues

Run Time Issues

- Maintainability
- Performance
- Predictability
- Reliability
- Scalability
- Size

Maintainability

- Who fixes bugs in the field?
  - Purchased RTOS: Shared between you and vendor
  - In-house RTOS: You
  - Synthesized RTOS: You

Performance

- Task latency times
- Interrupt latency times
- Data throughput
- Task execution times
- Other

Predictability

- Purchased RTOS
  - Characterized by vendor
- In-house RTOS
  - You must perform exhaustive testing
- Synthesized RTOS
  - Static timing analysis

Reliability

- **Deadlock.** A situation where a task is blocked from executing because it is waiting for a resource to become available while that resource is directly or indirectly waiting for the task to continue.
- **Priority inversion**. A situation where a high priority task is delayed while waiting to access a shared resource even though the resource is free to be used. In effect, the high priority task has been given a very low priority.
- **Race conditions.** This occurs when the outcome of an embedded system depends on the specific order in which tasks are executed.
- **Starvation.** A task cannot continue because it is waiting for a resource, but the operating system, usually due to a bug, will not give the task access even though the resource is available.

Scalability

- Purchased RTOS
  - Much functional scalability
- In-house RTOS
  - Difficult to design in scalability
- Synthesized RTOS
  - Need to re-synthesize entire system

Size

- Purchased RTOS
  - Modules removed to decrease size
  - Compression
- In-house RTOS
  - Can be designed to be fairly small
- Synthesized RTOS
  - Extremely small – automatically minimized according to application requirements

Development Time Issues

- Configurability
- Cost
- Driver and application libraries
- Maintainability
- Portability
- Scalability
- Standard interfaces
- Tool chain support

Configurability

- Purchased RTOS with source code
  - Doable but difficult
- Purchased RTOS without source code
  - Not possible
- In-house RTOS
  - Very configurable
- Synthesized RTOS
  - Automatically configurable

Cost

- Linux
  - There's no such thing as a free lunch
  - Development time
  - How does Monta Vista make money?

Driver and Application Libraries

- Purchased RTOS with source code
  - Extensive, supplied by open source community
- Purchased RTOS without source code
  - Large, supplied by vendor
- In-house RTOS
  - Small
- Synthesized RTOS
  - Small now, but just wait

Maintainability

- Who improves RTOS?
  - Purchased RTOS: Vendor
  - In-house RTOS: You
  - Synthesized RTOS: Tool vendor

Portability

- Who ports to new processors?
  - Purchased RTOS: Vendor
  - In-house RTOS: You
  - Synthesized RTOS: Automatic

Scalability

- Ability to add new functionality
  - Purchased RTOS: Scalability built in
  - In-house RTOS: If you build it in
  - Synthesized RTOS: Automatic

Standard Interfaces

- Purchased RTOS
  - Many standard interfaces built in
- In-house RTOS
  - Whatever you build in
- Synthesized RTOS
  - Uses currently non-standard interfaces

Tool Chain Support

- Purchased RTOS
  - Typically very good support
- In-house RTOS
  - Typically not very good support
- Synthesized RTOS
  - Very good support for basic tools
  - Ability to easily add support for advanced tools

SOC Issues

- Configurability
- Portability
- Scalability
- Size
- Tool chain support

Conclusions

- Many choices for an RTOS
- Purchasing RTOS
  - Object code
  - Source code
- Writing RTOS
- Synthesizing RTOS
- Issues regarding the RTOS
  - Run time
  - Development time
  - SOC issues

ZEIDMAN TECHNOLOGIES

15565 Swiss Creek Lane
Cupertino, CA 95014
Tel (408) 741-5809
Fax (408) 741-5231
www.zeidman.biz

Bob Zeidman
President
bob@zeidman.biz