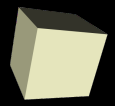
Estructura general de los Sistemas Empotrados

Manuel J. Bellido Díaz

Octubre 2016

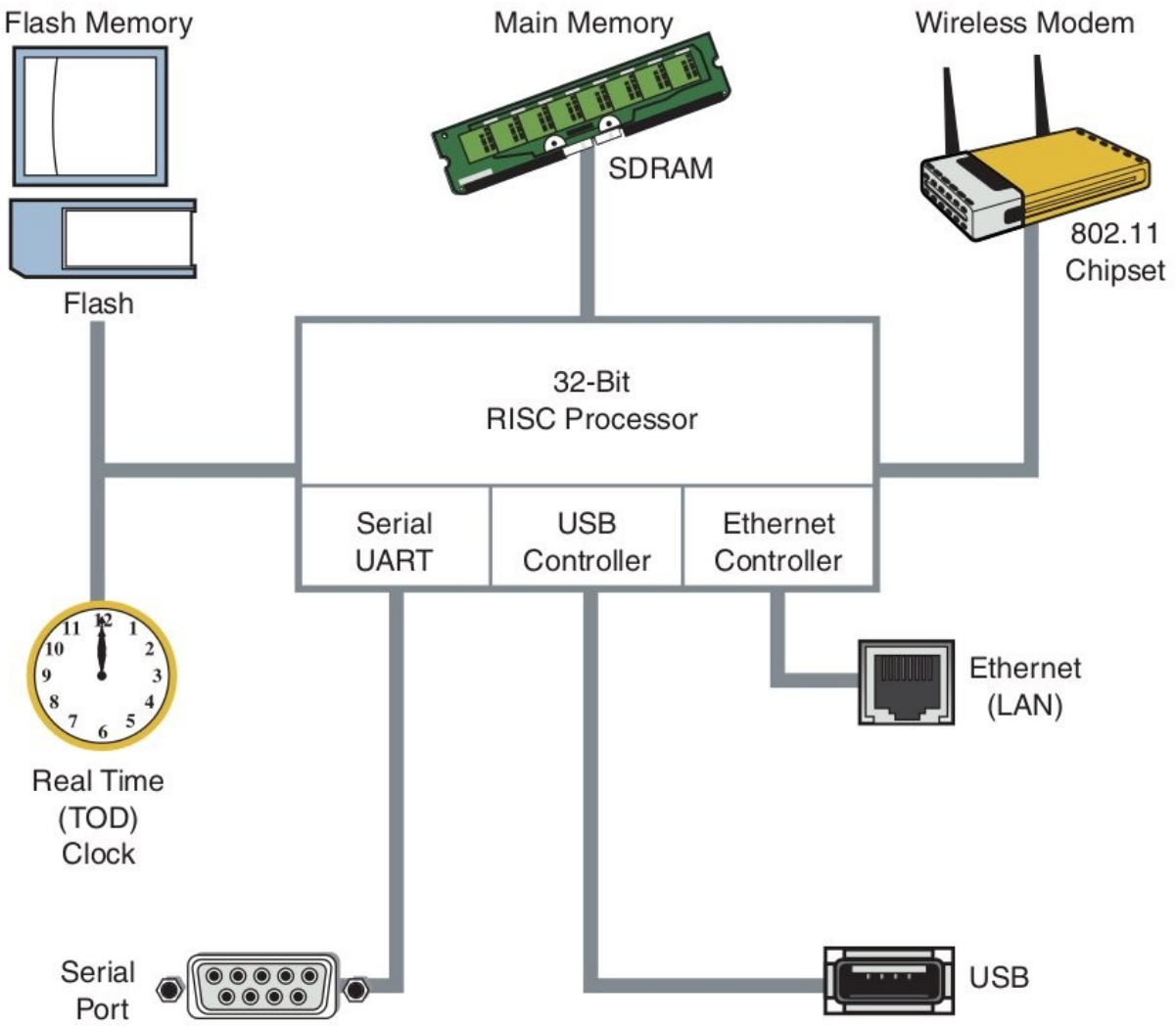


Guión del Tema

- Anatomía de Un Sistema Empotrado
- Arquitectura Hardware de un SE
- Arquitectura Software de un SE
- Desarrollando un Sistema Empotrado

Anatomía de Un Sistema Empotrado

- Arquitectura Hardware de alto-nivel de un SE
  - Núcleo principal: Microprocesador 32-bit que incluye controladores de periféricos: SoC
  - Capacidad de Almacenamiento:
    - Memoria FLASH
    - Memoria RAM
  - Interfaces:
    - Ethernet, USB, Puerto Serie para consola, Sistema de debug (JTAG, SWD, ...)



Anatomía de Un Sistema Empotrado

- Arquitecturas softwares de un S.E.
  - Modo standalone (bare metal)

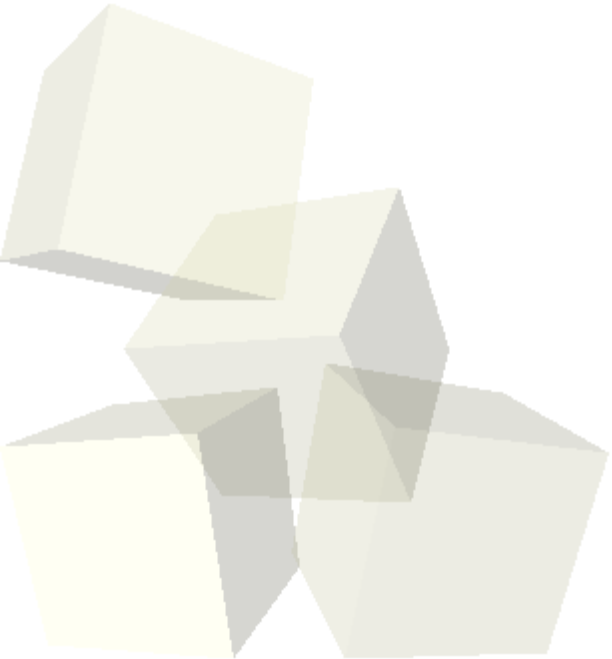
(ejemplos microcontroladores de 8bits, aplicaciones muy específicas del SE)

  - Modo Sistema Operativo

→ SE de cierto nivel de complejidad (arquitectura similar a la de un *desktop computer*)

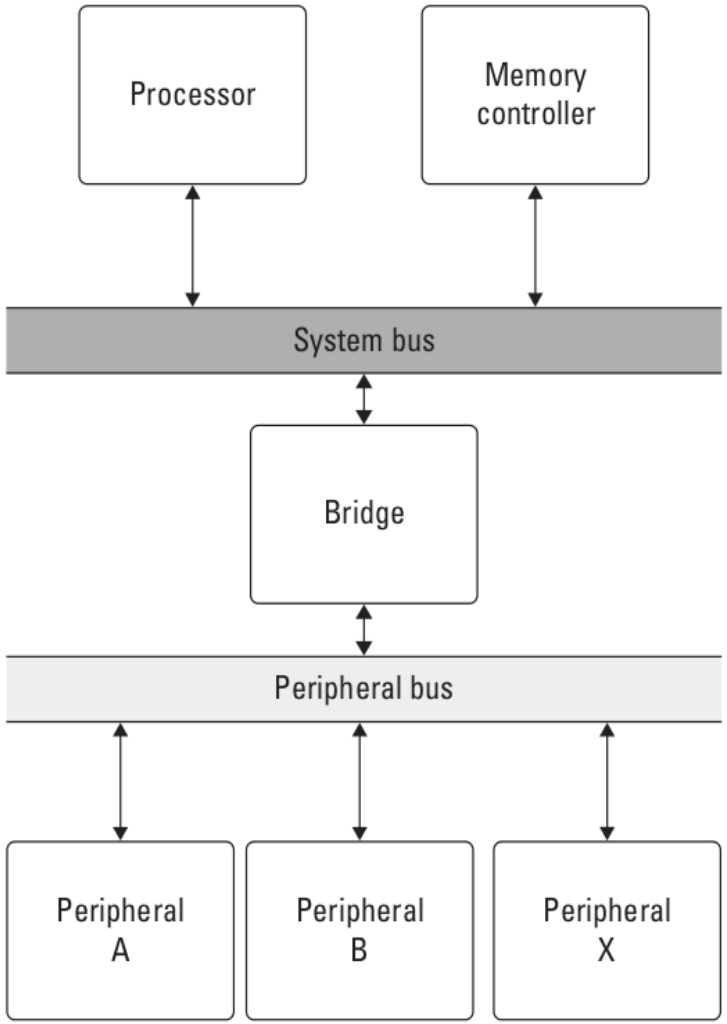
Guión del Tema

- Anatomía de Un Sistema Empotrado
- **Arquitectura Hardware de un SE**
- Arquitectura Software de un SE
- Desarrollando un Sistema Empotrado



Arquitectura Hardware de un SE

- Modelo básico de arquitectura HW del SE centrándonos en el componente principal (**SoC, o microcotrolador)**
  - Tomado de las arquitecturas básicas de PCs: Procesador-Memoria-Buses (de sistema y de periféricos)
  - **SE con microcontrolador o SoC:** arquitectura definida y no modificable
  - **Sobre Plataforma FPGA:** arquitectura hardware adaptable a los requerimientos del sistema
  - Ventaja de la FPGA: Los diseños pueden irse optimizando despues de que el producto este ya en la calle



- Componentes principales de la arquitectura hardware del SE
  - **Procesador (μPs)**:
    - Componente principal que ejecuta el software del SE.
    - Hoy en dia se evoluciona en arquitecturas de mas de un core de procesador (SMPs)
    - **μPs sobre FPGA:**
      - Hard Core: Implementado sobre la FPGA
      - Soft Core: Sintetizable (RTL). Mas flexibilidad a la hora de implementarlo en diferentes FPGAs. Posibilidad de adaptarlo según requerimientos. Generalmente, peor rendimiento que los hard core (5x – 10x perdida de MIPS)
      - Con soporte del Fabricante de FPGA: existe un IDE del micro/SoC del fabricante que facilita la construcción del micro/SoC. Se emplean cores propietarios generalmente para los μPs.
      - Sin soporte del fabricante: Son soft core sintetizables, implementados en LDH (VHDL, Verilog.....)

- **Memoria:**
  - La estructura y organización de la memoria es, quizás, lo que mas lo diferencia un PC desktop de un **SE**
  - La capacidad total de almacenamiento en los SE es uno de los recursos mas restrictivos cuando lo comparamos con otros computadores
  - Sobre la memoria el diseñador del SE tiene que tener claro una serie de conceptos:
    - De cuanta memoria on-chip y cuanta off-chip se dispone o se necesita
    - Cuanta memoria es volátil y cuanto no-volátil
    - Disponibilidad de los controladores de memoria off-chip
    - Interconexión del procesador y periféricos con la memoria para un uso eficiente de la misma

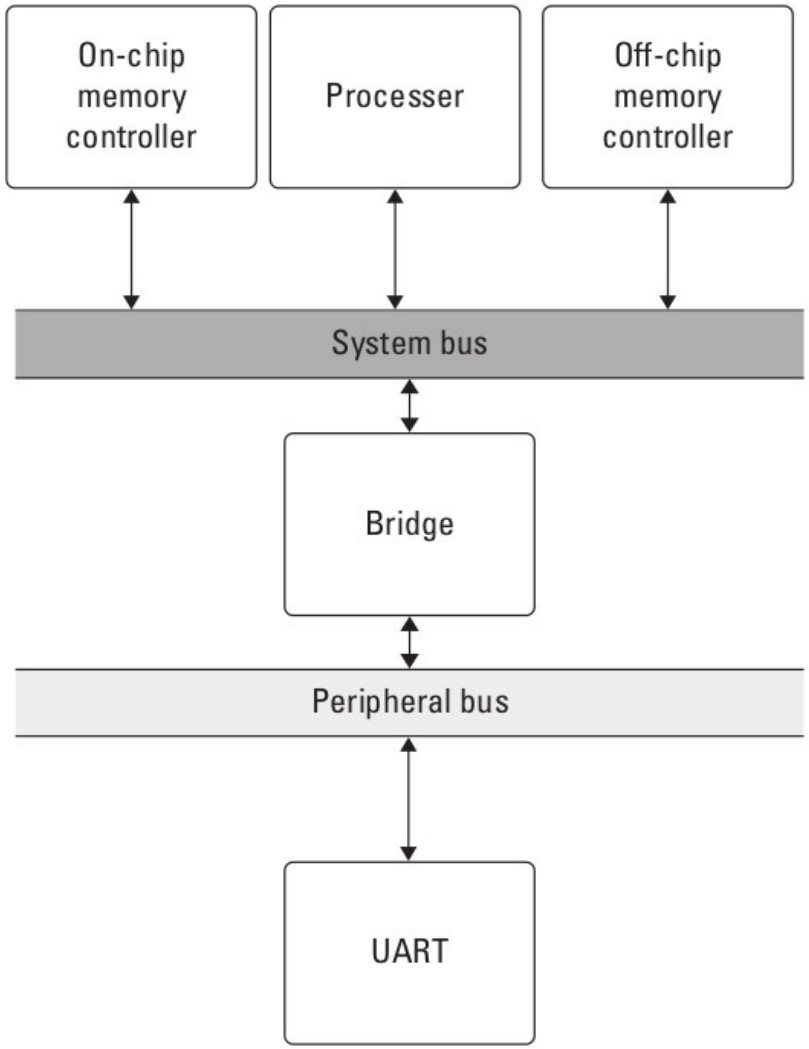
- **Memoria de los SE sobre FPGAs:**
  - On-Chip:
    - Blocks RAMS: La capacidad on-chip depende de la FPGA concreta. Ej. En la familia virtex5 varia entre 100Kb y 16000Kb. Son memorias síncronas Se emplean como memorias cache del SE y como memorias ROM
  - Off-chip:
    - No-Volatil:
      - EEPROM, FLASH: controladores disponibles
      - Tarjetas Compact Flash o SD
    - Volatil:
      - SRAM: controladores fácilmente disponibles
      - DDR: dificultad de disponer de los controladores

- **Buses:**
  - La interconexión entre componentes de un sistema empotrado no suele hacerse con una interconexión punto a punto sino empleando un bus. Así cada componente se interconecta al bus a través de la interfaz del bus.
  - Existen múltiples buses. Cada bus tiene su propia interfaz de bus. De manera común, una interfaz de bus implica señales de dirección, de datos, de control, así como un componente principal denominado arbitrador que se encarga de controlar el uso del bus.
  - Cuando un componente (core) necesita comunicarse con otro, solicita a través de la señal de petición el uso del bus. El arbitrador es el que concede o no el uso del bus.
  - Si un componente tiene la capacidad de solicitar el bus se le denomina Maestro del bus (bus master). No todos los componentes tiene que poder solicitar el uso del bus, sino en muchos casos solo responder a las peticiones de comunicación que les hagan. En este caso se les denomina esclavos del bus (bus slaves)

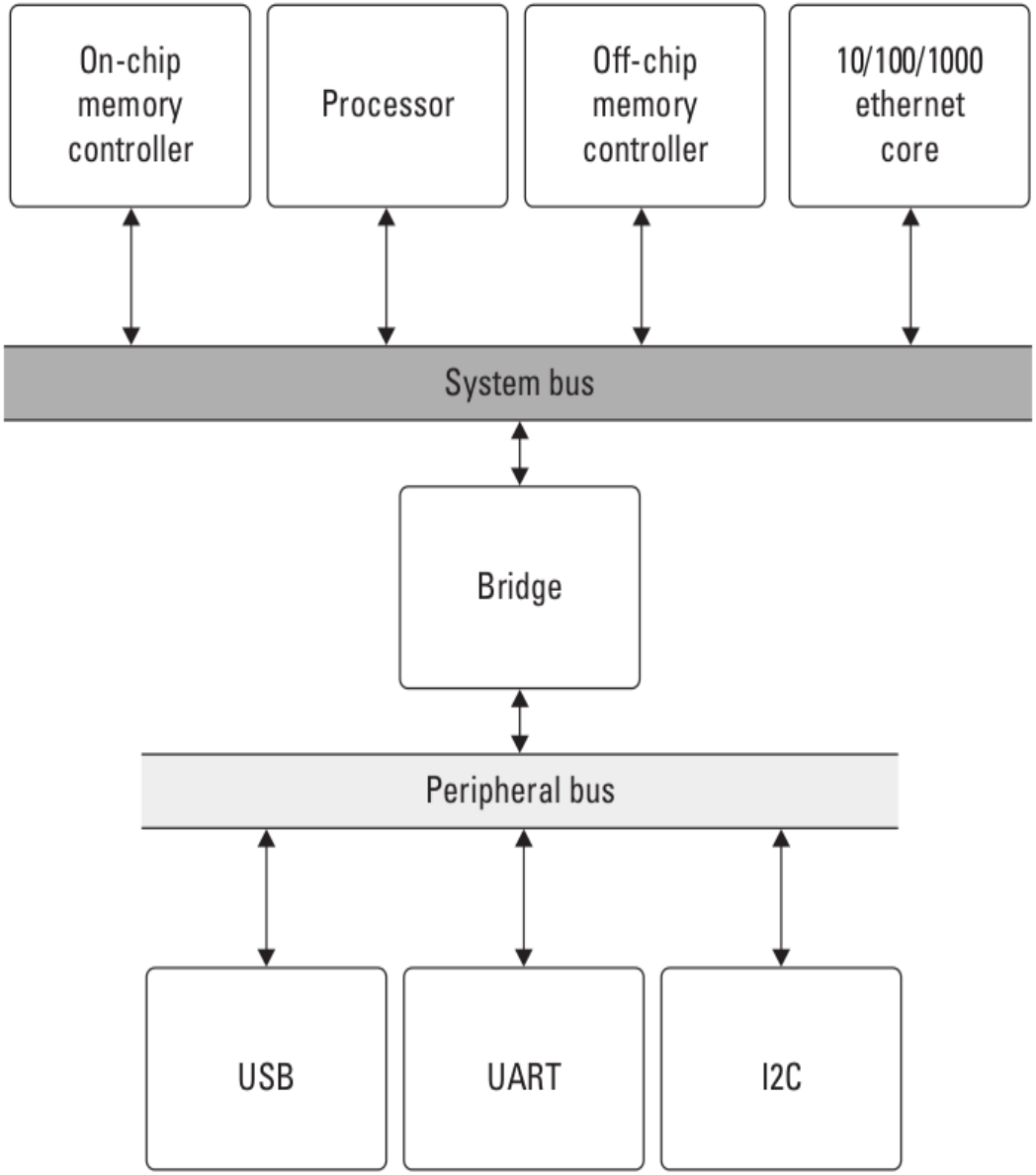
- **Buses:**
  - A la hora de conectar un componente a un bus es importante conocer:
    - Con cuantos otros cores se comunica
    - Con cuales establece una mayor comunicación
    - Ancho de banda que necesita en la comunicación
  - Es habitual disponer de dos buses en el sistema: Un bus rápido (System Bus- suelen estar los componentes principales μp, mem. etc), y un bus más lento (Peripheral Bus)
  - Los buses se interconectan entre si a través de componentes puentes (bridges). Un puente es un componente que propaga las peticiones de un bus a otro. Suele operar a través de una petición de master bus en el bus principal (system bus) hacia un bus slave en el bus secundario (peripheral bus)
  - Bus en FPGA: En una FPGA el bus se implementa en la lógica programable.

- **Periféricos:**
  - Un diseñador del sistema cuando habla de periféricos hace referencia a los componentes de interfaz entre el sistema (a través de los buses) y los periféricos propiamente dichos (leds, switches, monitor, impresora, teclado, ratón etc). Es decir, son los **cores o controladores** de los periféricos en si.
  - **Periféricos sobre FPGA:** Un de las verntajas principales de implementar un SoC sobre FPGA es que puedes incluir los perífericos que realmente necesites adaptando el SoC a las necesidades específicas del sistema
  - La mayoría de los cores de perífericos son reutilizables en diferentes sistemas empotrados. En plataformas FPGAs es habitual disponer de un catalogo o librería de cores con un amplio número de los mismos. Generalmente estos componentes están disponibles como IPs (Intellectuall Property) mediante códigos “software” de descripción del comportamiento hardware (habitualmente diseñados en HDLs)

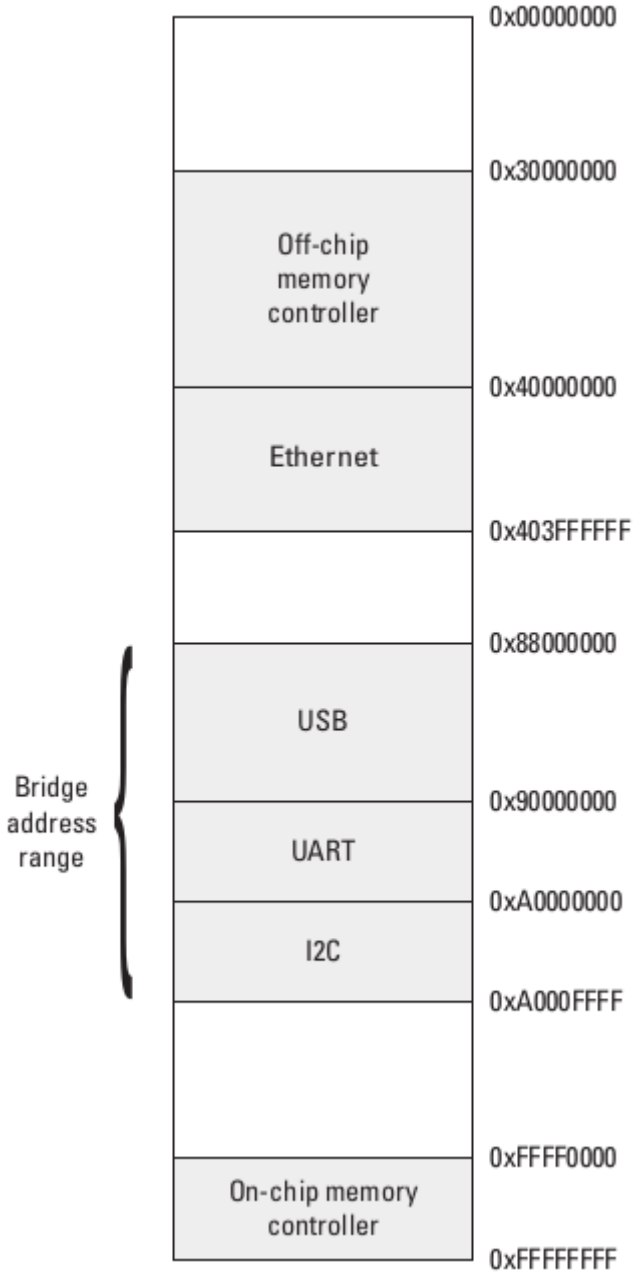
- **Sistema base o de referencia mínimo:**
  - Un sistema base mínimo para un procesador consta de al menos los siguientes componentes:
    - Core del mp, controlador de memoria on-chip y controlador de memoria off-chip, bus principal (system bus), bus de periféricos, bridge entre buses, periférico UART para interconexión desde un Host (PC) con el Sistema empotrado.



- Modificando un sistema base para adaptarlo a los requerimientos:
  - Adaptar el sistema base a los requerimientos significa añadir componentes o cores que se necesitan para la funcionalidad del sistema.
  - En la figura se observa un ejemplo en el que, además de los componentes del sistema base se han añadido: controlador ethernet -para interconexión via internet, controlador USB, I2C
  - Si un core esta adaptado al bus que se esta empleando es relativamente fácil añadir o eliminar nuevos cores al sistema



- Mapa de direcciones del SE
  - A la hora de diseñar un Sistema empotrado es fundamental el mapa de direcciones que resuelva como se puede acceder a cada uno de los componentes a través del bus.
  - Las zonas de memoria que se reservan para los diferentes componentes tienen que estar acordes con las direcciones físicas de cada uno de ellos.



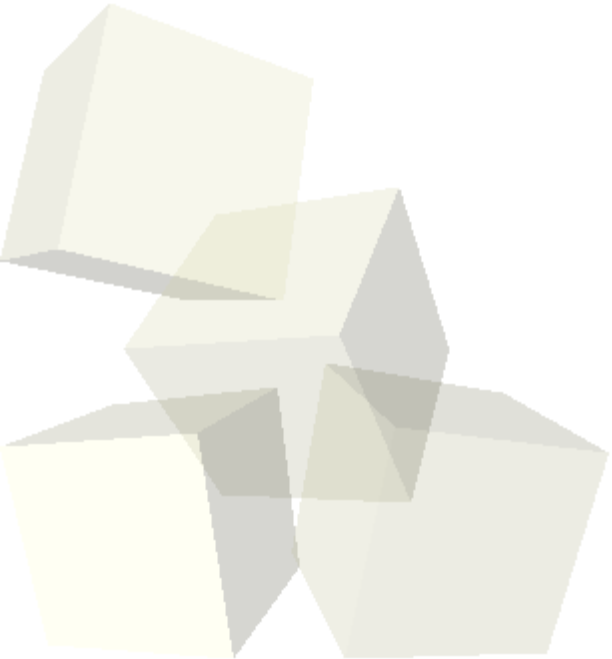
- **SoC sobre FPGA**
  - Característica principal frente a microcontroladores o SoC implementados: Posibilidad de añadir un core personalizado para el SE.
  - Ventajas de diseñar en hardware frente a la solución software:
    - Velocidad. Diseñar en hardware no siempre significa ser mas rápido que la solución software. Se dice que un diseño hw en FPGA es entre 5x a 10x veces mas lento que la implementación equivalente sobre ASIC.
    - Sin embargo, eligiendo bien el modelo de implementación hardware y centrándonos en aquellas partes criticas del sistema que suponga un alto grado de especialización la velocidad final del sistema suele mejorarse significativamente

Arquitectura Hardware de un SE: SoC sobre FPGA

- Ventajas de diseñar en hardware frente a la solución software:
  - Eficiencia: Un diseño hardware especifico puede ser mas eficiente desde, el punto de vista de los recursos, que buscar siempre la solución software que, como mínimo requerirá de un procesador, buses y memoria
  - Predictibilidad: las funciones implementadas en hardware son muchos mas predecibles en cuanto al tiempo de ejecución y finalización que las soluciones software, sobre todo en el caso de sistemas con sistemas operativos. Así para sistemas en tiempo real donde es crítico conocer la finalización de una tarea una gran solución es implementarlas en hardware.
- Desventaja: Mucho mayor esfuerzo de desarrollo del sistema

Guión del Tema

- Anatomía de Un Sistema Empotrado
- Arquitectura Hardware de un SE
- **Arquitectura Software de un SE**
- Desarrollando un Sistema Empotrado



Arquitecturas softwares de un S.E.

- Modo standalone

- Se emplean librerías, cabeceras, etc. que configuran el μp y los periféricos del SE
- En Plataforma FPGAs:
  - Es habitual usar el modo stand-alone para realizar un test del sistema hardware. Se suelen desarrollar test de los periféricos que se ejecutan por separado
  - Los códigos ejecutables suelen ocupar poco espacio en memoria por lo que se pueden carga en la memoria on-chip (block-rams de la FPGA)
  - Existen librerías específicas preparadas para trabajar en el modo standalone. Ej. Newlib for openrisc

- Modo Software de Sistema (Sistema Operativo)
  - Implantar un S.O. completo sobre una arquitectura hardware:
    - Ventaja: Proveen multitud de servicios fundamentales al diseñador del software
    - Inconveniente: Supone un alto coste desde el punto de vista de los recursos hardware (memoria, ciclos de procesador, potencia, etc)
    - Técnica habitual: implantar S.O. no completos (Software de sistema) en función de las necesidades y de los recursos disponibles
    - Existen un rango grande de alternativas de software de sistema que va desde las librerías básicas del modo stand-alone (Bare metal mode) hasta un S.O. completo

- Modos Software de Sistema
  - Modo de ejecución de múltiples hilos: Librería multihilos
    - Permite crear, ejecutar y destruir múltiples hilos de ejecución sobre un procesador Existen diferentes alternativas mas o menos complejas.
      - Ecos, Xilkernel, nucleos O.S., etc.
  - Modo Sistema operativo con restricciones: ej. MicroCLinux (μCLinux)
    - Basado en Kernel de Linux 2.0. Pretende dar soporte a procesadores sin MMU (no hay espacio virtual de memoria - -no hay protección de memoria-)
  - Full-featured operating system
    - Tiene un sobre coste de recursos muy alto:
      - MMU; Huella de memoria del OS muy grande (implica mucha memoria en el SE); muchos procesos propios del OS ejecutándose simultáneamente con la aplicación
      - Requieren de un Sistema de fichero para almacenamiento

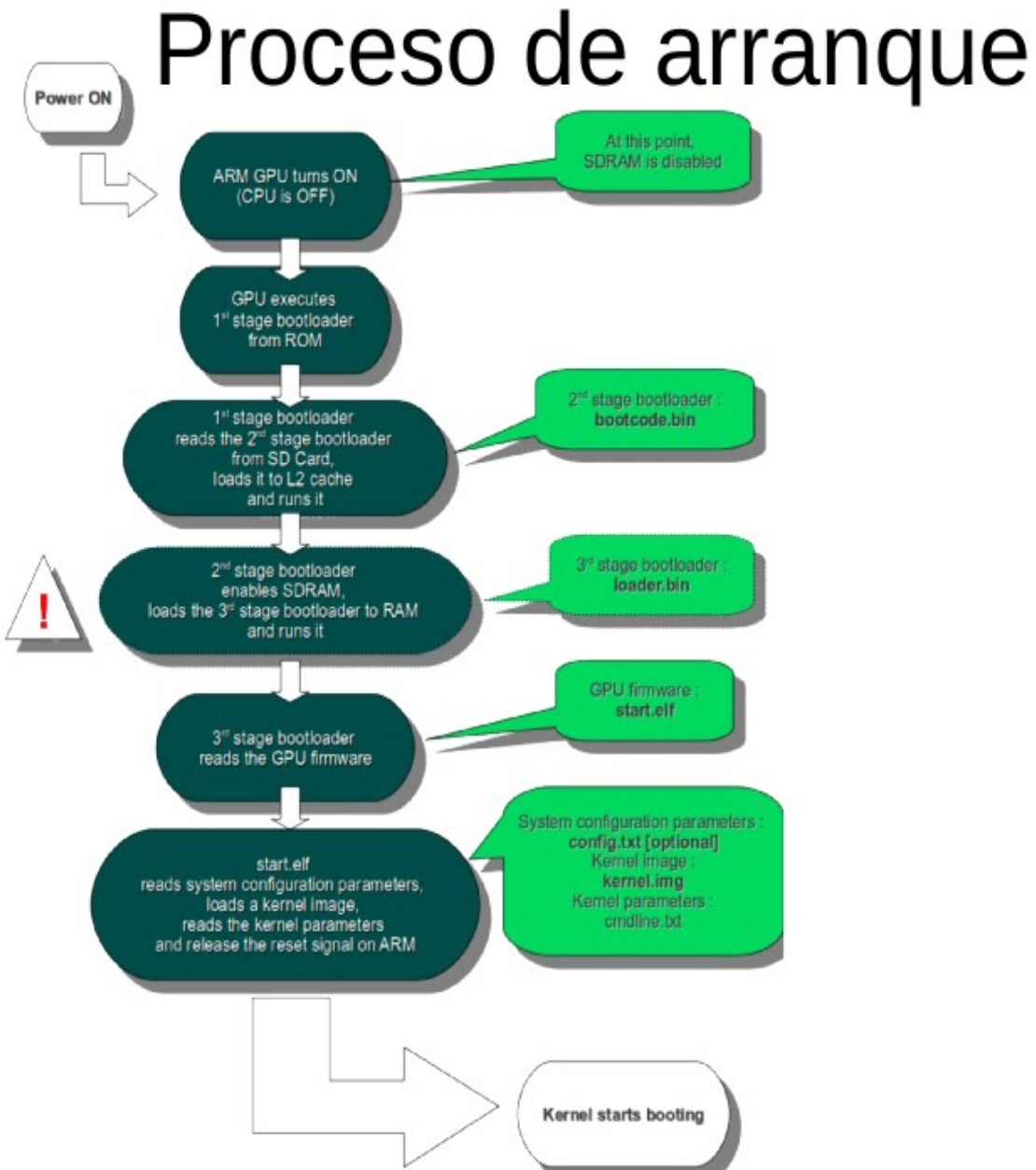
- Características principales del modo de sistema operativo completo
  - Hasta los últimos años no ha sido posible emplear un Full-featured O.S. en los SE
  - La tendencia en el diseño de SE es emplear siempre que sea posible un S.O. completo:
    - Los desarrolladores de software no necesitan conocimientos especiales respecto del desarrollo de software en un PC desktop
    - Existen un catalogo enorme de software disponible
    - Debido a la gran extensión de uso de los SE es necesario poder disponer de mucho software para no tener que estar constantemente desarrollando las aplicaciones
  - Característica fundamental de los SO:
    - Necesitan un **Sistema de Ficheros** para la organización y almacenamiento de datos ==> Dispositivos físicos de memoria que guarden dichos datos.

- Sistema de ficheros: Estructura de datos implementada en un dispositivo hardware de almacenamiento que se caracteriza por tener una colección de bloques de memoria de igual tamaño en los que se pueden crear, leer y escribir ficheros de tamaño variable
  - El sistema de ficheros es el nivel que existe entre el dispositivo físico de almacenamiento de datos y el sistema operativo que necesita tener estructurados y organizados los datos de algún modo.
- Sistema de ficheros Raíz (Root Filesystem)
  - UNIX y sus variantes comparten el concepto de root filesystem
  - En UNIX los datos se organizan entorno a una estructura jerárquica de directorios (carpetas) y ficheros dentro de las carpetas.
  - Primer nivel de la jerarquía es el root filesytem (/)

- Arranque de SE con Sistema operativo completo
  - Para poder acceder a un sistema de ficheros cualquiera, es necesario que este cargado en el sistema el driver (módulo) correspondiente.
  - Esto implica que el proceso de arranque de un sistema empotrado con un sistema operativo no es algo trivial
  - Generalmente implica varias fases con diferentes programas Bootloaders (cargadores de software)

- De una manera resumido podemos considerar las siguientes fases:
  - Fase 1, bootloader básico: software o bootloader básico almacenado en memoria (rom) dentro del propio SoC. Su misión es ser capaz de acceder al dispositivo de memoria donde se encuentra el siguiente bootloader
  - Fase 2, bootloader complejo: Este bootloader es capaz de acceder a las partes principales del software y de montar incluso sistemas de ficheros básicos, FAT, ext2.. Generalmente, accederá al kernel del núcleo del sistema operativo, lo cargará en memoria RAM y le pasará el control
  - Fase 3, ejecución kernel: El kernel deberá contener los drivers (módulos) de todos los componentes hardware del sistema para hacerlos accesibles al mismo y montará el sistema de ficheros raíz que se encontrara en algún dispositivo de memoria tipo FLASH (embedded flash, o tarjeta de memoria..)

Ejemplo de proceso de arranque de sistema empotrado con SO Linux: Raspberry PI http://wiki.beyondlogic.org/index.php?title=Understanding_RaspberryPi_Boot_Process

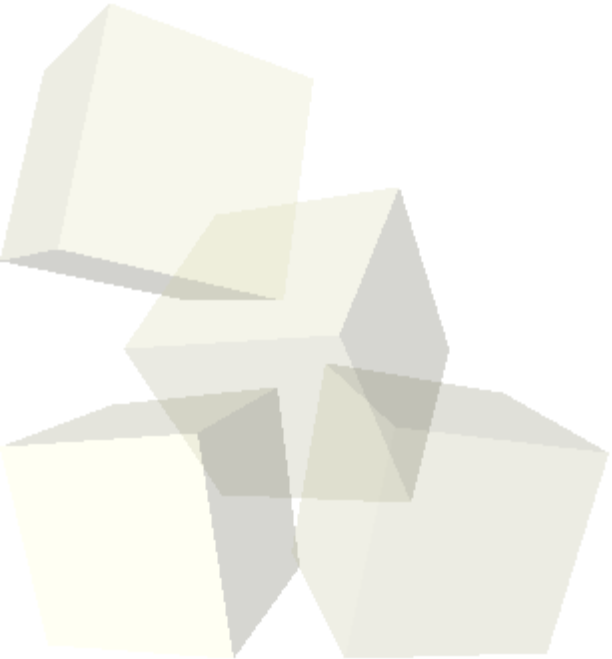


- Desktop computer (PC) vs Embedded System (SE)
  - PC: BIOS (Basic Input/Output Software)
    - Conjunto complejo de rutinas de software para la inicialización de un computador. Suele almacenarse en memoria Flash para permitir su actualización.
    - La mayoría de los usuarios de DC no son conscientes de las BIOS y su funcionalidad, incluidos los desarrolladores de aplicaciones para DC
  - SE: Bootloader
    - Es un software que realiza tareas similares a las de las BIOS en un SE. Sin embargo, por la escasez de recursos de los ES este software tiene que estar muy optimizado
    - Debido a que la arquitectura hardware de cada SE es diferente, el diseñador del SE tiene que adaptar o desarrollar un bootloader adecuado a su placa de desarrollo.

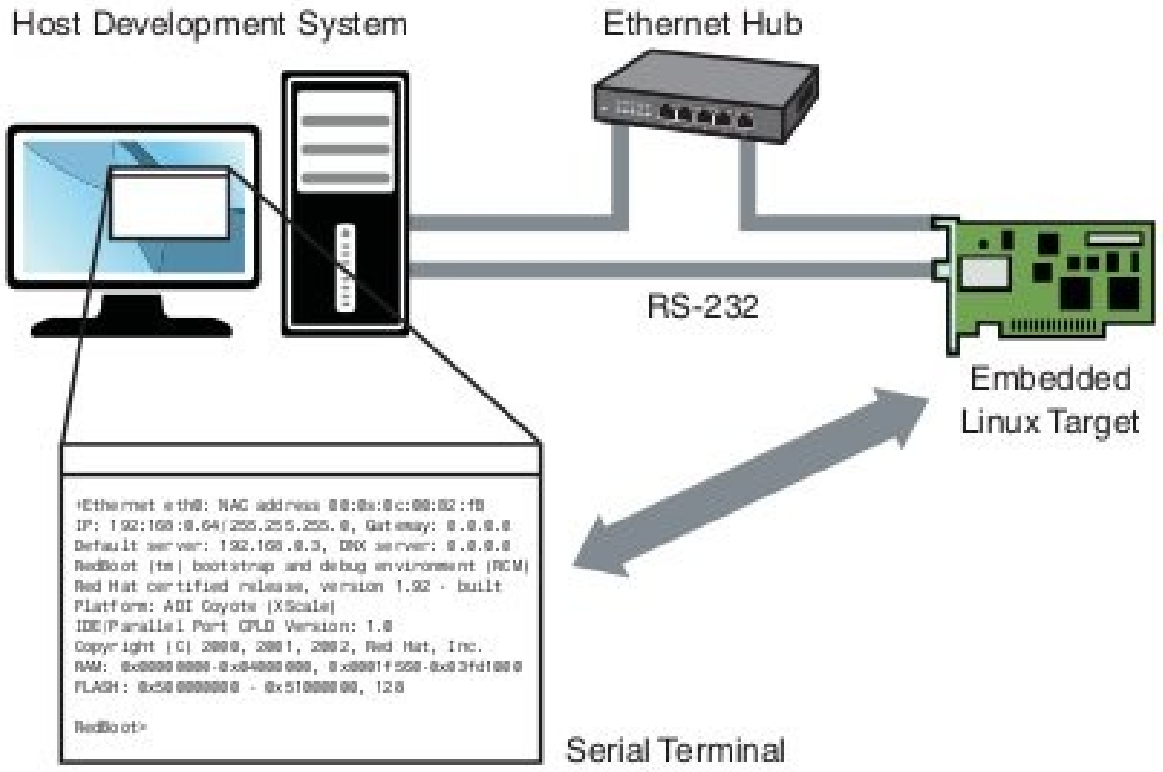
- BOOTLOADER para SE sobre FPGA:
  - Para los Kernel de LINUX el bootloader mayoritariamente usado es U-boot (http://www.denx.de/wiki/U-Boot)
  - U-boot inicializa el hardware de un SE y, además puede realizar otras tareas como transferir ficheros desde ethernet u otras.
  - Como tarea principal, u-boot descomprime la imagen de un kernel de linux y transfiere el control de ejecución al mismo.
  - Ocupa bastante memoria y por ello en SE con FPGAs no se suele cargar en memoria on-chip sino en memoria off-chip (FLASH)
  - En SE con FPGAs suele haber un Pre-bootloader simple cuya misión es transferir la ejecución a u-boot almacenado en FLASH
  - U-boot se suele emplear fundamentalmente en la fase de desarrollo y debug del sistema empotrado

Guión del Tema

- Anatomía de Un Sistema Empotrado
- Arquitectura Hardware de un SE
- Arquitectura Software de un SE
- **Desarrollando un Sistema Empotrado**



- Sistema empotrado sobre FPGA
  - Configuración típica de desarrollo de un SE

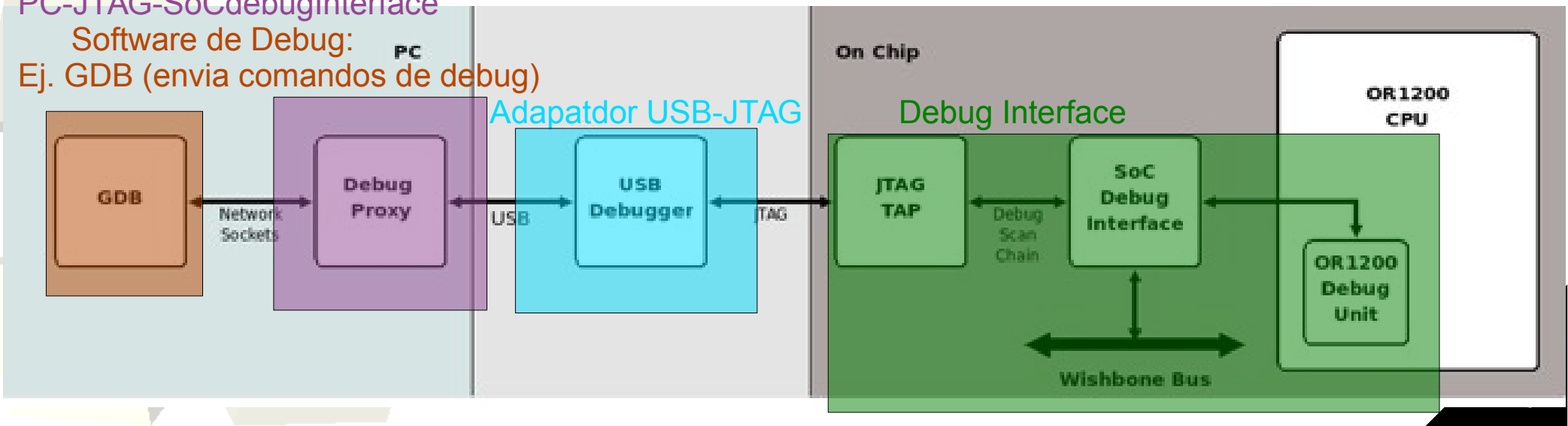


Configuración típica de desarrollo de un SE

- Host development system: Equipo donde están instaladas las herramientas necesarias para el desarrollo del hardware y software del SE
- Placa de desarrollo: hardware básico que, en el caso de SE sobre FPGAs incluye la FPGA asi como los periféricos que necesita el SE
- Interfaces de conexión del SE con el Host:
  - Terminal serie- conexión RS232: Se emplea para comunicar el sistema empotrado con el host y recibir información acerca del mismo y de sus estado de funcionamiento.
  - Interfaz de debugger: interfaz que permite enviar software al SE, ejecutarlo, hacer el debug.... Interfaz de debugger mas conocidas: **JTAG, SWD**

- Sistema de Debug en SE: JTAG
  - En la interconexión Host-SE mediante JTAG intervienen diversos componentes hardware y también software
  - **HARDWARE**:
    - Adaptador PC-JTAG (USB-JTAG)
    - Interface de debug on-chip: JTAG Tap y Unidad de debug del μPs
  - **SOFTWARE:**
    - Soft de comunicación: USB con chip JTAG y JTAG con Unidad de debug
    - Soft de debug: Envia los comandos de debug a traves del soft de comunicación

Software de comunicación:
PC-JTAG-SoCdebugInterface
Software de Debug:
Ej. GDB (envia comandos de debug)



Desarrollo hardware

- Podemos distinguir dos niveles diferentes de desarrollo de hardware:
  - Modificando el SOC (solo realizable en el caso de trabajar sobre FPGAs)
    - Se necesita tener disponible en el Host:
      - Herramientas de síntesis y programación del vendor de la FPGA
      - Sistema base del SE sobre el que se pretende trabajar
      - Catalogo de Cores disponibles para añadir al SE
  - Añadiendo Hardware al SE: Desarrollando alguna placa de expansión para conectar algún hardware en el algún puerto del SoC

- **Desarrollo software**
  - Se necesita tener disponible en el Host:
    - Herramientas de Compilación cruzada (Cross compiler)
      - Un compilador cruzado es un traductor de lenguaje de alto nivel a código ejecutable que se ejecuta en una plataforma pero que genera código de otra plataforma diferente
      - Una plataforma implica: Una arquitectura de procesador, una librería básica generalmente de C (C library) y un sistema operativo
      - TOOLCHAIN: conjunto de herramientas que permiten realizar la compilación cruzada
        - Compilador, linker, librerias, debbuger