EE382V: System-on-a-Chip (SoC) Design

Lecture 0 – Class Overview

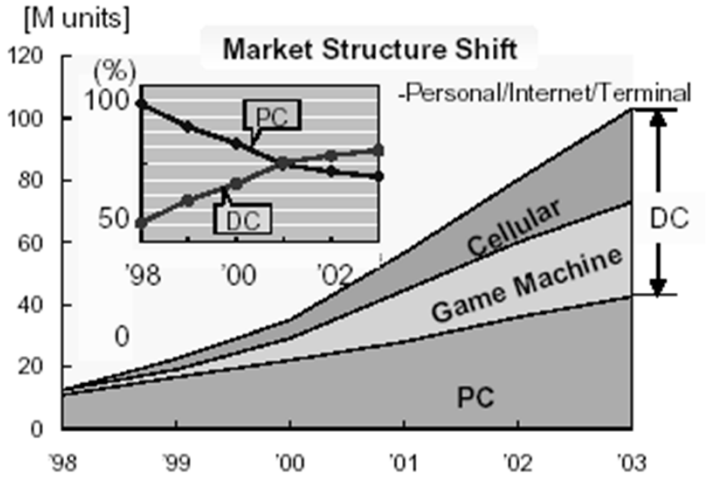
Andreas Gerstlauer
Electrical and Computer Engineering
University of Texas at Austin
gerstl@ece.utexas.edu

Lecture 0: Outline

- **Introduction**
  - Systems-on-Chip (SoCs)
  - Design flow
- **Course information**
  - Overview, goals, topics
  - Administration
  - Labs and project
- **Software-defined radio class project**
  - DRM software receiver
  - DRM SoC implementation

Industrial Structure Shift

- **Ubiquitous, embedded computing**
  - From personal to dedicated computers

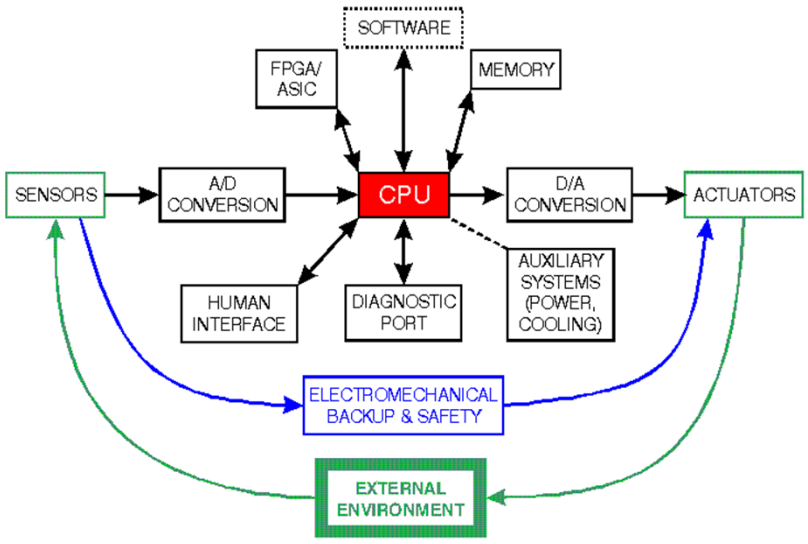


Source: SONY Corp & Market Estimates

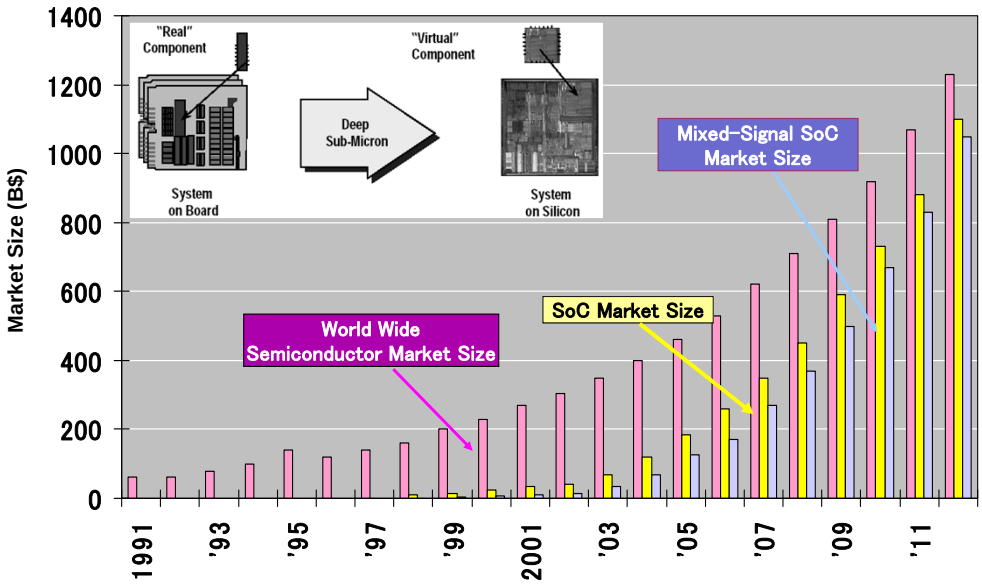
Embedded Systems

- **Ubiquitous computing**
  - Signal processing systems
    - Radar, sonar, real-time video, set-top boxes, DVD players, medical equipment
  - Mission critical systems
    - Avionics, space-craft control, nuclear plant control
  - Distributed control
    - Network routers, mass transit systems, elevators in large buildings, sensors
  - "Small" systems
    - Cellphones, appliances, toys, MP3, PDAs, digital cameras, smart badges
- **Part of a larger system - masquerading as non-computers**
  - Not a "computer with keyboard, display, etc."
  - Interact (sense, manipulate, communicate) with external world
  - Never terminate (ideally)
- **Application-specific function – not general-purpose**
  - Application is known a priori, but some re-programmability
- **Constrained**
  - Timing: latency, throughput (real time)
  - Power, size, weight, heat, reliability, etc.

(Embedded) Computer System Architecture

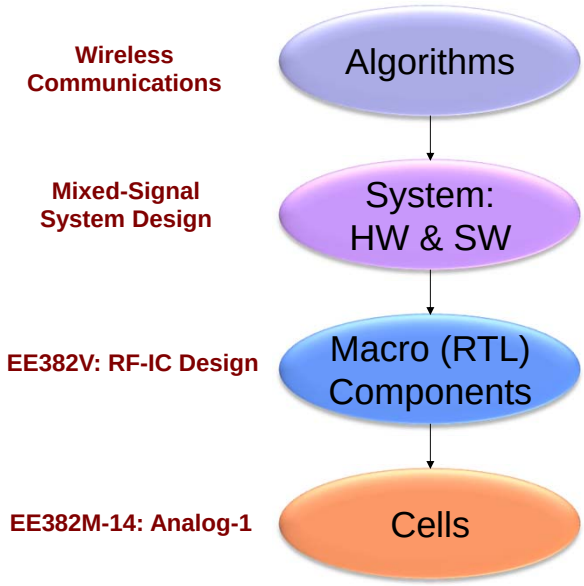


System-on-Chip (SoC) Era

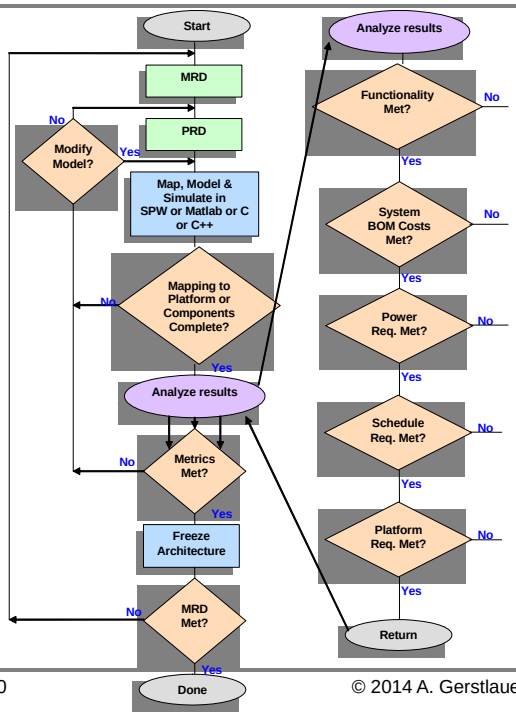


Source: SONY Corp & Market Estimates

Integrated Circuits and Systems at UT



SoC Design Flow



Course Overview

- **Provide an understanding of the concepts, issues, and process of designing highly integrated SoCs**
  - Systematic hardware/software co-design & co-verification
  - Industry-standard/advanced SoC design flow

➢ **Class labs and project: Software-defined radio SoC**
  - DRM (Digital Radio Mondiale) system
    - Hardware/software co-design
  - State-of-the-art synthesis and verification tools
    - Virtual platform prototyping and simulation in SystemC/TLM
    - High-level hardware synthesis from C++ to RTL
  - ARM-based FPGA prototyping platform
    - ARM processor, I/O devices, memory components, hardware accelerators

Course Goals

- **Course is designed to learn about:**
  - Early functional and nonfunctional performance analysis to support design decisions.
  - Analysis and optimization of hardware/software tradeoffs, algorithms, and architectures based on requirements and implementation constraints.
  - Architectures for control-dominated and data-dominated systems and real-time systems.
  - Hardware, software, and interface synthesis.
  - Interface design.
  - Co-simulation to validate system functionality.
  - Examples of applications and systems developed using a co-design approach.
  - Intellectual property, reuse, and verification issues.

Course Topics

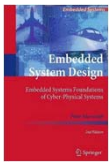
- **Covered in class:**
  - System-level and SoC design methodologies and tools;
  - HW/SW Co-design: analysis, partitioning, real-time scheduling, hardware acceleration;
  - Virtual platform models, co-simulation and FPGAs for prototyping of HW/SW systems;
  - Transaction-Level Modeling (TLM) and Electronic System-Level (ESL) languages: SystemC;
  - High-Level Synthesis (HLS): allocation, scheduling, binding, resource sharing, pipelining;
  - SoC and IP integration, verification and test.

Course Administration

- **Instructors**
  - Instructor: Andreas Gerstlauer <gerstl@ece.utexas.edu>
    - Office hours: POB 6.118, TW 2-3pm, or after class/by appointment
  - Guest lecturers from industry and academia (tentative):
    - David Black (Doulos), Mark McDermott, Steven Smith
  - TA: Seogoo Lee <sglee@cerc.utexas.edu>
    - Office hours: TBD
- **Information**
  - Web: http://www.ece.utexas.edu/~gerstl/ee382v_f14
    - Lecture notes, homework and lab exercises.
  - Canvas
    - Announcements, assignments, grades.
- **Dates (tentative)**
  - Exam: November 6, in class
  - Project design reviews: December 2 & 4, in class
  - Final project presentations: December 16, 9:00am-noon

Textbooks

- **No required textbook**
- **Suggested references**
  - P. Marwedel, *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems*, 2nd edition, Springer, 2011
    - Slides, etc.: http://ls12-www.cs.tu-dortmund.de/~marwedel/es-book/
  - D. Black, J. Donovan, B. Bunton, A. Keist, *SystemC: From the Ground Up*, 2nd edition, Springer, 2010
    - On-demand print option available
  - G. De Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, 1994
    - Out of print, but can be ordered
    - On reserve in library



Course Policies

- **Grading:**
  - Homework: 15%
  - Lab assignments: 30%
  - Exam: 20%
  - Project: 35%
- **Late penalties:**
  - 20% per day (24 hours)
- **Academic dishonesty**
  - Homeworks
    - Ok to discuss questions and problems with others
    - But turn in own, independent solution
  - Labs and project
    - In teams, one report and presentation
    - Collaboration encouraged and desired
  - Plagiarism
    - Use of any outside source of information without quoting or referencing is cheating

Labs (tentative)

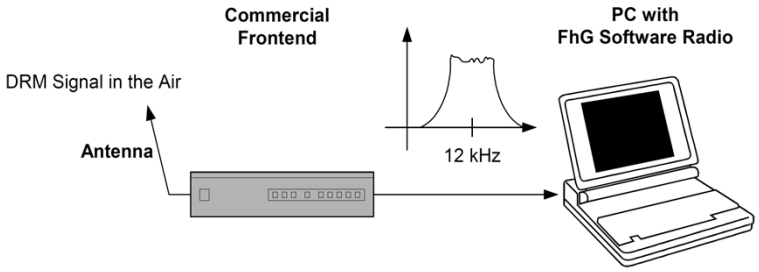
- **Lab 1: Software/algorithm (3 weeks, due Sep 22)**
  - Profiling of code on Linux host and ARM simulator
    - Identify time consuming bottlenecks to optimize
  - Software optimization
    - Floating-point to fixed-point conversion
    - Improve performance without loss in accuracy (SNR)
- **Lab 2: System architecture (3 weeks, due Oct 13)**
  - Hardware acceleration
    - Identify and partition code into hardware and software
  - Virtual platform modeling and simulation
    - Isolate and interface hardware as SystemC module
    - Co-simulation w/ software and firmware on ARM-Linux CPU
- **Lab 3: Hardware synthesis (3 weeks, due Nov 3)**
  - High-level synthesis of Viterbi decoder from C++ to RTL
    - Xilinx Vivado synthesis tool
  - Verification of synthesis results
    - Testbench around standalone C++ vs. RTL hardware module

Class Project

- **Implementation on prototyping board**
  - Synthesizing hardware into the FPGA
    - Accelerator modules, at minimum Viterbi decoder
    - Synthesis and download using Xilinx software
  - C/C++ program on the ARM host processor
    - Cross-compilation and download to run under Linux on the ARM
  - Hardware/software interfacing and communication
    - Software drivers and hardware bus interfaces
  - Analysis and validation of product metrics
    - Estimate timing, area and power consumption
- **Low-power SoC implementation of DRM receiver**
  - Reference design of DRM ASIC
    - FPGA-based implementation as prototype of ASIC design
  - To be incorporated into cell phones
    - Satisfy market and product requirements

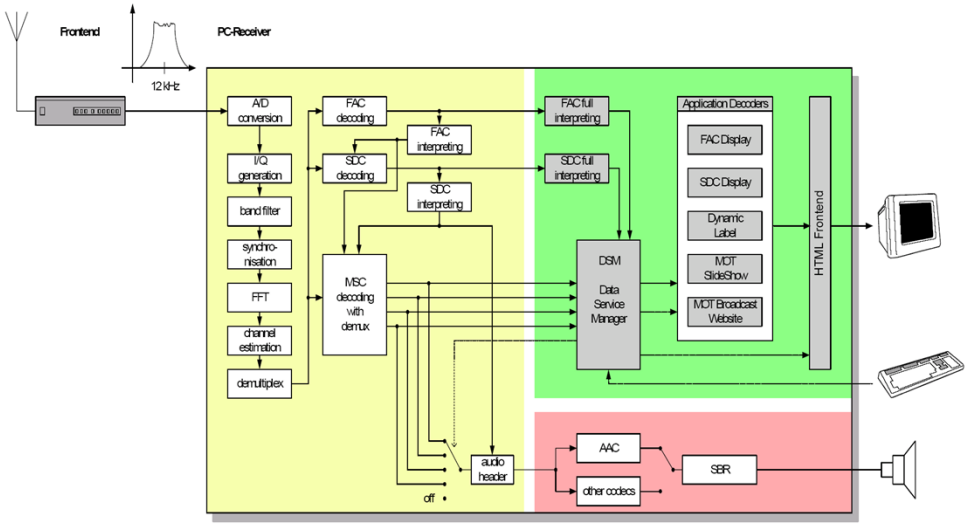
Software-Defined Radio (SDR) Receiver

- **Digital Radio Mondiale (DRM)**
  - World standard for digital broadcasting in the AM radio bands below 30 MHz
    - Based on OFDM modulation used in most modern radio standards
  - DRM-capable software radio developed by Fraunhofer (FhG) Institut für Integrierte Schaltungen
    - Push DRM technology and adoption thereof
    - Early availability and an easy way to reproduce the radio.



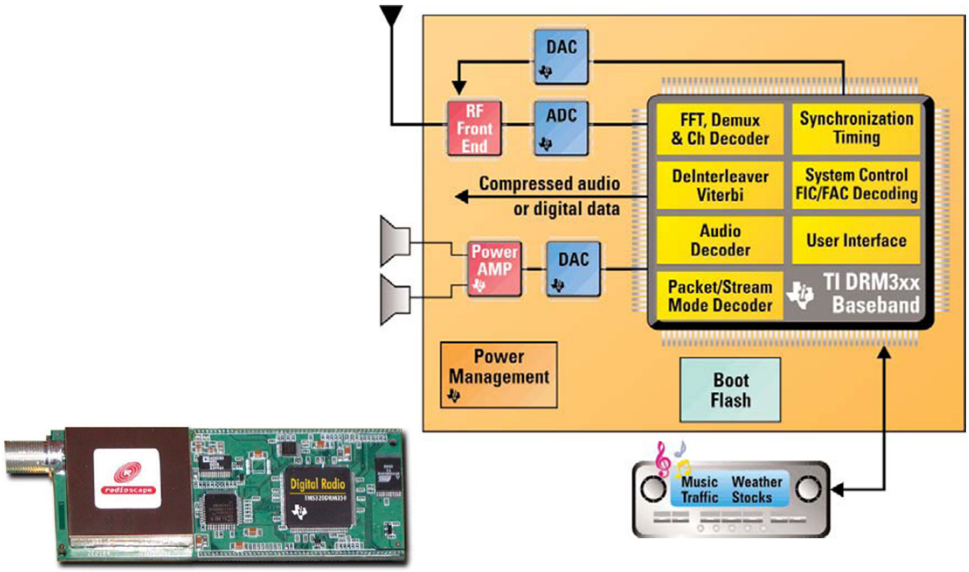
Dream DRM Receiver

- **Software implementation of DRM Receiver**
  - Open-source alternative to commercial FhG software



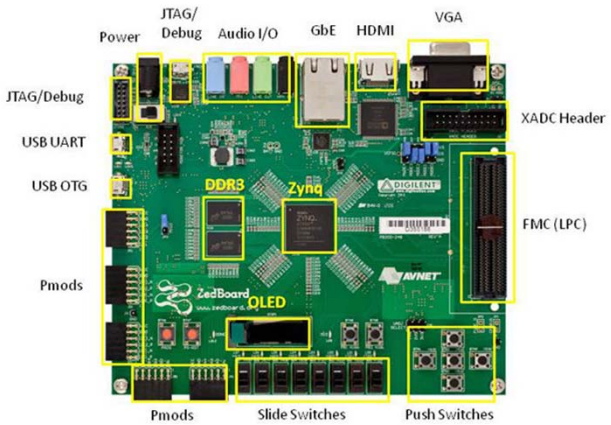
TI DRM Chip

- **Texas Instruments TMS320DRM300/350**



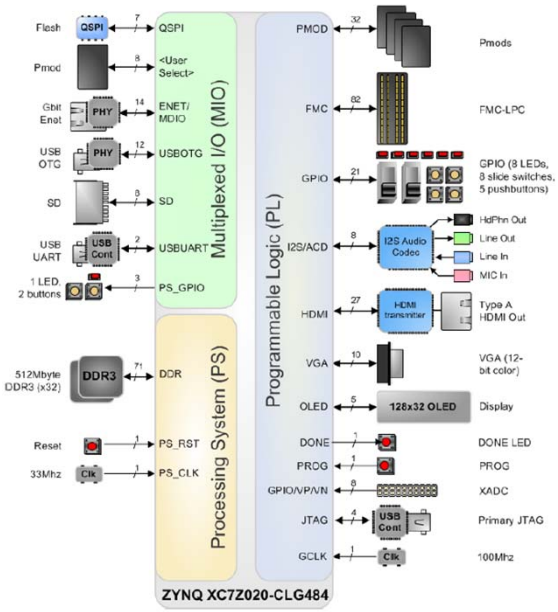
Prototyping Board

- **Zedboard by Digilent**
  - 10 boards in UTA 0.212 (basement) lab



- **Based on Xilinx Zynq-7000 All-Programmable SoC**
  - Dual-core ARM Cortex A9 in Virtex 7 FPGA fabric

Zedboard Block Diagram



Zynq-7000 All-Programmable SoC