ECE382M.20: System-on-Chip (SoC) Design

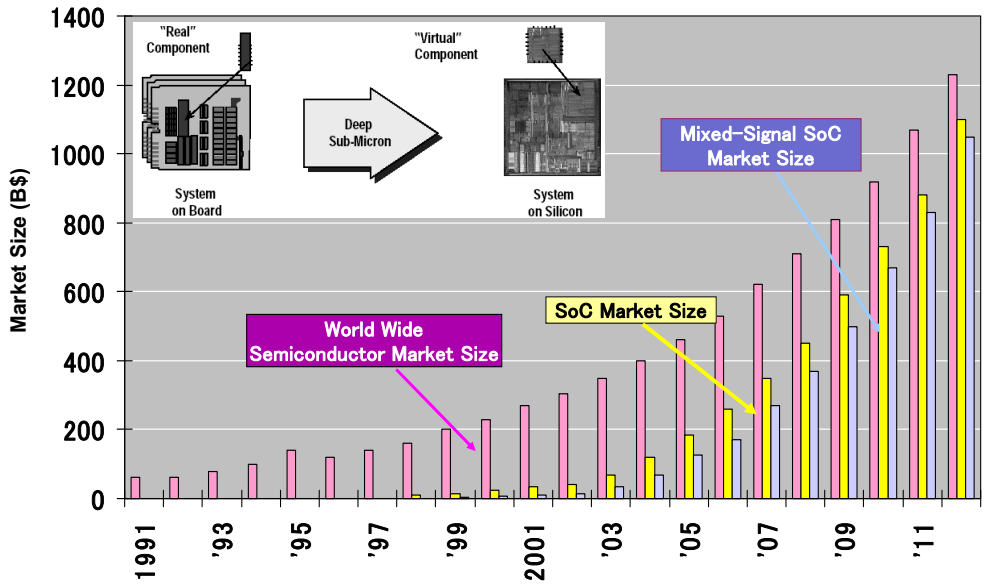
Lecture 0 – Class Overview

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
- **Class project**
  - Deep learning based visual object recognition
  - SoC implementation

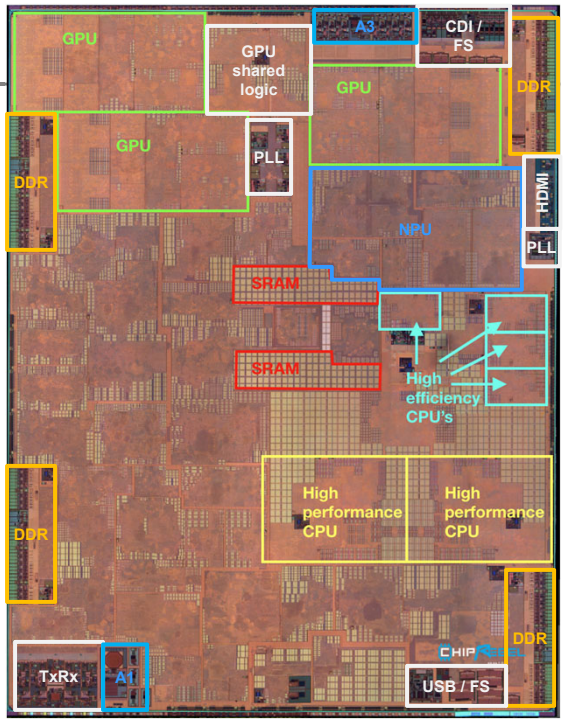
System-on-Chip (SoC) Era



Source: SONY Corp & Market Estimates

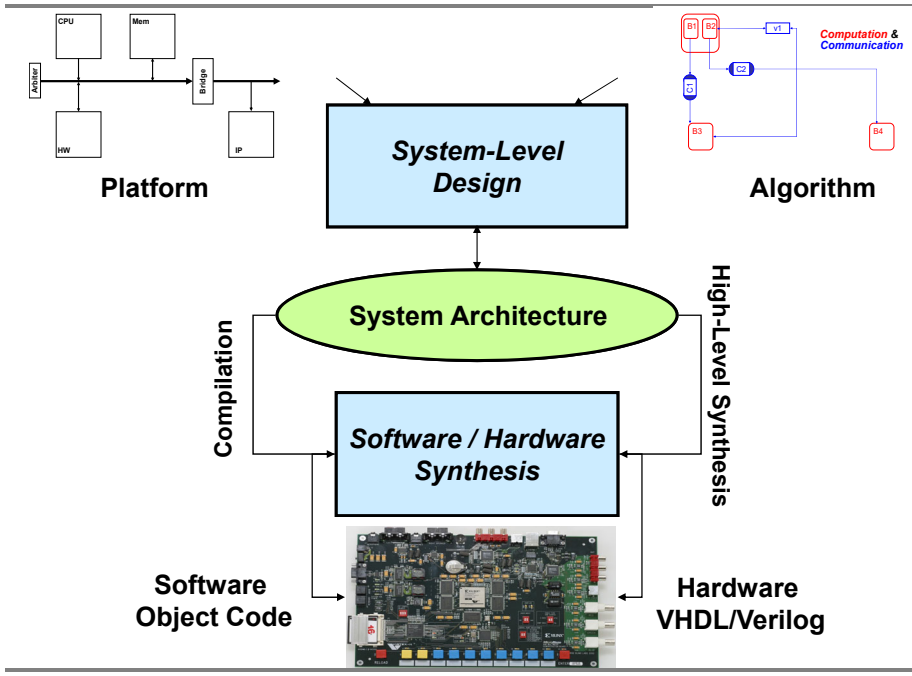
Apple A11 SoC

- **6-core ARMv8**
  - 2 big
  - 4 little
- **3-core GPU**
  - Proprietary
- **Co-processor**
  - M-class ARM
- **Neural processing unit (NPU)**
  - "Neural engine"
- **Accelerators**
  - Lots…

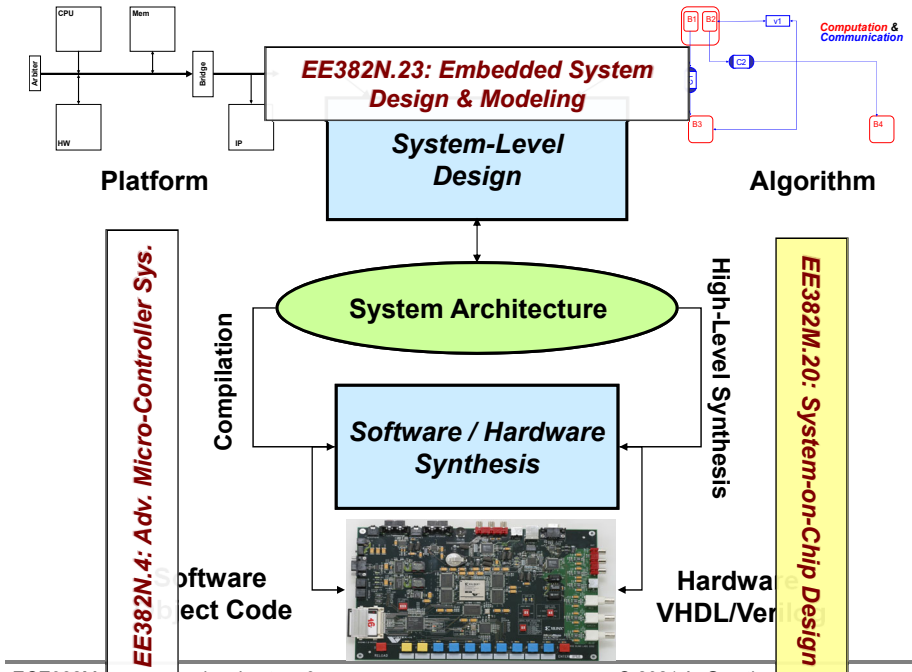


Source: ChipRebel & TechInsights

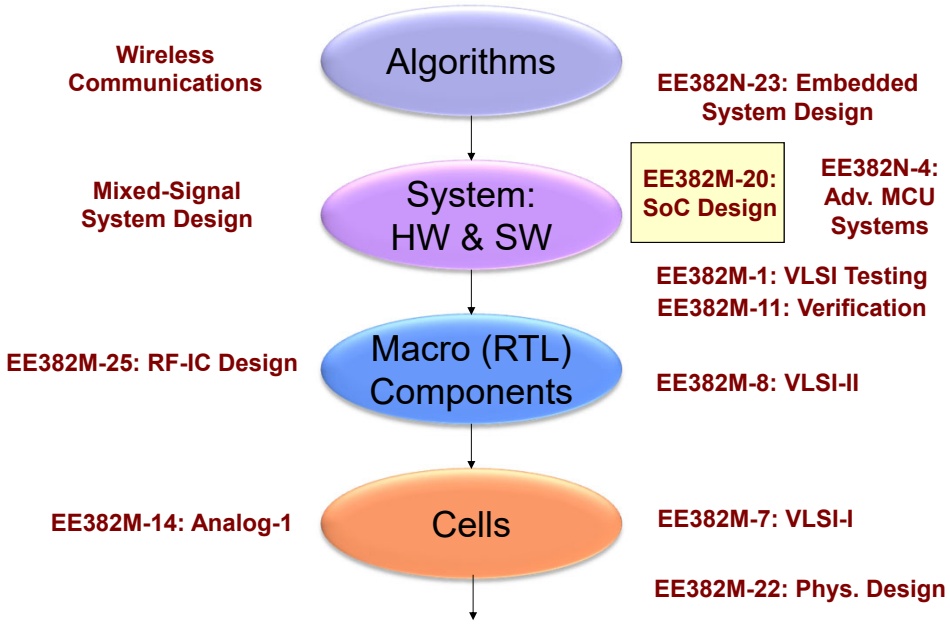
System Design Flow



UT ECE Courses



Integrated Circuits and Systems at UT



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
  - Manual SoC architecture definition + component synthesis

➢ **Class labs and project: visual object recognition SoC**
  - Deep/Convolutional Neural Network (DNN/CNN) inference
    - 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.
  - Analyze tradeoffs and explore architecture and mico-architecture design spaces to develop and synthesize custom hardware accelerators.
  - 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 and Prerequisites

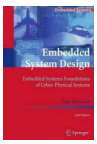
- **Covered in class**
  - System-level and SoC design methodologies and tools;
  - HW/SW co-design: analysis, partitioning, real-time scheduling, hardware acceleration;
  - Virtual prototyping: virtual platform models, electronic system-level languages, and HW/SW co-simulation;
  - High-Level Synthesis (HLS): allocation, scheduling, binding algorithms for C-to-RTL synthesis;
  - SoC integration: SoC communication architectures, IP interfacing, verification and test;
  - FPGA prototyping of HW/SW systems.
- **Prerequisites**
  - Working knowledge of C/C++ software development
  - Digital hardware design and HDLs (VHDL/Verilog)
  - Embedded system design and HW/SW interfacing basics

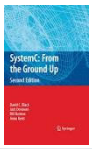
Course Administration

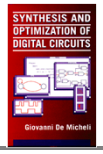
- **Instructors**
  - Instructor: Andreas Gerstlauer <gerstl@ece.utexas.edu>
    - Office hours: TTh 2-3pm, online (Zoom), scheduled via Canvas calendar
  - TA: Alexander Cathis <alexander.cathis@utexas.edu>
    - Office hours: TBD
- **Information**
  - Web: http://www.ece.utexas.edu/~gerstl/ece382m_f21
    - Lecture notes, homework and lab exercises.
  - Canvas
    - Announcements, assignments, grades.
  - Piazza: https://piazza.com/utexas/fall2021/ece382m20
    - All non-personal class discussions.
- **Dates (tentative)**
  - Exam: November 4, in class
  - Project design reviews: Nov. 30 & Dec. 2, in class
  - Final project presentations: December 11, 7-10pm

Textbooks

- **No required textbook**
- **Suggested references**
  - P. Marwedel, *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems and the Internet of Things*, 4th edition, Springer, 2021
    - Available under open access from Springer
    - 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
    - Electronic version available from the library
  - S. Pasricha, N. Dutt, *On-Chip Communication Architectures (System-on-Chip Interconnect)*, Morgan Kaufman, 2008
    - https://www.engr.colostate.edu/~sudeep/publications/on-chip-communication-architectures-system-on-chip-interconnect/
  - G. De Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, 1994
    - Out of print, but can be ordered; on reserve in the 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 20)**
  - Profiling of code on ARM board
    - Identify time consuming bottlenecks to optimize
  - Software optimization
    - Floating-point to fixed-point conversion
    - Improve performance, performance vs. detection accuracy tradeoffs
- **Lab 2: System architecture (3 weeks, due Oct 11)**
  - Hardware acceleration and system architecture
    - Identify and partition code into hardware and software
    - Hardware/software interface, communication architecture and drivers
  - Virtual platform modeling and simulation
    - Isolate and interface hardware as SystemC module
    - Co-simulation w/ software and firmware on ARM-Linux CPU
- **Lab 3: Accelerator design (3 weeks, due Nov 1)**
  - High-level synthesis of accelerator from C++ to RTL
    - Xilinx Vivado HLS tool
  - Verification of synthesis results
    - Testbench around standalone C++ vs. RTL hardware module

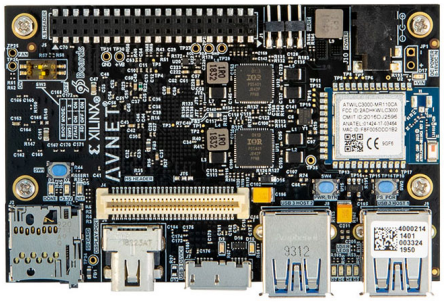
Class Project

- **Implementation on prototyping board**
  - Synthesizing hardware accelerator(s) into the FPGA
    - Synthesis and download using Xilinx software
  - C/C++ program on the ARM host processor
    - Cross-compilation or development under Ubuntu 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 real-time obj. detection**
  - Reference design of visual objection recognition ASIC
    - FPGA-based implementation as prototype of ASIC design
  - To be incorporated into cars, drones, …
    - Satisfy market and product requirements

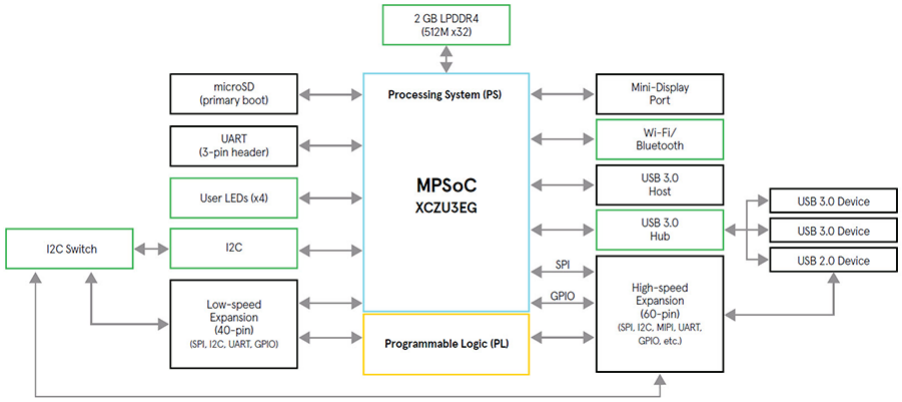
Prototyping Board

- **Ultra96 V2 by Avnet**
  - Xilinx ARM+FPGA programmable SoC
  - 2GB DDR4 RAM
  - microSD card socket
  - Wifi/Bluetooth (no wired Ethernet!)
  - 1x USB3 Micro-B upstream
  - 2x USB3 Type A downstream
  - DisplayPort
  - USB UART/JTAG port via daughter board

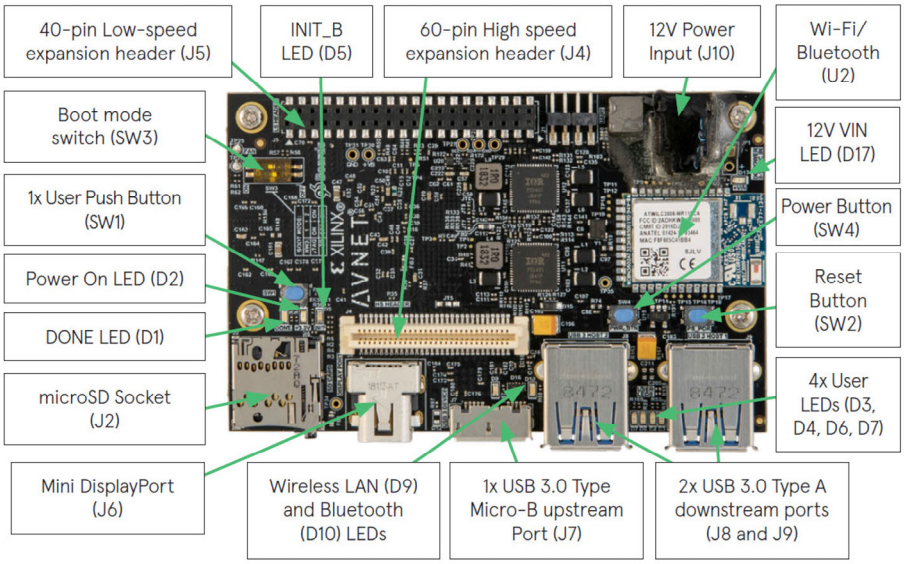


➢ **Based on Xilinx Zynq UltraScale+ MPSoC**
  - Quad-core ARM A53 + dual-core ARM R5 + ARM Mali GPU
  - Xilinx UltraScale FPGA fabric (16nm FinFET)

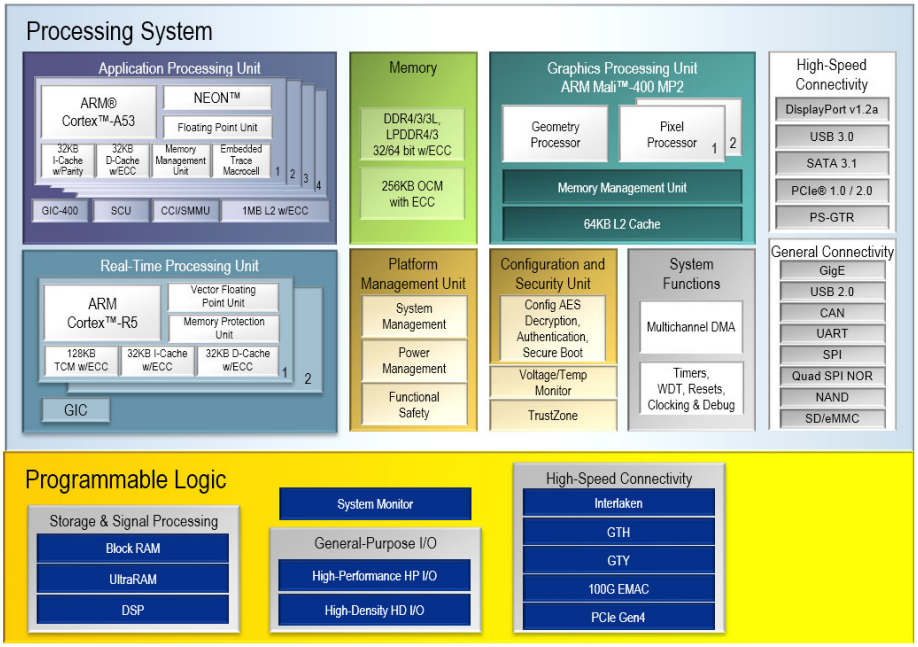
Ultra96 Block Diagram



Ultra96 Board Layout



Xilinx Zynq UltraScale+ MPSoC ZU3EG



Ultra96 Software Environment

- **We provide SD card running Ubuntu 18.04 Linux OS**
  - You will have root privileges
  - Two partitions
    - `/media/boot` – bootloader and Linux kernel
    - `/` - Ubuntu root filesystem
- **Some useful pre-installed utilities**
  - Accessing all memory-mapped hardware
    - `pm` – write memory location
    - `dm` – display memory location
    - `fm` – fill range of memory
    - `frm` – fill range of memory with random data
    - `smb` – set a bit in a memory word

Board Checkout

- **One board per team**
  - We have 21 boards, 2 students per team (max 3)
  - Pick up board from TA/instructor and sign contract
- **Board rules**
  - You will be responsible for board replacement if broken
    - These boards run $250
  - No food or drinks anywhere near the board (liquid damage)
  - Maintain ESD rules (grounding)
  - Be very careful with any commands run as root
    - Deleting files (you are responsible for your own backups)
    - Accessing memory-mapped locations can easily brick the board