EE382N-4
Embedded Systems Architecture Course

Mark McDermott

Goals of the Course

High-Level Goals

- Understand the scientific principles and concepts behind embedded systems, and
- Obtain hands-on experience in programming embedded systems.

By the end of the course, you will be able to

- Understand the "Big Ideas" in embedded systems
- Obtain direct hands-on experience on both hardware and software elements commonly used in embedded system design.
- Understand basic real-time resource management theory
- Understand the basics of embedded system application concepts such as signal processing and feedback control
- Understand, and be able to discuss and communicate intelligently about
  - embedded processor architecture and programming
  - I/O and device driver interfaces to embedded processors with networks, multimedia cards and disk drives
  - OS primitives for concurrency, timeouts, scheduling, communication and synchronization

What are the "Big Ideas"

- HW/SW Architecture
  - Non processor centric view of architecture
- Bowels of the "operating system"
  - Specifically, the lower half of the OS
  - Concurrency
- Real world design
  - performance vs. cost tradeoffs
- Analyzability
  - how do you "know" that your drive-by-wire system will function correctly?
- Application-level techniques
  - Power Aware Programming

So, what is an embedded system?

Simple answer -

Anything that uses a "processor" but isn't a general-purpose computer.

The user "sees" a smart (special-purpose) system as opposed to the computer inside the system

"how does it do that?"

"it has a computer inside it!"

"it does not or cannot run Windows or MacOS!"

But it might run Windows CE or Linux…

The end-user typically does not or cannot modify or upgrade the internals

Four General Categories of Embedded Systems

- General Computing
  - Applications similar to desktop computing, but in an embedded package
  - Video games, set top boxes, wearable computers, automatic tellers
- Control Systems
  - Closed loop feedback control of real time system
  - Vehicle engines, chemical processes, nuclear power, flight control
- Signal Processing
  - Computations involving large data stream
  - Radar, Sonar, video compression
- Communication & Networking
  - Switching and information transmission
  - Telephone system, Internet

Courtesy Intel

Types of Embedded System Functions

- Control Laws
  - PID control
  - Fuzzy logic, …
- Sequencing logic
  - Finite state machines
  - Switching modes between control laws
- Signal processing
  - Multimedia data compression
  - Digital filtering
- Application specific interfacing
  - Buttons, bells, lights,…
  - High speed I/O
- Fault response
  - Detection & reconfiguration
  - Diagnosis

Courtesy Intel

Distinctive Embedded System Attributes

- Reactive: computations occur in response to external events
  - Periodic events (e.g., rotating machinery and control loops)
  - Aperiodic events (e.g., button closures)
- Real-Time: timing correctness is part of system correctness
  - Hard real-time
    - Absolute deadline, beyond which answer is useless
    - May include minimum time as well as maximum time
  - Soft real-time
    - Missing a deadline is not catastrophic
    - Utility of answer degrades with time difference from deadline
  - Example:
    - a train is entering an urban area...
    - the railway gate in the city allows automotive traffic to go over the tracks
    - when should the railway gate close?

In general,

Real Time does not equal "Real Fast"

Courtesy Intel

Typical Embedded System Constraints

- Small Size, Low Weight
  - Handheld electronics
  - Transportation applications weight costs money
- Low Power
  - Battery power for 8+ hours (laptops often last only 2 hours)
  - Limited cooling may limit power even if AC power available
- Harsh environment
  - Heat, vibration, shock
  - Power fluctuations, RF interference, lightning
  - Water, corrosion, physical abuse
- Safety critical operation
  - Must function correctly
  - Must not function incorrectly
- Extreme cost sensitivity
  - $.05 adds up over 1,000,000 units

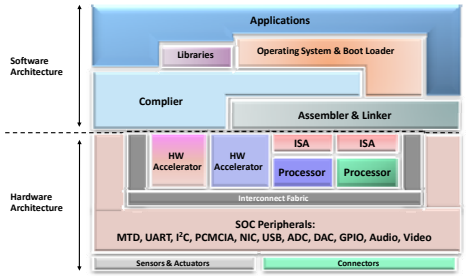
Courtesy Intel

A Hierarchy of Embedded Computer Design

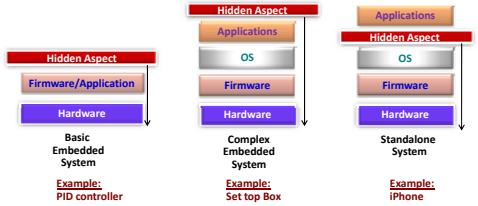
	Level	Name	Modules	Primitives	Descriptive Media
HARDWARE	1	Electronics	Gates, FF's	Transistors, Resistors, etc.	Circuit Diagrams
	2	Logic	Registers, ALU's	Gates, FF's	Logic Diagrams
	3	Organization	Processors, Busses, Memories	Registers, ALU's ...	Register Transfer Notation
FIRMWARE	4	Microprogramming	Assembly Language	Microinstructions	Microprogram
SOFTWARE	5	Assembly Language Programming	OS Routines	Assembly Language Instructions	Assembly Language Programs
	6	Procedural Programming	Applications, Drivers, etc.	OS Routines High Level Languages	HLL Programs
	7	Embedded Applications	Embedded Systems	Procedural Constructs	Problem Oriented Programs

Focus of this class

A Typical Embedded System

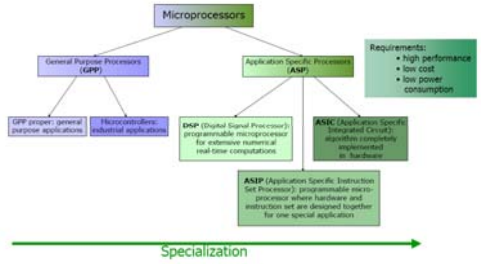


Types of System Abstraction

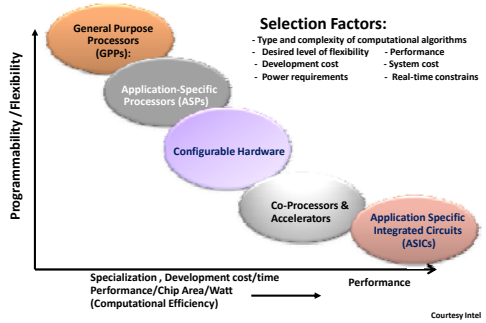


bmcnair@stevens.edu

Processor Classification



Computing Element Choices

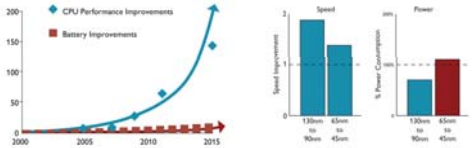


Selection Factors:

- Type and complexity of computational algorithms
- Desired level of flexibility
- Development cost
- Power requirements
- Performance
- System cost
- Real-time constrains

Courtesy Intel

The Real Problem facing Embedded Systems: POWER



Why ARM?

Because it is the "other" architecture

Course Syllabus

Hardware	Software
ISAs for Embedded Systems	Monitors & Bootloaders
GPP Microarchitectures	Embedded Linux
ARM ISA	Device Driver Development
FPGA Architectures	Interrupt Handlers
DSP Microarchitectures	Debugging Embedded Systems
Blackfin ISA	Real Time Operating Systems
Reconfigurable Logic	OS Services and Middleware
I/O Subsystems	SW Library Development
Networks-on-Chips	File System Development
Cognitive Sensors	SW Optimization

Grading

- Homework: 10%
- Labs: 30%
- Exam 1: 15%
- Exam 2: 15%
- Project: 30%

Penalty for late submission of homework and class project: 25% per working day. (Maximum: 100%).

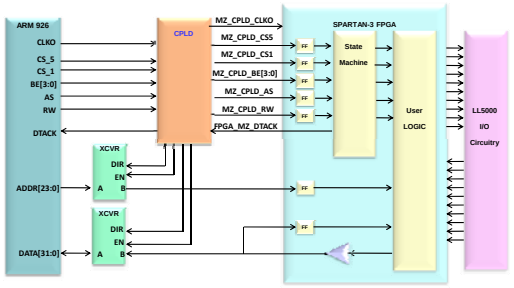
Overview of the Lab Assignments

- **Lab #1**
  - Develop an Interrupt handler to measure the latency of Linux running on the TLL2020 platform using the I/O ports on the FPGA.
- **Lab #2**
  - Write an ARM assembly language program (ALP) implementation of a memory test. Use the ARM ADS system to verify.
- **Lab #3**
  - Write an ALP and/or C-routine which slows down and speeds up the processor clock on the ARM processor without disrupting the I/O capability. The processor will be performing a memory tests on the Block RAM that is in the FPGA.

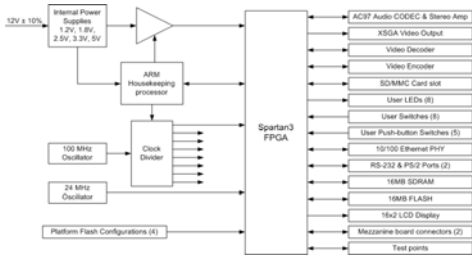
ENS 114 Lab

- **Located on first floor of ENS.**
- **Systems can be accessed via Remote Desktop**
  - MAC ,PC or Linux Desktop
- **ABSOLUTLEY NO FOOD OR DRINK ALLOWED IN THE LAB**
  - Zero tolerance. Two students received F's in the lab for spilling drinks on the equipment.

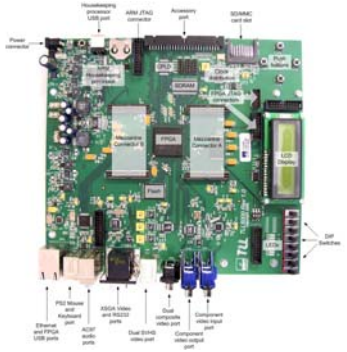
TLL5000/TLL6219 Prototype System Block Diagram



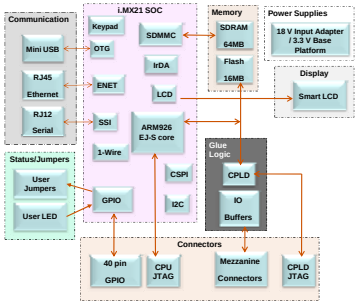
TLL-5000 Block Diagram



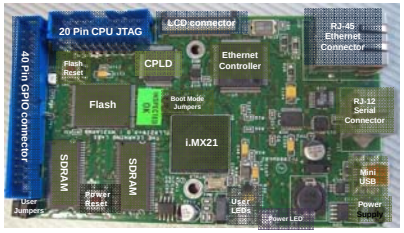
TLL-5000 Baseboard



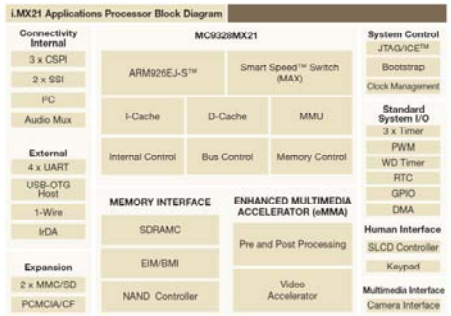
TLL 6219 Block Diagram



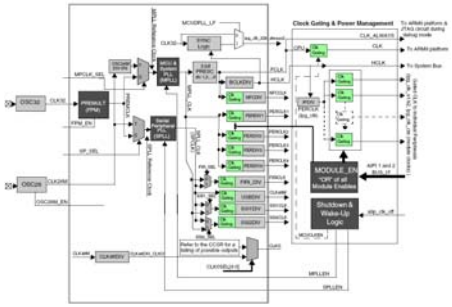
TLL6219 ARM 926-EJS Board



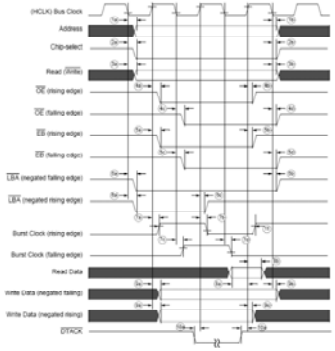
i.MX21 Features



ARM926 Clock Generation



ARM926 Bus Cycle Waveforms and Timing



TLL6219 Memory Map