Platforms & Platform-based design

Competing imperatives

- Technology push:
  - high-volume products;
  - feasible design.
- Marketing push:
  - fast turnaround;
  - differentiated products.
What is a (Hardware) platform?

- A partial design:
  - for a particular type of system;
  - includes embedded processor(s);
  - may include embedded software;
  - customizable to a customer’s requirements:
    - software;
    - component changes.

Why platforms?

- Any given space has a limited number of good solutions to its basic problems.
- A platform captures the good solutions to the important design challenges in that space.
- A platform reuses architectures.
Standards and platforms

- Many high-volume markets are standards-driven:
  - wireless;
  - multimedia;
  - networking.
- Standard defines the basic I/O requirements.

Standards and platforms, cont’d.

- Systems house chooses implementation of standards functions:
  - improved quality, lower power, etc.
- Product may be differentiated by added features:
  - cell phone user interface.
- Standards encourage platform-based design.
Platform vs. full-custom

- **Platform** has many fewer degrees of freedom:
  - harder to differentiate;
  - can analyze design characteristics.
- **Full-custom**:
  - extremely long design cycles;
  - may use less aggressive design styles if you can’t reuse some pieces.

Platforms and embedded computing

- Platforms rely on embedded processors:
  - can be customized through software;
  - can put considerable design effort into the CPU.
- Many platforms are complex heterogeneous multiprocessors.
Platforms and IP-based design

- Platforms use IP:
  - CPUs;
  - memories;
  - I/O devices.
- Platforms are IP at the next level of abstraction.

Advantages of platform-based design

- Fast time-to-market.
- Reuse system design---hardware, software.
- Allows chip to be customized to add value.
Costs of platform-based design

- Masks.
- NRE: design of the platform + customization.
- Design verification.

Two phases of platform-based design

- Design the platform.
- Use the platform.
Division of labor

- Platform design:
  - choose, characterize hardware units;
  - create the system architecture;
  - optimize for performance, power.
- Platform-based product design:
  - modify hardware architecture;
  - optimize programs.

Semiconductor vs. systems house

- Semiconductor house designs the platform.
- Systems house customizes the platform for its system:
  - customization may be done in-house or by contractor.
Platform design challenges

- Does it satisfy the application’s basic requirements?
- Is it sufficiently customizable? And in the right ways?
- Is it cost-effective?
- How long does it take to turn a platform into a product?

Platform design methodology

- Size the problem.
  - How much horsepower? How much power?
- Develop an initial architecture.
- Evaluate for performance, power, etc.
- Evaluate customizability.
- Improve platform after each use.
Platform use challenges

- How do I understand the platform’s design?
- How do I modify it to suit my needs?
- How do I optimize for performance, power, etc.?

Platform use methodology

- Start with reference design, evaluate differences required for your features.
- Evaluate hardware changes.
- Implement hardware and software changes in parallel.
- Integrate and test.
Design refinement

- Bad news:
  - hard to learn the platform in order to change it.

- Good news:
  - an existing design can be measured, analyzed, and refined.

Worldwide shipping by UPS ...

roughly US$ 50 for CD and US$ 100 for paper copy

(1500 pages, heavy!)

Hardware Platforms

A Hardware Platform is a family of architectures that satisfy a set of architectural constraints imposed to allow the re-use of hardware and software components.
Hardware Platforms Not Enough!

- Hardware platform has to be abstracted
- Interface to the application software is the “API”
- Software layer performs abstraction:
  - Programmable cores and memory subsystem “hidden” by RTOS and compilers
  - I/O subsystem with Device Drivers
  - Network with Network Communication Software

Software Platforms
Software and hardware reuse

- Want to reuse as many hardware components as possible:
  - known performance, power.
- Want to use software libraries where possible.
- RTOS simplifies design of multi-tasking systems.

Platforms: the hourglass model
HW-SW Platforms: Exploration and Reuse

Example system platforms

- Generic;
- Automotive;
- Wireless;
- Multimedia;
**PC-based platform**

- Basic hardware components:
  - CPU;
  - memory;
  - timers;
  - DMA;
  - minimal I/O devices.
- Basic software:
  - BIOS.

**PC-style hardware architecture**
Strong ARM

StrongARM system includes:

- CPU chip (3.686 MHz clock)
- system control module (32.768 kHz clock).
  - Real-time clock;
  - operating system timer
  - general-purpose I/O;
  - interrupt controller;
  - power manager controller;
  - reset controller.

Pros and cons

- Plentiful hardware options.
- Simple programming semantics.
- Good software development environments.
- Performance-limited.
**TI Open Wireless Multimedia Applications Platform**

- Dual-processor shared memory system:

  ![Diagram of dual-processor shared memory system]


---

**TI OMAP™ Hardware platform**

- ARM9 core
- 16KB I-cache
- 8KB D-cache
- 2-way set associative
- 150 MHz

- C55x DSP core
- 16KB I-cache
- 8KB RAM set
- 2-way set associative
- 200 MHz

![Diagram of OMAP hardware platform](http://www.ti.com/sc/docs/apps/wireless/omap/overview.htm)
OMAPI Standard (ST/TI)

- Goal: standardize the interfaces between application processor and peripheral devices in a mobile product
- Provide standard services (APIs) in the OS that can be used by application developers

STMicro Nomadik platform
Nomadik SW platform

- Compliant with OMAPI standard

Philips Digital Video Nexperia Platform

General-purpose Scalable RISC Processor:
- 50 to 300+ MHz
- 32-bit or 64-bit

Library of Device IP Blocks:
- Image coprocessors
- DSPs
- UART
- 1394
- USB
...and more

Scalable VLIW Media Processor:
- 100 to 300+ MHz
- 32-bit or 64-bit

Nexperia™ System Buses:
- 32-128 bit
Nexperia-DVP Software

- Nexperia™ -DVP Software Architecture
  - Supports multiple OSs and middleware software
  - Abstracts platform functionality via consistent APIs
- Nexperia™-DVP Streaming Software
  - Encapsulates implementation of streaming media components (hardware and software)
- Nexperia™ Platform Software
  - OS independent device drivers for on-chip and off-chip devices

Infineon Automotive Platform

- Applications
  - High Performance drives / servo drives,
  - Industrial control Robotics
- Features
  - 32-bit super-scalar TriCore™ V1.3 CPU, 4 stage pipeline
    - Fully integrated DSP capabilities
    - Single precision floating point unit (FPU)
    - 80 MHz at full industrial temperature range
  - 32-bit peripheral control processor with single cycle instruction (PCP2)
  - Memories
    - 1.5 MByte embedded progr. flash with ECC
    - 32 KByte data flash - EEPROM emulation
    - 56 KBSRAM, 8 KB IS, 16 KB Imem
  - 8-channel DMA controller
  - Interrupt system with 2 x 255 hardware priority arbitration levels serviced by CPU and PCP2 Coprocessor
  - Triple bus structure: 64-bit local memory buses to internal flash and data memory, 32-bit system peripheral bus, 32-bit remote peripheral bus
Summary

- Platform-based design helpful for reducing development time (by reuse)
- Tool support for design exploration available
- But... how to design the platform??