

Codesign: The World Of Practice

D. Sreenivasa Rao

Senior Manager, System Level Integration Group

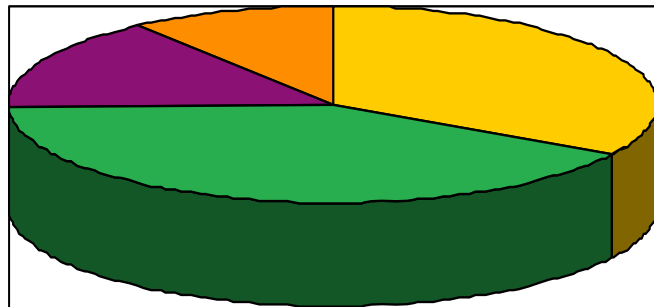
Analog Devices Inc.

May 2007

Analog Devices Inc.

ADI is focused on high-end signal processing chips and the markets that need them

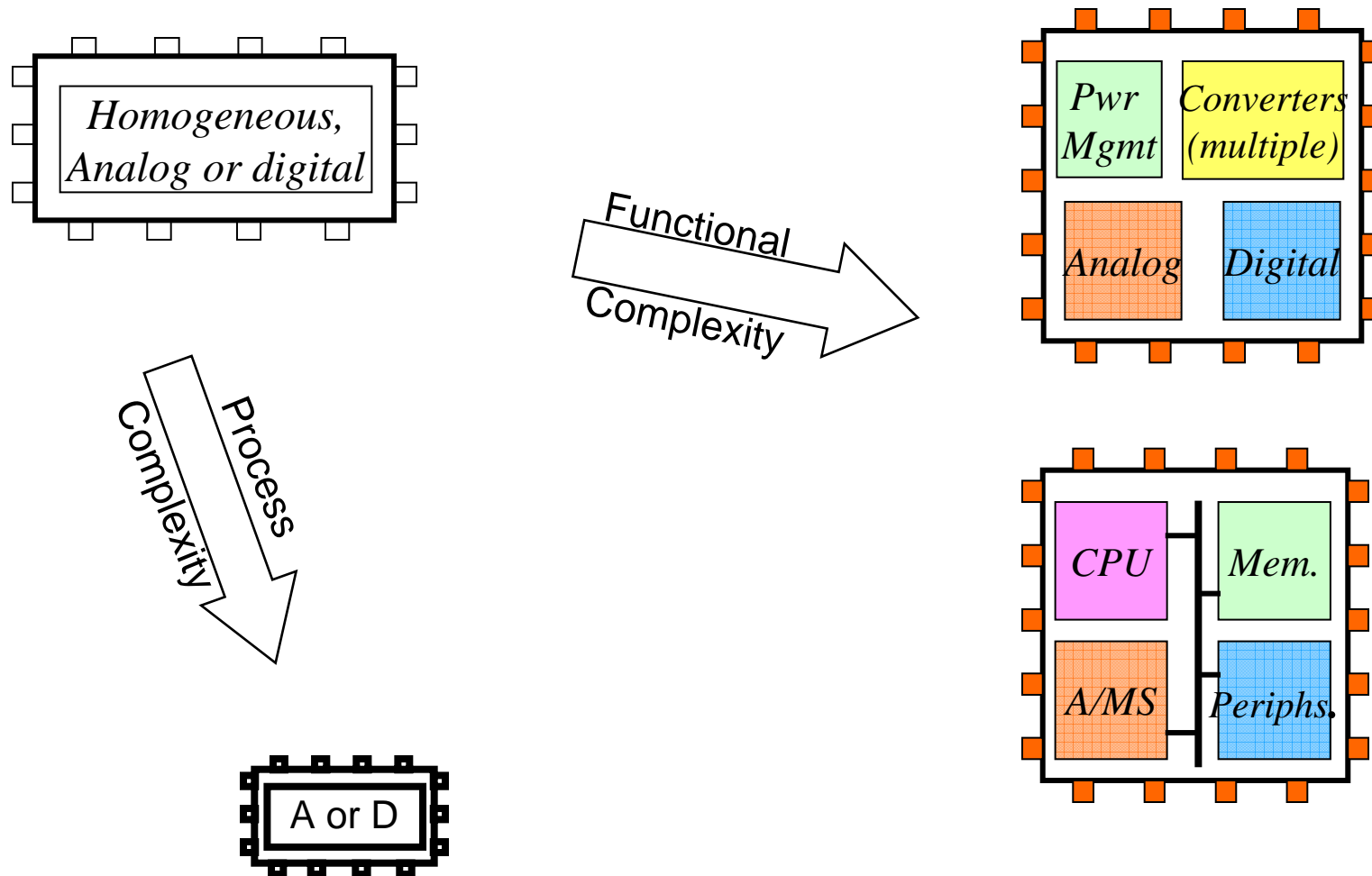
Markets Served



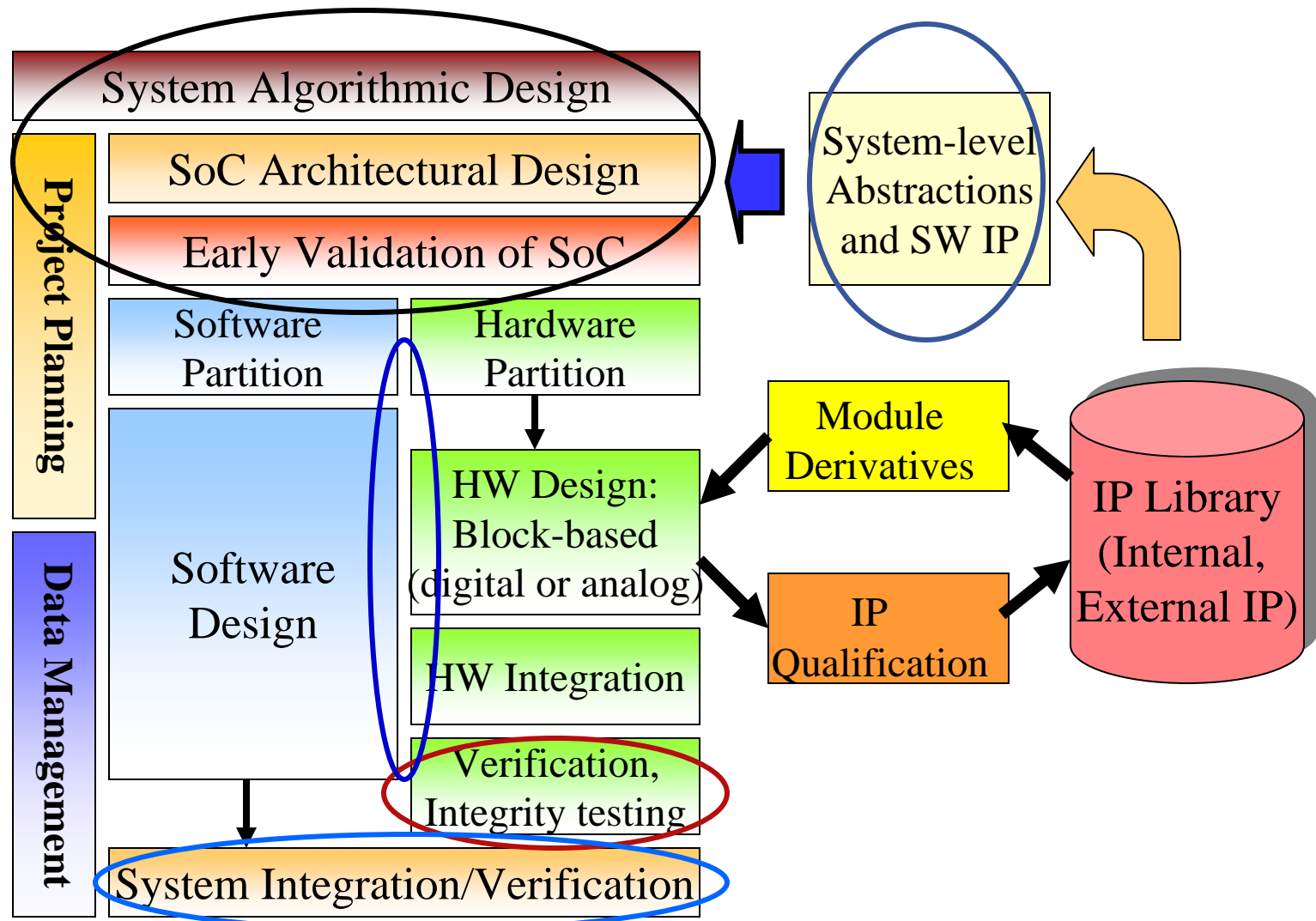
- ◆ Primary Competitors are:
Texas Instruments, Broadcom, Linear Tech, Maxim, STMicro

- ◆ Product portfolio includes:
 - Analog IC's, ASSP's (73%)
 - DSP's, Ex. Blackfin (22%)
 - MEMS products (5%)
- ◆ Basic Stats:
 - FY 06 revenues \$2.65B
 - Gross margins usually upwards of 60%
- ◆ Increasing focus is on CE, whose market characteristics are challenging:
 - Narrow windows
 - Short product lifetimes
 - High R&D/Manf. costs

Evolution of Chip Designs



SoC's, Or Where We Do Co-Design





The Torturous World of Co-Design

- ◆ For modeling and verifying complex hardware designs, solutions are slowly but surely becoming available
 - Early languages were imperfect for modeling and for simulation
 - Our path has taken us (and is still) through PSL, SVA, BSV, SysV, etc.
- ◆ EDA is (finally) converging on SystemC as the language of choice
 - Tool support is not mature yet for mainstream usage
 - “Abstraction bridges” are still lacking – ie., the ability to model untimed testbenches and combine them with timed RTL models
 - We routinely develop C models of algorithmic blocks and use them for testing of the developed RTL

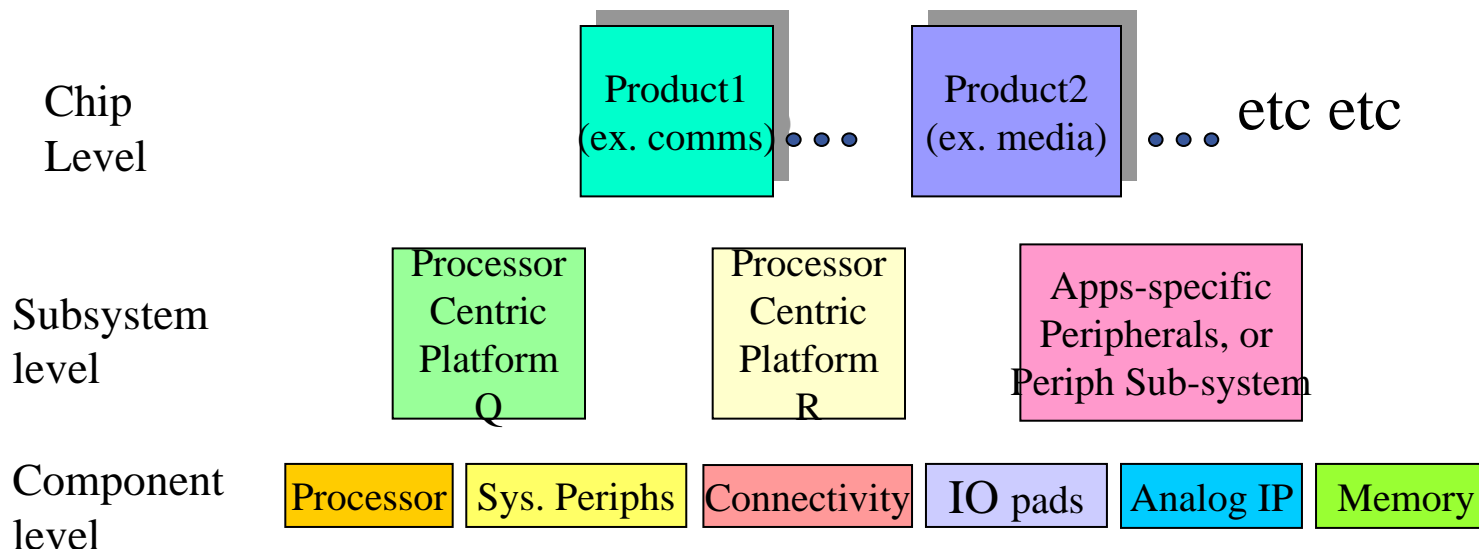


The Torturous World of Co-Design (Cont'd)

- ◆ Another aspect of co-design is hardware design at higher levels of abstraction
 - Usage of C or SystemC is popular, although other languages exist
 - ◆ There are a host of solutions available now
 - ◆ Direct synthesis from C is most preferable (one time development of algorithmic code), but existing tools work only for datapaths
 - ◆ Two big methodology holes:
 - **Algorithmic verification capabilities**
 - **C-to-RTL validation**
 - A key advantage is architectural exploration, for which other, graphical solutions are also appropriate

Our Vision: Develop Apps-Specific Platforms.....

- ◆ **Several end-use systems share similar core sub-systems**
 - Ex. A micro-controller with common peripherals, memory, bus management, and bridges
- ◆ **A *platform* is simply an abstraction that enables this level of sharing**
 - An architectural framework or “family”, consisting of a set of hardware blocks wrapped around a standard bus and other infrastructure, that supports rapid embedding into a larger chip for quick product development
 - A platform should be accompanied by a set of methods/methodologies to support customization and integration
- ◆ **The advantages of a platform include increased design productivity, higher reliability and first silicon success, as well as lower manufacturing costs**





.... And Prototype them for Software Development

- ◆ Platforms are meant to accelerate development of key apps software for market segments or classes
 - In addition, platforms make it easy for us explore design alternatives
- ◆ Accurate, flexible, and scalable models of platform hardware must be easily available
 - Platform abstraction must handle new or partly-defined functions
 - It must avoid double modeling – once in HW and once in SW
- ◆ Creating software models of platforms is more than cobbling together the individuals C models or BFM's
 - The model must fully mirror the functionality of the embedded system
 - RTL simulation is too slow, and Instruction Set Simulation doesn't scale well for complex SW
 - The platform must be 'hybrid', allowing some portions at implementation level (ASIC or FPGA), some as VM's, some as C models, some cycle-accurate, some cycle-approximate, etc.
 - The methodology of platform creation and usage are keys to success