Embedded Systems on a Chip: Introduction

Sergio Lopez-Buedo, Elías Todorovich
etodorov@exa.unicen.edu.ar
July 2013

Computers…

- When you hear the word computer, you always think of a desktop computer:

- But we are surrounded by thousands of invisible computers, embedded in complex devices…

Embedded Computers

- Every electronic or electromechanical system has a computer (or many) embedded in it:

So… What Is an Embedded System?

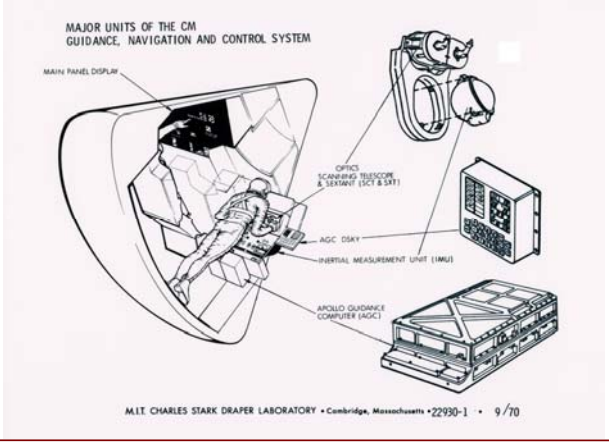
- An embedded system is a **special-purpose** computer system built into a larger device
  - Embedded systems are required to meet very different requirements than a general-purpose personal computer.
- Two distinguishing characteristics:
  - **Cost**: Since embedded systems are produced in the tens of thousands to millions of units range, reducing cost is a major concern. Embedded systems often use a (relatively) slow processor and small memories to minimize costs.
  - **Space and Power** consumption: Usually embedded systems have tight space limitations, so they must be built out of a few components (this also benefits the cost) and complex power dissipation mechanisms (like fans) are not allowed.

What Is an Embedded System? Software Issues

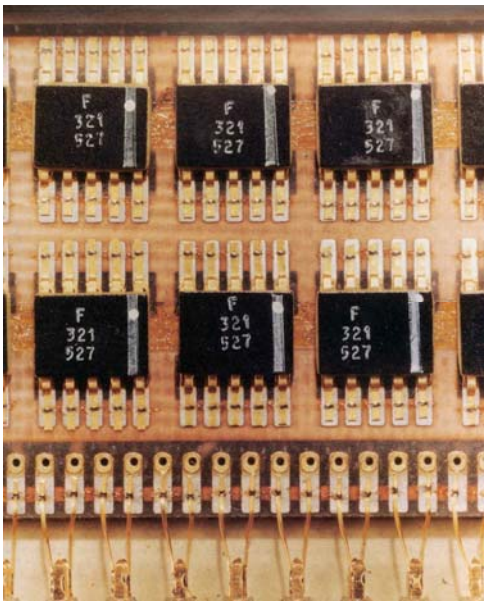
- The third feature of an embedded system is that it is single-functioned, not general-purpose like a desktop
  - The SW run in the embedded systems is called **Firmware.**
  - It is stored in non-volatile memory (Flash, ROM).
  - At most they have a simple operating system, but in most cases they have no OS at all.
- An embedded system is a real time computer.
- Reliability is a must
  - In most embedded systems a software crash is simply not allowed, e.g. ABS brake system or DVD player.

A Little History…

- One of the first embedded systems was the Apollo Guidance Computer, started in 1962.
- "One of the reasons that we were able to do all that we did during the Apollo Program was that we had a terrific computer", David Scott, Astronaut.



The Apollo Guidance Computer



- It was developed by the Charles S. Draper team at the MIT.
- The first computer built using ICs (4100 3-input NORs. connected by wire-wrapping)
- 4K RAM (ferrite core)
  32K ROM (core rope mem)
- 11 instructions, 16 registers, 5 vectored interrupts
- Programmed in assembler, it had a non-preemptive multi-tasking OS (up to 8 jobs).

The Apollo Guidance Computer

- En el MIT ganaron el contrato para desarrollar la computadora que llevó a la Apollo 11 a su destino y que se usó en el resto de las misiones Apollo.
- Tenía que ser muy eficiente y gastar muy poca energía. Patentaron el acceso directo a la memoria.
- Ramón Alonso es el argentino que definió la arquitectura de esa computadora. También creó la interfaz de la computadora, es decir, el teclado y el visor que usaron los astronautas para operar la computadora.

http://www.lanacion.com.ar/1240769-ramon-alonso-el-argentino-que-llevo-a-la-apollo-11-a-la-luna

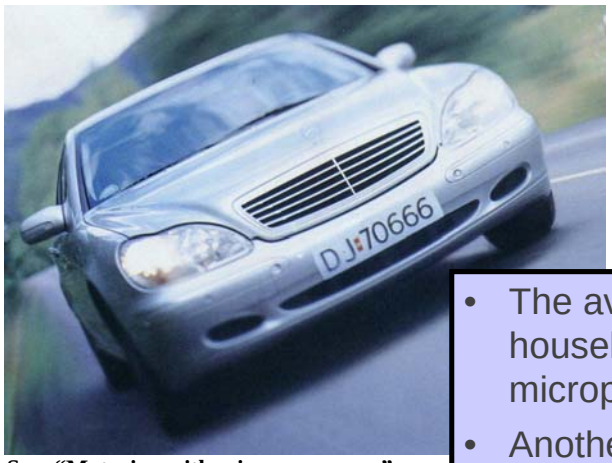
Embedded Systems Today

- Most of the components are integrated in one chip
  - System on a chip, SoC, with millions of transistors
  - Can integrate analog and RF sections, and flash memories
  - Only SDRAM, some passive components and power supply are needed
- Multiprocessing:
  - A general purpose processor together with a DSP, and also there might be custom co-processors (e.g. MPEG-4)
- Plenty of peripherals:
  - General-purpose (Timers, INTC, UART) as well as specific ones (HDD, Memory Card, Wireless, USB…)

Embedded Systems Today

- Telecommunications: from telephone switches for the network to mobile phones
- Consumer electronics: PDA, mp3 players, videogame consoles, digital cameras, DVD players, GPS receivers, and printers
- Household appliances: microwave ovens, washing machines, dishwashers
- Home automation uses wired- and wireless-networking that can be used to control lights, climate, security
- Transportation systems from flight to automobiles: Other automotive safety systems include anti-lock braking system (ABS), Electronic Stability Control (ESC/ESP), traction control (TCS) and automatic four-wheel drive
- Medical equipment

Embedded Systems Are Everywhere

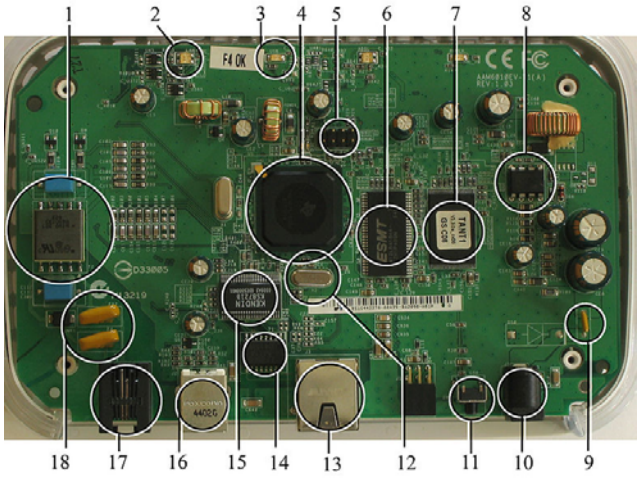


- The average middle-class American household has about 40 to 50 microprocessors in it.
- Another 10 microprocessors are in every PC.
- The 7-Series BMW and Mercedes S-class has about 100.

See: "Motoring with microprocessors" By Jim Turley http://www.eetimes.com, 2003

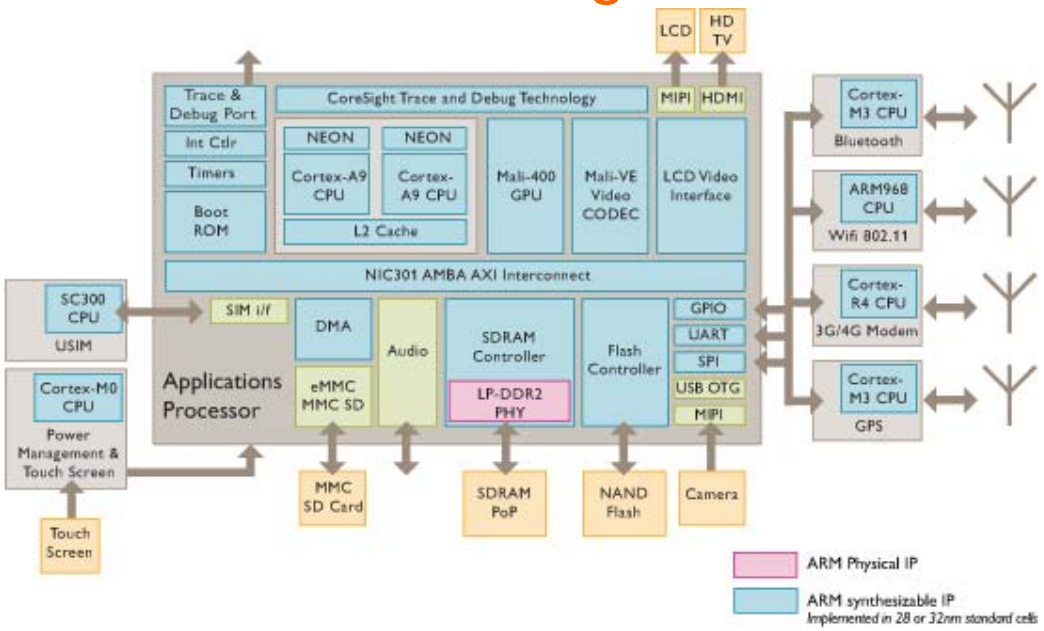
See: "This Car Runs on Code" By Robert N. Charette , http://spectrum.ieee.org, 2009

Embedded Systems Today: ADSL modem/router



4. Main processor, a TNETD7300GDU, a member of Texas Instruments' AR7 product line.
6. RAM, a single ESMT M12L64164A 8 MB chip.
7. Flash memory, obscured by sticker.

Optimized ARM Smartphone Block Diagram



Smartphones for a wide range of tasks

- Accessing the full Internet including social networking
- Creating and consuming multimedia including HD video, and accessing full online content using Adobe's Flash Player.
- GPS navigation and location based services.
- Gaming including using the latest OpenGL ES 2.0 technology.
- Messaging from SMS to Email.
- And of course **a phone**, as many consumers "cut the cord" and use their Smartphone as their only point of communication.

HTC Desire HD

Most Embedded Systems Are Simple

- Intel just holds a 2% of the global processor market
  - Currently there are 40 MCU vendors, 1300 different 8-bit devices and 240 32-bit alternatives
- Plenty of problems are solved with a simple 8-bit microcontroller:
  - Successful PIC and Motorola HC08 families
  - Low-cost and low-power, and have SoC features (a lot of peripherals, analog sections)
  - Known devices, no need for learning new design flows
- The best solution is not the most complex, is the one that optimizes the tradeoff between all the different design metrics.

Design Metrics

- Cost
  - Unit cost: The cost of manufacturing each unit (just the component and assembly costs)
  - NRE (Non-Recurring Engineering Cost): The one-time cost of designing the system (includes prototyping)
- Performance
  - Most important: Ability to work in real time
  - Time to give a response to the user (e.g. navigation system)
- Physical Constraints
  - Size
  - Power

Design Metrics (2)

- Development time
  - Time-to-protoype: The time needed to build a first working version of the system.
  - Time-to-market: The latter time, plus the time to manufacture the product and distribute it to the resellers.
- Other design issues
  - Flexibility: the system can be used for different tasks without significantly increasing the NRE or the unit costs.
  - Maintainability: the system can easily be updated and/or repaired.
  - Fail-safe operation: Necessary in mission-critical systems, like life support.

8-bit or 32-bit Processors?

Most embedded systems do simple tasks, so a 8-bit MCU would be enough to build the system

- Advantages:
  - Unit cost for the MCU is very low, cents per unit.
  - Size and power very good as well.
- Disadvantages:
  - Poor performance (old architectures).
  - Time-to-prototype: bad because they do not have powerful SW tools (OS and libraries, due to the limited address space).
  - The NRE is not as low as expected (poor development time, skilled developers are needed to obtain the maximum performance out of limited MCUs).

Why Is it Better to Use a 32-bit Processor?

- Increased addressing space
  - Complex development tools can be used (GNU tools, operating systems such as Linux, libraries and C++).
  - Inexpert programmers are not so harmful.
- Ability to work with DRAM
  - 8-bit processors can only use SRAM which is much more expensive and has less capacity.
- Increased performance
  - 8-bit processors have old (20 year) architectures.
- Network support
  - Today Ethernet is as necessary as it was an UART in the past, and 8-bit processors do not have the performance needed to deal with complex TCP/IP applications.

Why not 16-bit?

- There is not a real advantage in using 16-bit processors, because:
- They have the same cost as 32-bit processors
  - The core size is much less than the area occupied by the peripherals and the memories.
- But its performance is much less
  - Once again, they do not have the advanced (pipelined) architectures that 32-bit processors have.
- And they do not have available complex SW tools
  - Lack of sophisticated operating systems, such as Linux .

What Is the Best Alternative?

- A lot of systems are quite simple, so an 8-bit processor would be enough. However, we prefer using 32-bit processors because:
  - They are easier to program (we use C or C++ together with an operating system or complex libraries).
  - The increased performance leaves room for future updates.
  - It is easy to connect them to the Ethernet (all have available a TCP/IP stack).
- What frightens us about 32-bit MCUs is the time needed to learn a complex processor:
  - MMUs, caches, pipelining, coprocessors…

What is the Best Alternative? (2)

The best alternative for moderately complex systems is to use a simple 32-bit processor:

- Pipelined RISC processors (good performance) that conceptually work like a simple 8-bit MCU:
  - They do not have MMUs or complex caches
  - Simple architecture (easy to understand)
- But they have available all the powerful tools that the 32-bit processors have (GNU tools, libraries, operating systems…)
- Examples: ARM7, MIPS32, IBM PowerPC, Altera Nios II, Xilinx MicroBlaze…

Advantages of a Simple 32-bit Processor

- Low unit cost ($2 higher than a 8-bit MCU), and this higher cost is masked by the possibility of using lower cost components (like DRAM instead SRAM).
- Low NRE because the development process is easier
  - They are programmed in C, and we have plenty of libraries/OS to support the development.
  - They can be used by inexperienced programmers (they are simple and have space for poor programming techniques).
- Moderate performance, enough for most tasks.
- Better flexibility (because of the increased performance and addressing space).
- The only drawbacks in comparison to a 8-bit processor are space and power.

Technology Options for SoCs

- Microcontroller
  - Find an off-the-shelf SoC that has all the components needed in your system.
- ASIC
  - Create your custom silicon, usually using IPs (intellectual property) from various vendors.
- FPGA
  - Use a general-purpose programmable device to build your system. The chips are off-the-shelf (like MCUs) but you construct your own circuit using IPs (like ASICs).

MCUs

- Currently there are a lot of microcontroller options.
  - Basically, they are SoCs composed by a processor and a lot of peripherals.
- There are general purpose MCUs, but the most common are the MCUs for a specific application:
  - Motor control
  - Video or audio processing
  - Automotive
- They are a very good option when they fit your needs:
  - Very good performance, like ASICs.
  - Low unit cost, and the lowest NRE.

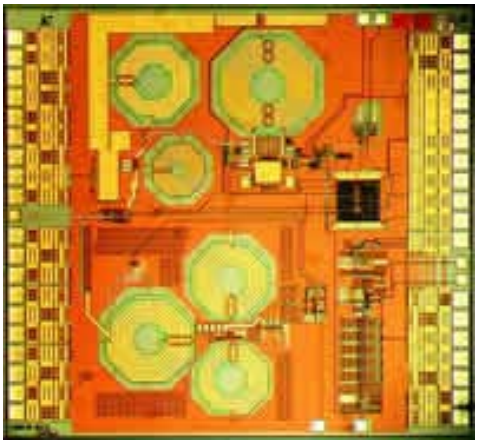
MCUs (2)

- The have the lowest NRE because:
  - There are no expenses to design or manufacture the chip.
  - They usually provide drivers to handle the peripherals.
- Another advantage of MCUs is that you do not need to care about IPs.
  - All royalties have been paid when you buy the chip.
- But unfortunately, you always need something more:
  - A certain peripheral.
  - Glue logic.
- So they almost never end up being true SoCs.

ASICs

- Custom silicon
  - Usually based on standard-cell technology.
  - Also gate arrays and structured ASICs.
- Technically, the best option:
  - Best speed (hundreds of MHz) and low-power.
  - Integration of analog components (mixed-signal design).
- The system is designed for satisfying all your needs
  - Minimal external components, only the ones that cannot be implemented in silicon due to technology limitations.
  - Proper SoC technology.

ASIC Technology



- The CMOS technology can also be used to implement analog cores
  - RF CMOS adds a few layers and permits implementing inductors.
  - Higher performance BiCMOS can also be used (but at a higher cost).
- CMOS is also compatible with flash memory processes
- However, it cannot implement DRAM (at a reasonable cost).

NREs for ASICs

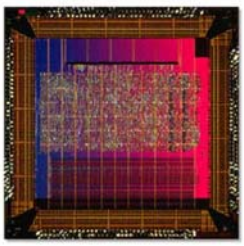
The biggest problem of ASICs is their huge NRE

- Minimum €250,000 - €500,000 to develop the first prototypes (masks and manufacturing setup).
- Time-to-prototype is poor, because:
  - The complexity of the design flow.
  - The exhaustive verification needed.
  - The fabrication delays (several weeks).
- Expert HW designers are needed.
- Also the IP royalties.
  - Can reach a million € for leading microprocessors (ARM).
  - And a small amount must be paid per unit sold.

Structured ASICs

- Though the unit cost is very low, ASICs are only worthwhile when they are produced in millions because of NREs.
- An alternative with lower NREs are structured ASICs
  - Only 10,000 to 100,000 € NRE cost.
  - Like gate arrays, but containing other components as large memory blocks.
  - Only the upper metal layers are custom.
  - Faster implementation times (few weeks).
- Some vendors are NEC, AMI, Gigoptix (ChipX) and LSI Logic.

Example: Gigoptix (ChipX) CX50000



- 30K to 1.1M usable ASIC gates.
- Up to 2.6M bits of fast block memory.
- 2ns access time single-port SRAM, dual-port SRAM and ROM.
- Low power consumption (0.06uW/MHz/Gate).
- 200MHz general core logic operation, up to 650MHz.
- PCI, PCI-X, SSTL, HSTL, USB1.1, LVPECL and LVDS up to 622Mbps.
- 1.5V or 1.8V or mixed supply voltage operation.
- Up to 1152 total pads.
- Low-jitter analog PLL macros with internal loop filter.
- 3-4 week lead time for tested prototypes.

FPGAs

- A solution between MCUs and ASICs
  - The best of the two worlds: off-the-shelf components (like MCUs) where you create your own logic (like ASICs).
- A good alternative for creating SoCs
  - Huge capacity, they can reach millions of equivalent gates.
- Current FPGAs are no longer just gate arrays
  - Large SRAM blocks
  - Signal processing blocks
  - Processors, Ethernet, Gigabit transceivers…
- In the most popular FPGA technology the configuration is stored in SRAM, so their functionality can be easily changed.

Advantages of FPGAs

- Low NRE costs
  - They are off-the-shelf components
  - Much easier to design than ASICs
  - The verification usually is not critical (if there is a failure, just reprogram the FPGA)
- Very fast design times
  - System on chip in days
- Very good flexibility and maintainability
  - You create the logic that you need
  - There are different FPGA models with the same pinout
  - HW updates, even remotely!

Disadvantages of FPGAs

- High unit cost
  - Much higher than a ASIC
- High power
  - Nowadays, this is especially true for static power
  - An ASIC running can dissipate as much power as an idle FPGA
- Lower speed
  - ASICs are several times faster (at least two times)
- Lack of mixed-signal support
  - FPGAs are just digital circuits
  - ASICs can be used to implement analog (even RF) sections

Example: MicroBlaze

- 32-bit RISC pipelined processor, 4GB addressing space.
- It´s a soft processor core designed for Xilinx FPGAs.
- Many aspects of the MicroBlaze can be user configured:
  - Cache size, pipeline depth (3-stage or 5-stage), embedded peripherals, memory management unit, and bus-interfaces.
- Plenty of tools available:
  - GNU tools
  - Libraries (libc, threads, TCP/IP, …)
  - Operating systems: uCLinux, VxWorks…
  - IDE: proprietary, EDK (Embedded Development Kit)

Example: MicroBlaze (2)

- Drivers for a lot of peripherals:
  - Ethernet, flash, UART, timers…
- Speed: 210 MHz on Virtex 5, 303 MHz – 392 DMIPs on Virtex 6, 311 MHz – 404 DMIPs on Kintex 7.
- With the MMU, MicroBlaze is capable of hosting operating systems requiring hardware-based paging and protection, such as the Linux kernel.
- Drawback: proprietary.
- Clones:
  - MB-Lite, implemented in VHDL, LGPL license (OpenCores)

Example: Nios II

- It's a 32-bit RISC pipelined processor, 2GB addressing space.
- Soft processor core designed for Altera FPGAs.
- Many aspects of the MicroBlaze can be user configured:
  - Cache size, pipeline depth (up to 6-stage), embedded peripherals, memory management unit, and bus-interfaces.
  - Custom Instructions.
- Plenty of tools available:
  - GNU tools
  - Libraries (libc, threads, TCP/IP, …)
  - Operating systems…
  - IDE: proprietary, Qsys System Integration Tool (Embedded Development Suite (EDS) up to software version 10.0)

Example: Nios II (2)

- Drivers for a lot of peripherals:
  - Ethernet, flash, UART, timers…
- Speed:
  - Up to 250 MHZ, 1,13 MIPS/MHz (Nios II-e) on Stratix V.
- With the MMU supports the Linux kernel. Without the MMU the OS which use a simplified protection and virtual memory-model: e.g., µClinux and FreeRTOS.
- Nios II is comparable to MicroBlaze.
  - Nios II is licensable for standard-cell ASICs through a third-party IP provider, Synopsys Designware.
- Drawback: proprietary.

Other processor soft cores

- LatticeMico32: Another 32-bit RISC pipelined processor, designed by Lattice Semiconductor.
  - The Verilog HDL source code is licensed under an open intellectual property (IP) core license. This means that the Mico32 is not restricted to Lattice FPGAs, and can be legally used on any host architecture (FPGA, ASIC, or software emulation).
  - The development toolchain is available in source-code form.
- JOP: a Java processor soft-core for FPGAs (http://www.jopdesign.com)
  - This processor is open-source (all VHDL and Java sources are on the website) and free for educational purpose.

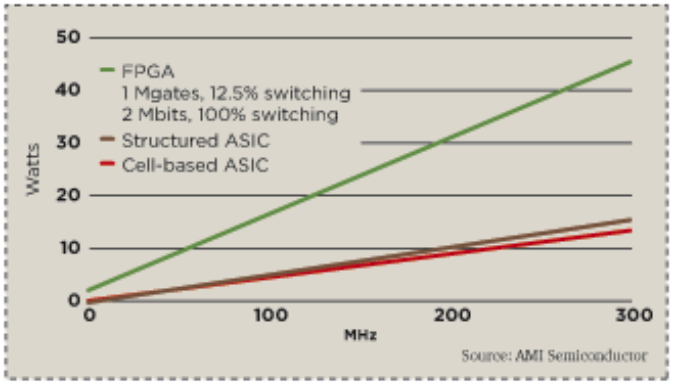
Other processor soft cores (2)

- Altera (Arrow Electronics) provides as a megafunction the ARM Cortex-M1 core.
  - It's a streamlined, three-stage, 32-bit RISC processor. Implements the Thumb®-2 instruction set enabling both the processor and software footprint to meet the area budget of the smallest FPGAs, while retaining compatibility with Thumb code for any ARM processor from the ARM7TDMI® processor upwards. Despite being the smallest processor in the Cortex processor family, the Cortex-M1 processor delivers 0.8 DMIPS per MHz.
- Also as a megafunction Altera provides cores for the Freescale Coldfire-V1, the Intel® Atom™ Processor E6x5C Series, and the MP32, a 100% MIPS compatible processor.

Hard processor cores in FPGA

- Altera
  - Dual-Core ARM Cortex-A9: 800-MHz supporting symmetric and asymmetric multiprocessing, dual-issue superscalar pipeline (2.5 DMIPS per MHz) available in Cyclone V SoC FPGAs
- Xilinx
  - Dual-Core ARM Cortex-A9: Up to 800MHz, enhanced with NEON Extension and Single & Double Precision Floating point unit, 32kB Instruction & 32kB Data L1 Cache

FPGAs: The Problem of Power



Considering all of the power-saving options, it is common for a structured ASIC design to use 20 to 50 percent less power than the same design implemented in an FPGA.

Vince Hopkin, FPGA-to-ASIC conversion a crucial concern, *EE Times 13 Sept. 2004*

The Realistic Option: FPGAs

What would your boss think if you asked €3M for an ASIC?

- Nowadays, it is very difficult to have a project that economically justifies using ASICs.
- But MCUs are not a true SoC solution because there is always some peripheral missing (Murphy's law).

Example: Spartan-6

- Thirteen-member family delivers expanded densities ranging from 3,840 to 147,443 logic cells
- Built on a 45 nm low-power copper process technology
- Dual-register 6-input lookup table (LUT) logic
- System-level blocks
  - 18 Kb (2 x 9 Kb) block RAMs,
  - DSP48A1 slices,
  - SDRAM memory controllers,
  - enhanced mixed-mode clock management blocks, SelectIO™ technology,
  - Power-optimized high-speed serial transceiver blocks,
  - PCI Express® compatible Endpoint blocks,
  - advanced system-level power management modes,
  - auto-detect configuration options,
  - IP security with AES and Device DNA protection.
- Low cost (from 11$, www.digikey.com)

Example: Cyclone V

- Three targeted variants:
  1. Cyclone V E—for lowest system cost and power
  2. Cyclone V GX—for 614 Mbps to 3.125 Gbps transceiver applications
  3. Cyclone V GT—for 5.0 Gbps transceiver applications
- Key hard IP blocks include:
  - Hard memory controllers supporting 400 MHz DDR3 and DDR2 SDRAM, LPDDR, and LPDDR2 SDRAM
  - PCI Express Gen2 with multi-function, supporting up to eight peripherals
  - Variable precision DSP blocks
- Built on TSMC's 28-nm Low-Power process
- 8-input adaptive logic module (ALM), up to 12M of memory,
- Support partial reconfiguration
- Provide comprehensive design protection features
- All mainstream single-ended and differential I/O standards
- From $ 15 (www.digikey.com)

Design Methodology

- Design an embedded system is a challenging task:
  - Implement a lot of hardware components
  - Running complex software applications
- Three alternatives (design styles):
  - SoC design, when your competitive advantage is in the circuit design.
  - Embedded software design, when your competitive advantage is in the software.
  - Component based design, when you do not have the money to pay the design expenses, or the competitive advantage lies outside the electronics.

Platform-Based Design

- The basic idea behind the platform-based design approach is to avoid designing a chip from scratch.
- Some portion of the chip's architecture is predefined for a specific type of application:
  - There is a processor
  - A real-time operating system (RTOS)
  - Peripheral intellectual property (IP) blocks
  - Memory and a bus structure
- Users might customize it by adding hardware IP, programming FPGA logic or writing embedded software.

What is Platform-Based Design?

- It is neither a top-down nor a bottom-up methodology. It is a meet-in-the-middle design approach
- The starting point is a platform, that it is going to be customized:
  - Downwards when you add new peripherals or new HW.
  - Upwards when you write a code that uses all the components of the platform to create your application.
- It supports the embedded SW design style, because its final goal is to provide a SoC and OS that will be the starting point for the SW development.
  - It is not component based design, because this design style is just integration of components, including also SW components.

Advantages of Platform-Based Design

- The most important is the development time.
  - It allows us to create a SoC in a few days.
  - When designing with IPs, each component is solved but you still have to connect them.
  - This is solved with platform-based design.
- The implementation details of the SoC are abstracted, and you can center in the application development.
  - All the technology-dependent details are hidden, so you only have to customize the reference platform.
  - The development system will provide you the circuit and the libraries, OS and drivers needed to work with it.

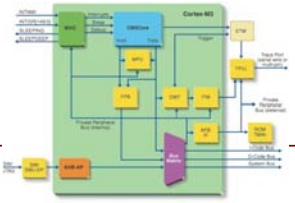
Example: Xilinx Platform Studio

- Is a powerful tool to design FPGA-based SoC platforms, using MicroBlaze or ARM Cortex A9.
- GUI: The peripherals are automatically connected when you add them, and it has a push-button implementation interface.
- Platform Studio also generates the SW libraries.
- It has also an integrated compiler and debugging system.

Example: Actel SmartDesign

- Block-based design creation tool for system-on-chip (SoC) systems, including SmartFusion and other processor and bus-based designs.
- SmartDesign is included in Libero IDE.
- SoftConsole: Facilitates software development for Actel's ARM® Cortex™–M3, Cortex-M1 and Core8051s microprocessors.
- Firmware Catalog: Streamlines the location and generation of firmware that is compatible with IP cores used in Actel FPGA designs.
- IAR Embedded Workbench: An IDE for compiling and debugging embedded ARM applications, including Cortex-M3 for SmartFusion, using assembly, C and C++.
- Keil MDK: An IDE for ARM Cortex-M3 containing the ARM C/C++ compiler, µVision®4 IDE debugger, MicroLib highly optimized run-time library, Real-Time Trace for Cortex-M3 processor-based devices and RTX RTOS.

Cortex
Intelligent Processors by ARM

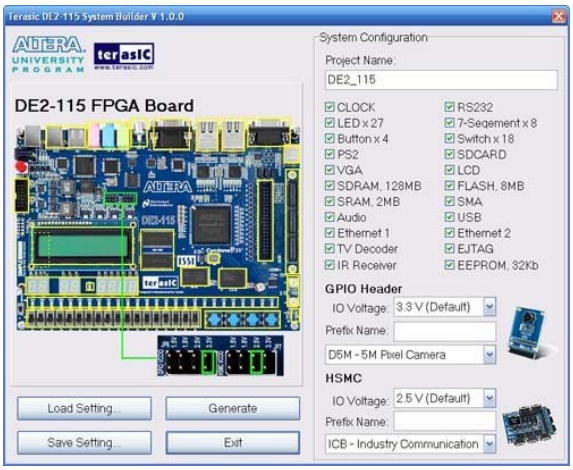


Example: Altera Qsys

Qsys

- Qsys is a modern SOPC Builder tool
  - Automatically generates interconnect logic to connect IP and subsystems.
  - Powered by a new FPGA-optimized network-on-a-chip (NoC) technology delivering higher performance, improved design reuse, and faster verification compared to SOPC Builder.

Development tools for boards



DE2-115 System Builder – a powerful tool that comes with the VEEK kit (originally developed for DE2-115).

This tool allows users to create customizable Quartus II projects depending on their requirements.

The top-level design file, pin assignments, and I/O standard settings for the DE2-115 board will be generated automatically from this tool.

In addition, through the GPIO and HSMC connectors you can select various daughter cards in conjunction with the tPad using the DE2-115 System Builder.

Summary… What are the Best Options for an Embedded System?

- Simple 32-bit microprocessors
  - 8-bit processors have old architectures and lack powerful development tools.
  - Higher-end processors might be too difficult to understand, it is a nonsense to use them for simple apps.
- FPGAs
  - ASICs are too expensive.
- Platform-based design
  - The only way to end the project in a reasonable time.
  - Avoids having to learn too much about the underlying technology.

So Guess What
You are Going to Study

FPGAs

Nios II - MicroBlaze

Altera Qsys (replaced EDS) - Xilinx EDK

Embedded Systems on a Chip: Introduction

Sergio Lopez-Buedo, Elías Todorovich
etodorov@exa.unicen.edu.ar
July 2013