


**CS 211: Computer Architecture**  
**course wrapup: goodbye to cs211!**

Instructor: Prof. Bhagi Narahari  
Dept. of Computer Science  
Course URL: [www.seas.gwu.edu/~narahari/cs211/](http://www.seas.gwu.edu/~narahari/cs211/)



**Recall from Course Intro... Perspective**

---

- **Computer architecture design is directly linked to underlying technology**
  - Semiconductor
  - Compiler technology
  - Computational models
- **Goal of software designers is to run an application program efficiently on the architecture**
  - Compiler plays a key role
  - what is the interplay between architecture features and application program properties
  - Bottom line is performance of application


CS 211: Computer Architecture, Bhagi Narahari



---

**Trends In Technology,  
Applications, Architectures**

CS 211: Computer Architecture, Bhagi Narahari

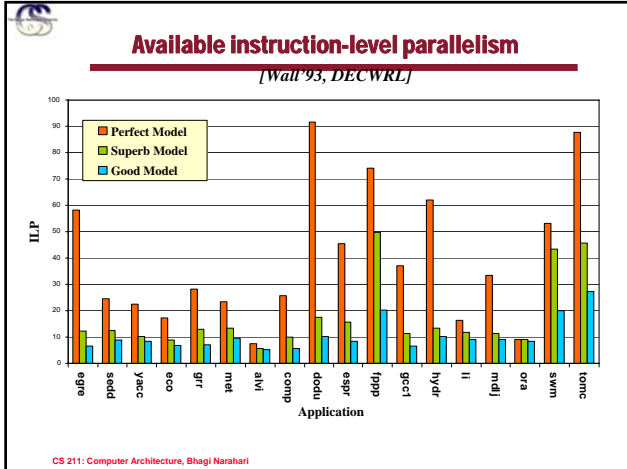


**Emerging trends in Semiconductor Technology: Recall from Week 1!**

---

- **What's the trend in Semiconductor technology and its impact on new types of processor architectures ?**
  - **some aspects to consider:**
    - Delay: switching time of transistor – impacts clock cycle
    - Feature size: size of transistor – impacts amount of logic in processor
    - Interconnect delay: clock cycle/delay in sending signal across the interconnect lines on a chip

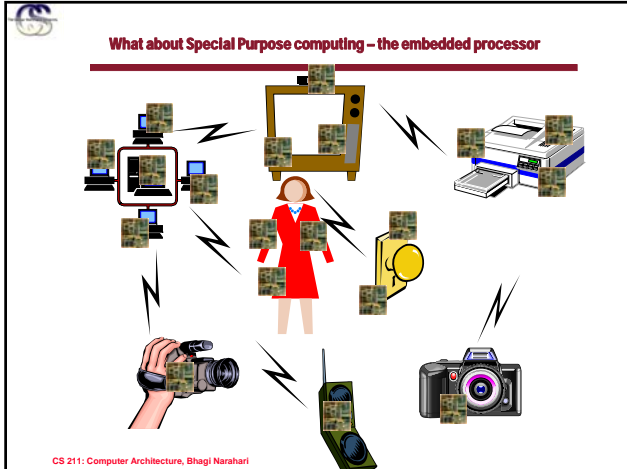
CS 211: Computer Architecture, Bhagi Narahari



- ### To summarize...
- **Lots of hardware parallelism available**
    - can accommodate approx. 50 pentiums on one die in few years
- However,
- **Conventional architectures and compilation**
    - cannot expose enough parallelism in applications
    - even the “superb” model yields an ILP < 10 on average
  - **Need for new architectures and compilation techniques!**
- CS 211: Computer Architecture, Bhagi Narahari

- ### Emerging Architecture Directions
- **ILP Architectures: This has been the focus of our discussions !**
  - **ILP Wall leads to Multithreaded Processors**
    - Multi-Core Processors
  - **Reconfigurable Processors—better for special purpose applications**
    - let compiler handle everything
    - no commitment to a particular architecture
    - compiler generates architecture and code for it
    - Example: FPGA based processors
- CS 211: Computer Architecture, Bhagi Narahari

- ### What “applications” have we considered ?
- **Focus has been on processor design for general purpose computing**
  - **Problems encountered during design of processors for general purpose computing**
    - Workstations, no power limitation, no size limitation...sometimes cost not an issue to begin with
- CS 211: Computer Architecture, Bhagi Narahari



## Glimpse into alternate approaches

---

- **Reconfigurable Architectures?**
  - We have looked at fixed ISA and fixed datapath/logic processors..
  - Can we design architectures that can be reconfigured to meet the application requirements?
    - What new problems/challenges?
    - Embedded Systems: What ? Challenges
  - Field Programmable Gate Arrays (FPGA) as an example of reconfigurable architectures
- **Embedded Systems**
  - What?
  - Challenges in design
    - What do we need to do differently from what we have studied thus far

CS 211: Computer Architecture, Bhagi Narahari

---

- **overview of Reconfigurable computing and FPGAs**
  - FPGAs are the enabling technology today
- **Overview of embedded systems design issues**
  - Want to learn more – take CS 297/CS 190

CS 211: Computer Architecture, Bhagi Narahari

## What is Reconfigurable Computing?

---

- **Computation using hardware that can adapt at the logic level to solve specific problems**
- **Why is this interesting?**
  - Some applications are poorly suited to microprocessor.
  - VLSI “explosion” provides increasing resources.
  - Hardware/Software codesign
  - Relatively new research area.

CS 211: Computer Architecture, Bhagi Narahari

### Microprocessor-based Systems

> Generalized to perform many functions well.  
 > Operates on fixed data sizes.  
 > Inherently sequential.

CS 211: Computer Architecture, Bhagi Narahari

### Reconfigurable Computing

```

If (A > B) {
  H = A;
  L = B;
}
Else {
  H = B;
  L = A;
}
  
```

> Create specialized hardware for each application.  
 > Functional units optimized to perform a special task.

CS 211: Computer Architecture, Bhagi Narahari

### Reconfigurable Hardware

$A \wedge B \wedge C \wedge D = \text{out}$

> Each logic element operates on four one-bit inputs.  
 > Output is one data bit.  
 > Can perform any boolean function of four inputs

$2^{(2^4)} = 64\text{K functions!}$

CS 211: Computer Architecture, Bhagi Narahari

### Field-Programmable Gate Array (FPGA)

> Each *logic element* outputs one data bit.  
 > Interconnect programmable between elements.  
 > Interconnect *tracks* grouped into channels.

CS 211: Computer Architecture, Bhagi Narahari

### Two Bit Adder

Made of Full Adders

$A + B = D$

Logic synthesis tool reduces circuit to SOP form

$$S = \bar{A}\bar{B}C_i + \bar{A}B\bar{C}_i + A\bar{B}\bar{C}_i + ABC_i$$
$$C_o = \bar{A}B\bar{C}_i + \bar{A}BC_i + A\bar{B}C_i + ABC_i$$

CS 211: Computer Architecture, Bhagi Narahari

### FPGA Architecture Issues

- > Need to explore architectural issues.
- > How much functionality should go in a logic element?
- > How many routing tracks per channel?
- > Switch "population"?
- > Cost of reconfiguration

CS 211: Computer Architecture, Bhagi Narahari

### Translating a Design to an FPGA

*C program*

$C = A + B$

*Circuit*

*Array*

- > CAD to translate circuit from text description to physical implementation well understood.
- > CAD to translate from C program to circuit not well understood.
- > Very difficult for application designers to successfully write high-performance applications

**Need for design automation!**

CS 211: Computer Architecture, Bhagi Narahari

### High-level Compilers

- > Difficult to estimate hardware resources.
- > Some parts of program more appropriate for processor (hardware/software codesign).
- > Compiler must parallelize computation across many resources.
- > Engineers like to write in C rather than pushing little blocks around.

$C = A + B$

...

for (i = 0; i < n, i++)

{

  .

  .

}

CS 211: Computer Architecture, Bhagi Narahari

## Processor + FPGA Three possibilities

1. FPGA serves as coprocessor for data intensive applications.

2. FPGA serves as embedded computer for low latency transfer. **“Reconfigurable Functional Unit”**

CS 211: Computer Architecture, Bhagi Narahari

## Processor + FPGA (cont.)

### 3. Processor integration

- FPGA logic embedded inside processor.
- A number of problems with 2 and 3.
  - Process technology an issue.
  - ALU much faster than FPGA generally.
  - FPGA much faster than the entire processor.

CS 211: Computer Architecture, Bhagi Narahari

## Dynamic Reconfiguration

- What if I want to exchange part of the design in the device with another piece?
- Need to create architectures and software to incrementally change designs.
- Effectively a “configuration cache”

Examples: encryption, filtering.

CS 211: Computer Architecture, Bhagi Narahari

## Dynamic Reconfiguration: Challenges ?

- Ability to change the configuration/functionality at run-time
  - Different local configurations can be thought of as instructions
  - Minimizing the number and size of instructions a key to successfully achieving efficient design
- Resources must be directed to do different things at different times through instructions.
- What is the overhead of reconfiguration ?
  - Is it worth it ?
- Do we reconfigure during run-time or statically ?
- Partial reconfigurability: reconfigure part of the logic while keeping the rest running
  - Emerging designs will support this model ?
- What applications benefit from this dynamic reconfigurability ?
- What software challenges are presented ?

CS 211: Computer Architecture, Bhagi Narahari

**Definition – Hardware/Software Co-Design**

A design methodology supporting the cooperative and concurrent development of hardware and software (co-specification, co-development, and co-verification) in order to achieve shared functionality and performance goals for a combined system<sup>1</sup>.

+ economic goals

1. Gupta, R. and De Micheli, G., "Hardware-Software Cosynthesis for Digital Systems," *IEEE Design & Test of Computers*, September 1993, pp. 29-41.

CS 211: Computer Architecture, Bhagi Narahari

**Objective**

The target is to develop a methodology for performing hardware and software development, fabrication and support cost modeling concurrent with hardware/software co-design.

- not possible to put everything into hardware due to resource limitations
- Some things done better in hardware if we can get hardware configured to the application

CS 211: Computer Architecture, Bhagi Narahari

**Partitioning Analysis**

> Result of compilation is synthesizable HDL and assembly code for the processor

> Compiler profiler determines dependence and rough performance estimates

```

graph TD
    A[C or C++ specification] --> B[Compiler inserts profiling marks]
    B --> C[Execute the program to collect profiling information]
    C --> D[Local Candidate Preselection]
    D --> E[Estimate hardware implementation and calculate speedup]
    E --> F[Perform partitioning]
    F --> G[Compiler generates Code]
    G --> H[Behavioural VHDL]
    G --> I[Assembly code]
  
```

CS 211: Computer Architecture, Bhagi Narahari

**Embedded Systems**

George Washington University

### Definition: Embedded System

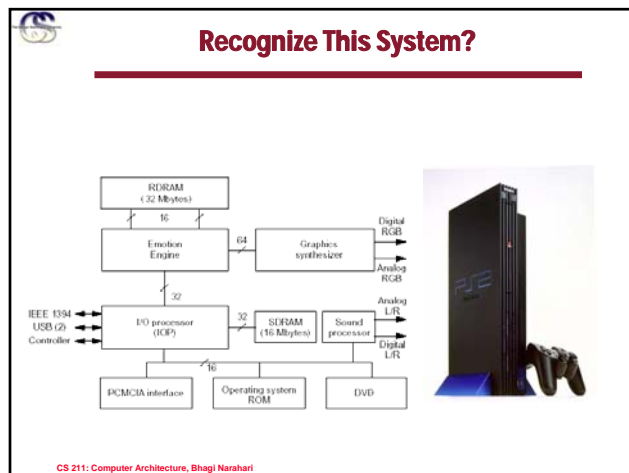
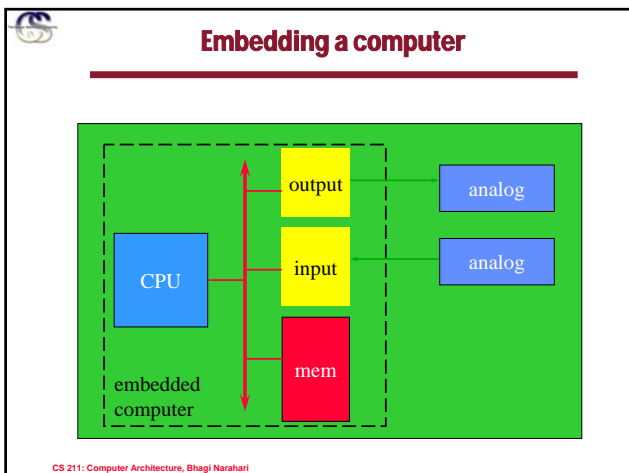
- **Embedded system:** any device that includes a programmable computer but is not itself a general-purpose computer.
- Take advantage of application characteristics to optimize the design:
  - don't need all the general-purpose bells and whistles.
  - Constrained system resources
  - Minimized human-machine interface (if at all)
  - Singly-focused application that runs when power is applied and terminates when power source is turned off or depleted.

CS 211: Computer Architecture, Bhagi Narahari

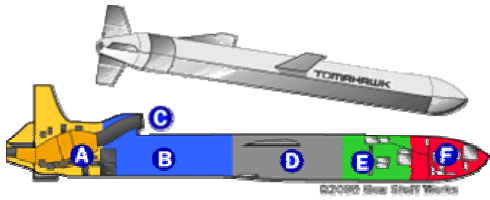
### Modern Embedded Systems

- Personal digital assistant (PDA).
- Consumer electronics (e.g. digital cameras and household appliances)
- Cell phone
- Automobile engines, fuel control, etc.
- Global Positioning System (GPS) units
- Printers
- Home automation systems
- Manufacturing plants – process control

CS 211: Computer Architecture, Bhagi Narahari



**Cruise Missile Guidance**



The diagram shows a cruise missile with a yellow nose section labeled 'A', a blue section labeled 'B', a grey section labeled 'D', a green section labeled 'E', and a red section labeled 'F'. A yellow section labeled 'C' is shown above the blue section. The text '©2000 Blue Staff Works' is visible at the bottom of the diagram.

CS 211: Computer Architecture, Bhagi Narahari

**Examples: Minimized Human-Machine Interface**

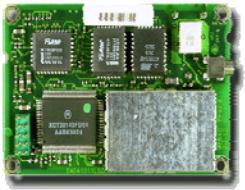


The image shows two examples of minimized human-machine interfaces: an ATM on the left and a vending machine on the right.

CS 211: Computer Architecture, Bhagi Narahari

**Singly-Focused Application**

- This Global Positioning System (GPS) module receives a signal from a satellite and calculates its geographic position.
- It may be used as a component in a larger embedded system application.
- Notice...no on/off switches!!



The image shows a small green printed circuit board (PCB) which is a GPS module. It features several integrated circuits, including a prominent square chip in the center, and various electronic components.

CS 211: Computer Architecture, Bhagi Narahari

**A little history**

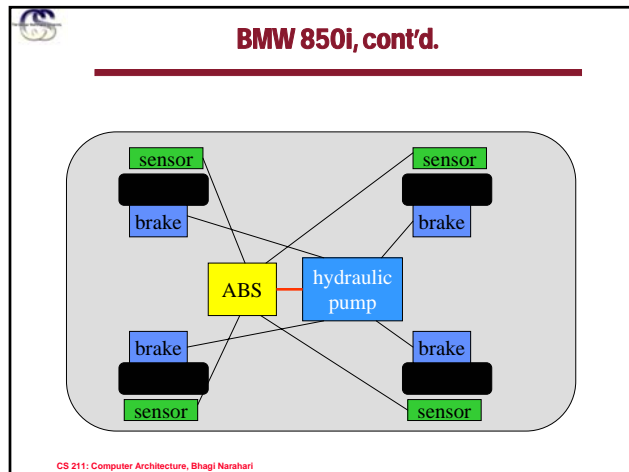
- Automobiles used microprocessor-based engine controllers starting in 1970's.
  - Control fuel/air mixture, engine timing, etc.
  - Multiple modes of operation: warm-up, cruise, hill climbing, etc.
  - Provides lower emissions, better fuel efficiency.
- First microprocessor was Intel 4004 in early 1970's
- What types of processors account for over 95% of all processors sold today ?

CS 211: Computer Architecture, Bhagi Narahari

**BMW 850i brake and stability control system**

- **Anti-lock brake system (ABS):** pumps brakes to reduce skidding.
- **Automatic stability control (ASC+T):** controls engine to improve stability.
- **ABS and ASC+T communicate.**
  - ABS was introduced first---needed to interface to existing ABS module.
- **Action based on data collected in real-time from sensors**
  - Note need to integrate sensors into your system!

CS 211: Computer Architecture, Bhagi Narahari



**Characteristics of embedded systems**


- **Sophisticated functionality**
  - Often run complex algorithms; eg. cell phones
- **Real-time operation.**
  - Hard real-time constraints = system fails if time deadlines are not met...(ABS)
- **Low manufacturing cost – mass market items**
  - Simple processors; small memory, microcontrollers, DSPs
- **Low power.**
  - Battery powered devices; battery size
- **Designed to tight deadlines by small teams.**

CS 211: Computer Architecture, Bhagi Narahari

**Challenges in embedded system design**

- **How much hardware do we need?**
  - How big is the CPU? Memory?
- **How do we meet our deadlines?**
  - Faster hardware or smarter software?
- **How do we minimize power?**
  - Turn off unnecessary logic? Reduce memory accesses?

CS 211: Computer Architecture, Bhagi Narahari




## Program design and analysis for Embedded Systems

---

- Optimizing for execution time.
- Optimizing for energy/power.
- Optimizing for program size.

CS 211: Computer Architecture, Bhagi Narahari




## Energy/power optimization

---

- **Energy**: ability to do work.
  - Most important in battery-powered systems.
- **Power**: energy per unit time.
  - Important even in wall-plug systems---power becomes heat.

CS 211: Computer Architecture, Bhagi Narahari




## Power Equation

---


$$P_{AVG} = \frac{1}{2} N_G f_{clk} C_L V_{DD}^2$$

- $P_{AVG}$  - the average power consumed by the gates
- $N_G$  - the number of gates that transition
- $f_{clk}$  - the frequency of the system clock
- $C_L$  - the average capacitive load per gate
- $V_{DD}$  - the supply voltage

CS 211: Computer Architecture, Bhagi Narahari




## Considerations

---

- Which techniques are appropriate for incorporation into a compiler?
  - Code Optimization
  - Clock Gating
  - Voltage Scaling
  - Bit Correlation
  - Memory power consumption
- What other benefits can be derived from these techniques?
  - Heat and component reliability
  - EMI noise reduction

CS 211: Computer Architecture, Bhagi Narahari




## Issues in Compiling for Embedded Systems

---

- Code Generation for Specialized Architecture
- Code size
  - Using techniques for performance optimizations, such as loop unrolling etc., can increase code size
- Timing requirements
  - To get precise timing may need to use assembly language
  - Not always “fast as you can” – may need minimum time, and duration
- Energy/Power optimization
  - Can we control power through software??
  - Instruction power – use of low power instructions
  - Dynamic voltage scaling
  - Memory power optimization
    - Placement of data
    - Dynamic power control of memory modules
    - For  $i=1$  to  $N$   $A[i]=A[i]^2$ ; For  $j=1$  to  $N$   $B[j]=B[j]+10$ ;

CS 211: Computer Architecture, Bhagi Narahari




## Key Insights: Literature

---

- Do all type of ALU instructions consume same amount of power ?
  - Run different type of instructions and collect power numbers
  - **Observed that add/sub take much less than multiply**
  - Try to replace multiply instructions with series of additions and shifts (strength reduction)
  - However, care must be taken since time penalty incurred can offset savings in energy!

CS 211: Computer Architecture, Bhagi Narahari




## Key Insights: Memory

---

- Do number of accesses to memory matter
  - Major impact on timing and therefore energy
- Does location in memory play a role ?
  - Effect on number of memory modules that need to be “active”
  - Placing data into memory can be used to switch memory modules on or off and have big savings in power. -- Levy, Crilly, Narahari, 2000

CS 211: Computer Architecture, Bhagi Narahari




## Optimizing for program size

---

- Goal:
  - reduce hardware cost of memory;
  - reduce power consumption of memory units.
- Two opportunities:
  - data;
  - instructions.

CS 211: Computer Architecture, Bhagi Narahari




### Data size minimization

---

- Reuse constants, variables, data buffers in different parts of code.
  - > Requires careful verification of correctness.
- Generate data using instructions.
- May need to limit amount of optimization
  - > Example of loop unrolling

CS 211: Computer Architecture, Bhagi Narahari




### Reducing code size

---

- Avoid function inlining.
- Choose CPU with compact instructions.
- Use specialized instructions where possible.

CS 211: Computer Architecture, Bhagi Narahari




### Adding logic to a board

---

- **Programmable logic devices (PLDs)** provide low/medium density logic.
- **Field-programmable gate arrays (FPGAs)** provide more logic and multi-level logic.
- **Application-specific integrated circuits (ASICs)** are manufactured for a single purpose.

CS 211: Computer Architecture, Bhagi Narahari




### What is an SoC?

---

- A System on a Chip is a collection of components that are designed as a system and implemented in a fashion similar to that of an Application Specific Integrated Circuit (ASIC)
- The goal is to pack as much functionality into a single die as possible

CS 211: Computer Architecture, Bhagi Narahari




## Summary

---

- **Embedded computers are all around us.**
  - Many systems have complex embedded hardware and software.
- **Embedded systems pose many design challenges: design time, deadlines, power, etc.**
- **Relevance to Software designers?**
  - Trend is towards software development environments
    - May no longer need sophisticated hardware designers to build embedded systems
- **More on Embedded Systems – take CS297 in Spring 2009**

CS 211: Computer Architecture, Bhagi Narahari




## Goodbye to CS 211

---

- **Computer Architecture: key components and interplay between hardware and software**
- **Performance of application in terms of architecture features**
- **Improving performance via new classes of processors**
  - Instruction level parallelism (ILP)
    - Superscalar and EPIC
  - ILP Wall leads to Multi-Core Processors
    - Multiprocessing on a chip
- **Impact of memory organization on program performance**
- **Role of compiler**
- **Conclusions ??**
  - Many parameters to choose from: application characteristics need to be taken into account

CS 211: Computer Architecture, Bhagi Narahari



## Course Trivia...

---

- **Project 2 due this week**
- **Exam 2 next Tuesday**
  - Focus on material after mid-term
    - But need to remember enough from first part of the course
  - Similar format to Exam 1
    - Multiple choice, short answers, and some detailed analysis
- **Project 3 due December 9<sup>th</sup>**
  - One set of experiments per team
  - Each team member writes their own report
    - Yes, you can reach different conclusions with the same data!

CS 211: Computer Architecture, Bhagi Narahari