Subject: Embedded Real-time System (EE104IU)

- ***Course Description:*** Development of microcontroller-based systems for embedded applications. Interfacing to input/output peripherals such as displays, keypads, sensors, digital-to-analog and analog-to-digital converters, and communication devices among others. Emphasizes hardware and software design.

- ***Objectives:*** At the end of the course the students are expected to know how to specify, design, and prototype a controller-based embedded system. To achieve this objective the students have to develop a project consisting of specifying, designing, and prototyping an embedded system solution to a real life problem.

cuu duong than cong . com

References

- F. Vahid & T. Givargis, “EMBEDDED SYSTEM DESIGN: A Unified Hardware Software Introduction” John Wiley & Sons, Inc. 2002
- Sid Katzen, The Quintessential PIC Microcontroller, Springer 2000
- **Martin Bates, Interfacing PIC Microcontrollers, Elsevier 2006**
- **Nigel Gardner, An introduction to programming the Microchip PIC in CCS C, Bluebird Electronics 2002.**
- Tim Wilmshurst, Designing Embedded Systems with PIC Microcontrollers —Principles and applications, Elsevier 2007
- A. Gaonkar, "Fundamentals of Microcontrollers and Applications In Embedded Systems: With the PIC18 Microcontroller Family", Thomson Delmar Learning, 2007

Embedded Systems on the Web (by Srivastava)

- Berkeley Design technology, Inc.: http://www.bdti.com
- EE Times Magazine: http://www.eet.com/
- Linux Devices: http://www.linuxdevices.com
- Embedded Linux Journal: http://embedded.linuxjournal.com
- Embedded.com: http://www.embedded.com/
  - *Embedded Systems Programming* magazine
- Circuit Cellar: http://www.circuitcellar.com/
- Electronic Design Magazine: http://www.planetee.com/ed/
- Electronic Engineering Magazine: http://www2.computeroemonline.com/magazine.html
- Integrated System Design Magazine: http://www.isdmag.com/
- Sensors Magazine: http://www.sensorsmag.com
- Embedded Systems Tutorial: http://www.learn-c.com/
- Collections of embedded systems resources
  - http://www.ece.utexas.edu/~bevans/courses/ee382c/resources/
  - http://www.ece.utexas.edu/~bevans/courses/realtime/resources.html
- Newsgroups
  - comp.arch.embedded, comp.cad.cadence, comp.cad.synthesis, comp.dsp, comp.realtime, comp.software-eng, comp.speech, and sci.electronics.cad

cuu duong than cong . com

Grading

• Home works:	15%
• Project (2-3 students/project):	15%
• Midterm exam:	20%
• Final Exam:	50%

Embedded Systems Design: A Unified Hardware/Software Introduction

Chapter 1: Introduction

cuu duong than cong . com

Outline

- Embedded systems overview
  - What are they?
- Design challenge – optimizing design metrics
- Technologies
  - Processor technologies
  - IC technologies
  - Design technologies

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Future of IT?

•According to forecasts characterized by the terms such as

- *Post-PC era*
- *Disappearing computer*
- *Ubiquitous computing*
- *Pervasive computing*
- *Ambient intelligence*
- *Embedded systems*

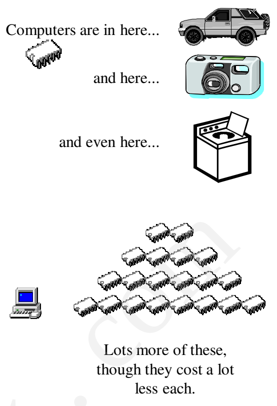
cuu duong than cong . com

Embedded systems overview

- Computing systems are everywhere
- Most of us think of "desktop" computers
  - PC's
  - Laptops
  - Mainframes
  - Servers
- But there's another type of computing system
  - Far more common...

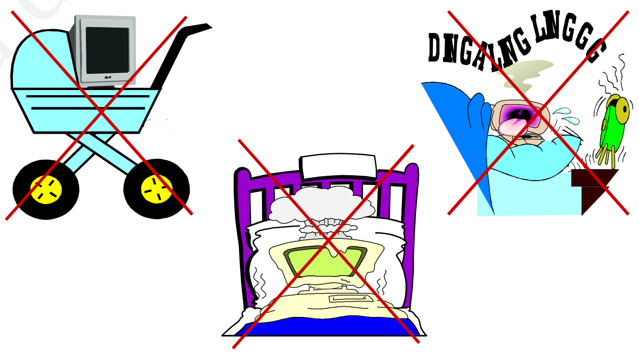
Embedded systems overview

- Embedded computing systems
  - Computing systems embedded within electronic devices
  - Hard to define. Nearly any computing system other than a desktop computer
  - Billions of units produced yearly, versus millions of desktop units
  - Perhaps 50 per household and per automobile



cuu duong than cong . com

What is an embedded system?



Embedded systems (ES) = **information processing systems embedded into a larger product**

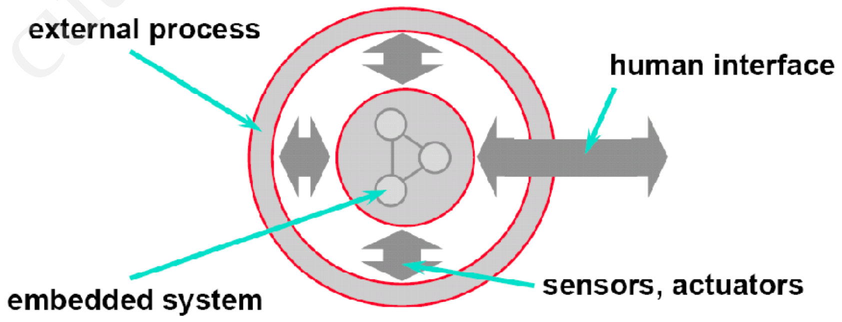
!

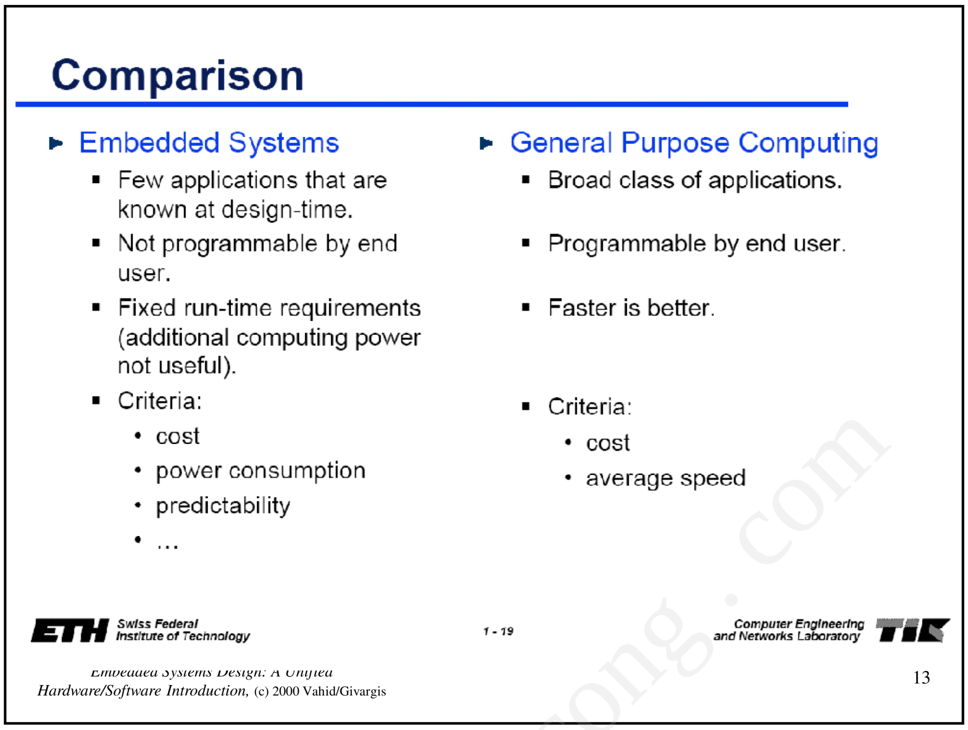
Definition

- **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.

cuu duong than cong . com

Embedded Systems

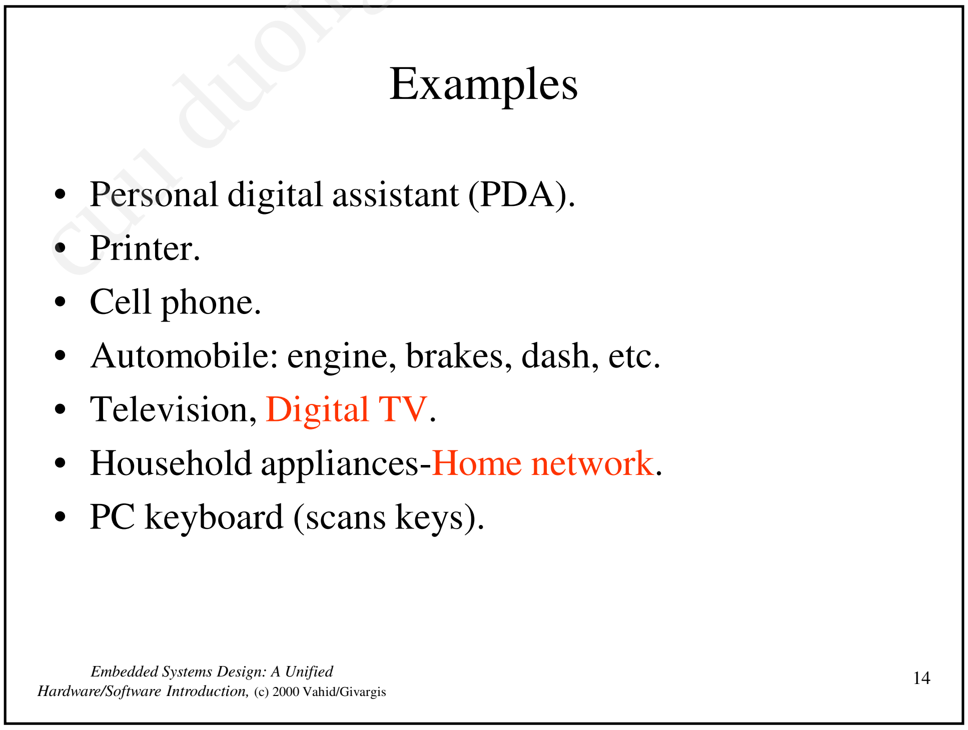




Comparison

- Embedded Systems
  - Few applications that are known at design-time.
  - Not programmable by end user.
  - Fixed run-time requirements (additional computing power not useful).
  - Criteria:
    - cost
    - power consumption
    - predictability
    - …
- General Purpose Computing
  - Broad class of applications.
  - Programmable by end user.
  - Faster is better.
  - Criteria:
    - cost
    - average speed

Swiss Federal Institute of Technology 1 - 19 Computer Engineering and Networks Laboratory



Examples

- Personal digital assistant (PDA).
- Printer.
- Cell phone.
- Automobile: engine, brakes, dash, etc.
- Television, Digital TV.
- Household appliances-Home network.
- PC keyboard (scans keys).

cuu duong than cong . com

Application examples

- Simple control: front panel of microwave oven, etc.
- Canon EOS 3 has three microprocessors.
  - 32-bit RISC CPU runs autofocus and eye control systems.
- Analog TV: channel selection, etc.
- Digital TV: programmable CPUs + hardwired logic.

cuu duong than cong . com

A “short list” of embedded systems

Anti-lock brakes
Auto-focus cameras
Automatic teller machines
Automatic toll systems
Automatic transmission
Avionic systems
Battery chargers
Camcorders
Cell phones
Cell-phone base stations
Cordless phones
Cruise control
Curbside check-in systems
Digital cameras
Disk drives
Electronic card readers
Electronic instruments
Electronic toys/games
Factory control
Fax machines
Fingerprint identifiers
Home security systems
Life-support systems
Medical testing systems

Modems
MPEG decoders
Network cards
Network switches/routers
On-board navigation
Pagers
Photocopiers
Point-of-sale systems
Portable video games
Printers
Satellite phones
Scanners
Smart ovens/dishwashers
Speech recognizers
Stereo systems
Teleconferencing systems
Televisions
Temperature controllers
Theft tracking systems
TV set-top boxes
VCR’s, DVD players
Video game consoles
Video phones
Washers and dryers

And the list goes on and on

CuuDuongThanCong.com https://fb.com/tailieudientucntt

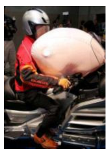
Application areas

1. Automotive electronics

2. Aircraft electronics

3. Trains

4. Telecommunication



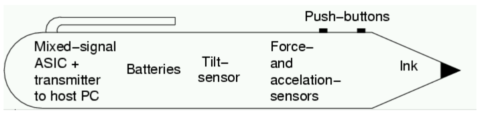
cuu duong than cong . com

Application areas

5. Medical systems
   e.g. "artificial eye"

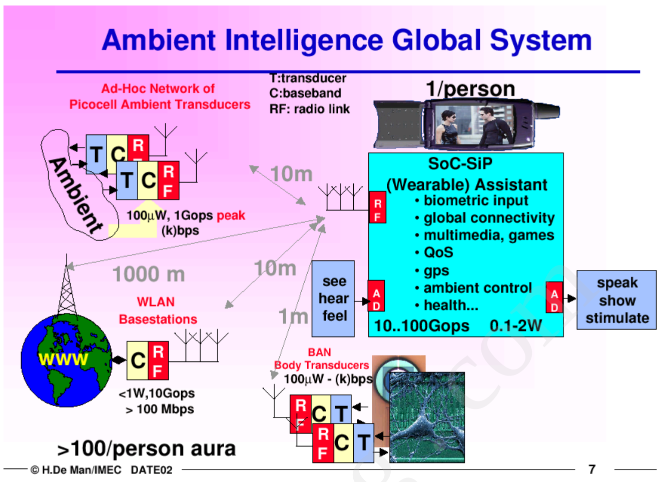
6. Military applications

7. Authentication


Push–buttons
Mixed–signal ASIC + transmitter to host PC
Batteries
Tilt–sensor
Force– and accelation–sensors
Ink

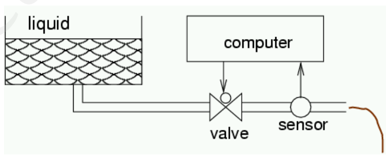
Application areas

8. Consumer electronics



Application areas

9. Fabrication equipment



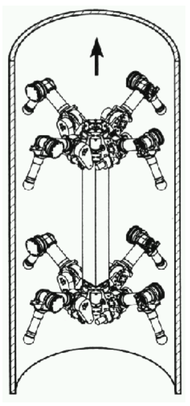
10. Smart buildings

cuu duong than cong . com

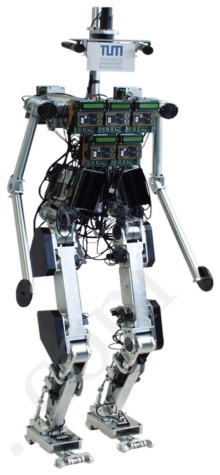
Application areas

11. Robotics

„Pipe-climber“



Robot „Johnnie“ (Courtesy and ©: H.Ulbrich, F. Pfeiffer, TU München)



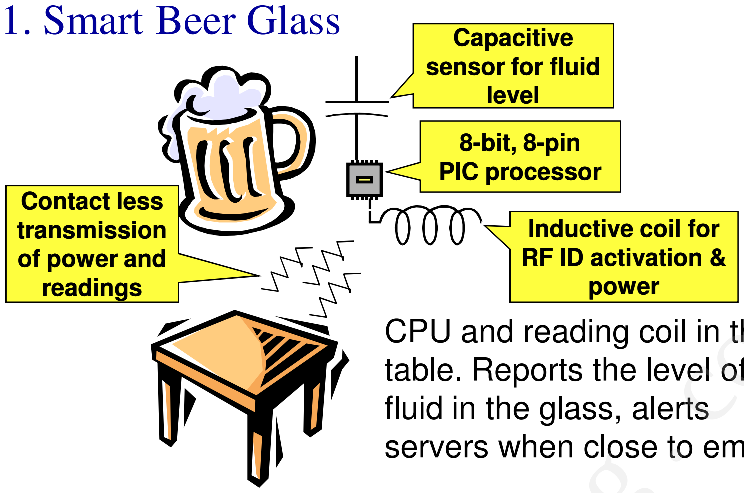
cuu duong than cong . com

Embedded systems from real life

- Typical embedded solution
- Integrates several technologies:
  - Radio transmissions
  - Sensor technology
  - Magnetic inductance for power
  - Computer used for calibration
- Impossible without the computer
- Meaningless without the electronics

Embedded systems from real life

1. Smart Beer Glass


Capacitive sensor for fluid level
8-bit, 8-pin PIC processor
Inductive coil for RF ID activation & power
Contact less transmission of power and readings

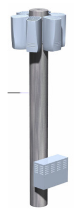
CPU and reading coil in the table. Reports the level of fluid in the glass, alerts servers when close to empty

cuuduongthancong.com

Embedded systems from real life

2. Mobile Phones and Base Stations

- Multiprocessor
  - 8-bit/32-bit for UI; DSP for signals
  - 32-bit in IR port; 32-bit in Bluetooth
- 8-100 MB of memory
- All custom chips



- Massive signal processing
  - Several processing tasks per connected call
- Based on DSPs
  - Standard or custom
  - 100s of processors

Embedded systems from real life

3. Sewing Machine

- User interface
  - Embroidery patterns
  - Touch-screen control
- ”Smart”
  - Sets pressure of foot depending on task
  - Raise foot when stopped
- New functions added by upgrading the software

cuu duong than cong . com

Embedded systems from real life

4. Cars

- Multiple processors
  - Up to 100
  - Networked together
- Multiple networks
  - Body, engine, telematics, media, safety
- Large diversity in processor types:
  - 8-bit – door locks, lights, etc.
  - 16-bit – most functions
  - 32-bit – engine control, airbags

- Functions by embedded processing:
  - ABS: Anti-lock braking systems
  - ESP: Electronic stability control
  - Airbags
  - Efficient automatic gearboxes
  - Theft prevention with smart keys
  - Blind-angle alert systems
  - ... etc ...

Embedded systems from real life

5. Extremely Large

- Functions requiring computers:
  - Radar
  - Weapons
  - Damage control
  - Navigation
  - basically everything
- Computers:
  - Large servers
  - 1000s of processors

cuu duong than cong . com

Embedded systems from real life

6. Inside Your PC

- Custom processors
  - Graphics, sound
- 32-bit processors
  - IR, Bluetooth
  - Network, WLAN
  - Harddisk
  - RAID controllers
- 8-bit processors
  - USB
  - Keyboard, mouse

Growing importance of embedded systems

- Growing economical importance of embedded systems: [www.itfacts.biz]
  - *Worldwide mobile phone sales surpassed 156.4 mln units in Q2 2004, a 35% increase from Q2 2003.*
  - *The worldwide portable flash player market exploded in 2003 and is expected to grow from 12.5 mln units in 2003 to over 50 mln units in 2008.*
  - *Global 3G subscribers will grow from an estimated 45 mln at the end of 2004 to 85 mln in 2005.*
  - *The number of broadband lines worldwide increased by almost 55% to over 123 mln in the 12 months to the end of June 2004.*
  - *Today's DVR (digital video recorders) users - 5% of households - will grow to 41% within five years.*
  - 79% of all high-end processors are used in embedded systems
  - The future is embedded, Embedded is the future!

Embedded Systems Design: A Unified Hardware/Software Introduction, (c) 2000 Vahid/Givargis

29

cuu duong than cong . com

Application Areas of Embedded Systems

Automatic chocolate vending machine	Anti-lock brakes	Auto-focus cameras	Automatic teller machines	Automatic toll-booth systems	Automatic transmission / reception	Avionic systems
Battery chargers	Camcorders	Cell phones	Cell-phone base stations	Cordless phones	Cruise control systems	Curbside check-in systems
Digital cameras	Disk drives	Data Acquisition systems	Electronic card / tape readers / drives	Electronic instruments	Electronic toys / games	Electronic Control Units (ECUs)
Factory control	Fax machines	Fingerprint identifiers		Home security systems	LCD TVs / Monitors	Life-support systems
Medical testing systems	Modems	MPEG decoders	Mobile Phones	Network cards	Network switches/ routers	Night vision cameras
On-board navigation		Pagers	Photocopiers	Point-of-sale systems	Portable video games	Printers
Robots	RFID Tags	Satellite phones	Scanners	Smart Ovens /dishwashers	Smart Rooms / Buildings	Speech recognizers
Stereo systems	Teleconferencing systems	Telemonitoring medical systems	Televisions	Temperature controllers	Theft tracking systems	
VCR's, DVD players	Video game consoles	Video phones		Washers and dryers	Wireless Devices	

And many more devices will be added - -> A never ending list

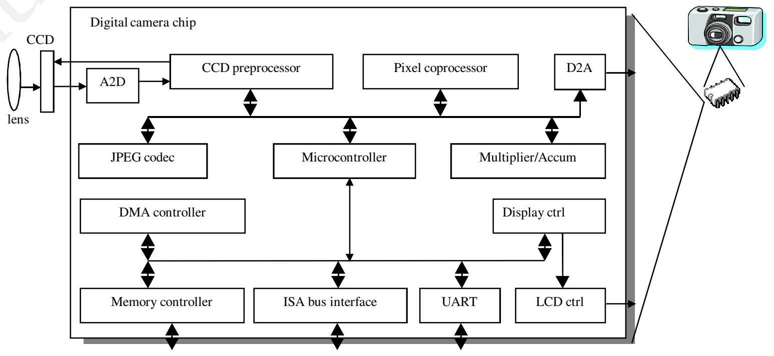
Embedded System Design (EEE G512), © 2009, Rajiv Ranjan Singh, BITS Pilani, India

Some common characteristics of embedded systems

- Single-functioned
  - Executes a single program, repeatedly
- Tightly-constrained
  - Low cost, low power, small, fast, etc.
- Reactive and real-time
  - Continually reacts to changes in the system's environment
  - Must compute certain results in real-time without delay

cuu duong than cong . com

An embedded system example -- a digital camera



- Single-functioned -- always a digital camera
- Tightly-constrained -- Low cost, low power, small, fast
- Reactive and real-time -- only to a small extent

Characteristics of Embedded Systems (1)

- Must be **dependable**,
  - **Reliability** ***R(t)*** **=** probability of system working correctly provided that is was working at *t*=0
  - **Maintainability** ***M(d)*** **=** probability of system working correctly *d* time units after error occurred.
  - **Availability** ***A*(*t*)**: probability of system working at time *t*
  - **Safety**: no harm to be caused
  - **Security**: confidential and authentic communication

  Even perfectly designed systems can fail if the assumptions about the workload and possible errors turn out to be wrong. Making the system dependable must not be an after-thought, it must be considered from the very beginning

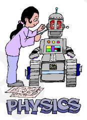
cuu duong than cong . com

Characteristics of Embedded Systems (2)

- Must be **efficient**
  - Energy efficient
  - Code-size efficient (especially for systems on a chip)
  - Run-time efficient
  - Weight efficient
  - Cost efficient
- **Dedicated** towards a certain **application**
  Knowledge about behavior at design time can be used to minimize resources and to maximize robustness
- **Dedicated user interface**
  (no mouse, keyboard and screen)
- **Hybrid systems** (analog + digital parts).

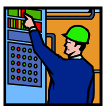
Characteristics of Embedded Systems (3)

- Many ES must meet **real-time constraints**
  - A real-time system must react to stimuli from the controlled object (or the operator) within the time interval **dictated** by the environment.
  - For real-time systems, right answers arriving too late are wrong.
  - **„A real-time constraint is called hard, if not meeting that constraint could result in a catastrophe“** [Kopetz, 1997].
  - All other time-constraints are called **soft**.
  - A guaranteed system response has to be explained without statistical arguments
- Frequently **connected to physical environment** through sensors and actuators,



cuu duong than cong . com

Characteristics of Embedded Systems (4)



- Typically, ES are **reactive systems**:
  „A reactive system is one which is in continual interaction with is environment and executes at a pace determined by that environment“ [Bergé, 1995]
  Behavior depends on input **and current state**.
  ☞ automata model appropriate,
  model of computable functions inappropriate.

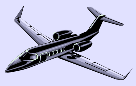
•Not every ES has all of the above characteristics.

•**Def**.: **Information processing systems having most of the above characteristics are called embedded systems.**

•Course on embedded systems makes sense because of the number of common characteristics.

Quite a number of challenges, e.g. dependability

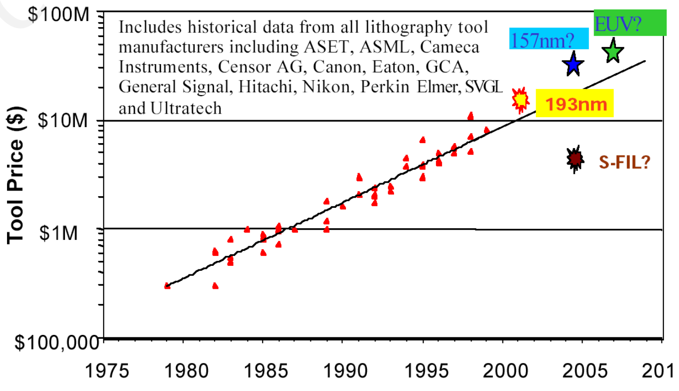
•Dependability?

- Non-real time protocols used for real-time applications (e.g. Berlin fire department)
- Over-simplification of models (e.g. aircraft anti-collision system) 
- Using unsafe systems for safety-critical missions (e.g. voice control system in Los Angeles; ~ 800 planes without voice connection to tower for > 3 hrs

cuu duong than cong . com

Challenges for implementation in hardware

- Lack of flexibility (changing standards).
- Mask cost for specialized HW becomes very expensive

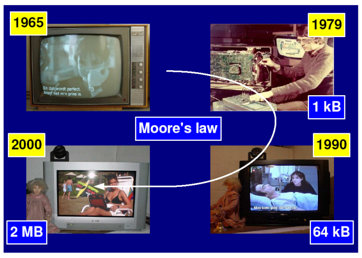


☞Trend towards implementation in Software

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Software complexity is a challenge

- Exponential increase in software complexity
- In some areas code size is doubling every 9 months [ST Microelectronics, Medea Workshop, Fall 2003]
- *... > 70% of the development cost for complex systems such as automotive electronics and communication systems are due to software development* [A. Sangiovanni-Vincentelli, 1999]



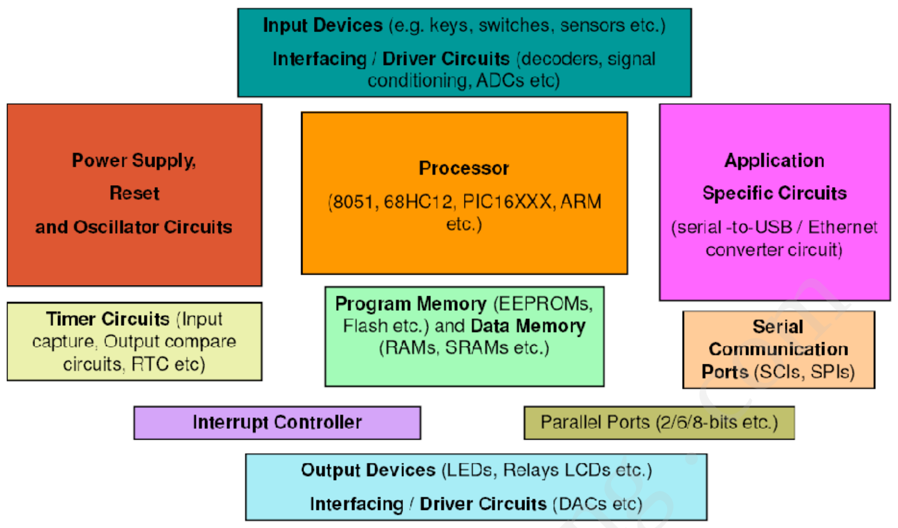
Rob van Ommering, COPA Tutorial, as cited by: Gerrit Müller: Opportunities and challenges in embedded systems, Eindhoven Embedded Systems Institute, 2004

cuu duong than cong

More challenges for embedded software

- Dynamic environments
- Capture the required behaviour!
- Validate specifications
- Efficient translation of specifications into implementations!
- How can we check that we meet real-time constraints?
- How do we validate embedded real-time software? (large volumes of data, testing may be safety-critical)

Components of Embedded Systems (Hardware)


Input Devices (e.g. keys, switches, sensors etc.)
Interfacing / Driver Circuits (decoders, signal conditioning, ADCs etc)
Power Supply, Reset and Oscillator Circuits
Processor (8051, 68HC12, PIC16XXX, ARM etc.)
Application Specific Circuits (serial-to-USB / Ethernet converter circuit)
Timer Circuits (Input capture, Output compare circuits, RTC etc)
Program Memory (EEPROMs, Flash etc.) and **Data Memory** (RAMs, SRAMs etc.)
Serial Communication Ports (SCIs, SPIs)
Interrupt Controller
Parallel Ports (2/6/8-bits etc.)
Output Devices (LEDs, Relays LCDs etc.)
Interfacing / Driver Circuits (DACs etc)

cuu duong than cong . com

Components of Embedded Systems (Software)

1) **Main application software:**
   - performs series of tasks or multiple tasks concurrently.
   - constrained due to low memory, low processing power requirement and power dissipation etc.
2. **Real Time Operating System (RTOS):**
   - supervises the application software.
   - provides a mechanism to let the processor run a process as per scheduling.
   - performs context-switching between various processes (tasks).
   - organizes access to a resource in sequence of the series of tasks of the system.
   - schedules their working and execution by following a plan to control the latencies and to meet the deadlines.
   - sets the rules during the execution of the application software.

Real-Time (1)

The Oxford dictionary of Computing offers this definition for real-time systems:

Any system in which the time at which the output is produced is significant. This is usually because the input corresponds to some movement in the physical world, and the output has to relate to that same movement. The lag from input time to output time must be sufficiently small for acceptable timeliness.

Real-Time system is defined as a system where the correctness of the system depends not only the result of computations but also on the time at which it is produced. Therefore the *time* is the most important item to be managed.

cuu duong than cong . com

Real-Time (2)

Definition in Laplante's book:

A real-time system is a system that must satisfy explicit (bounded) response-time constraints or risk severe consequences, including failure.

Failed System:
A failed system is a system which cannot satisfy one or more of the requirements laid out in the formal system specification.

Real-Time (3)

It can be argued that
all practical systems are real-time!

Hard Real-Time
Systems where failure to meet system response time constraints leads to a system failure are called hard real-time systems.

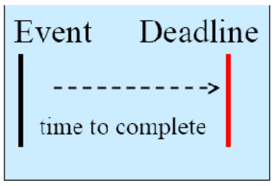
Soft Real-Time:
Systems where performance is degraded but not destroyed by failure to meet system response time constraints.

Firm Real-Time:
Systems with hard deadlines where some low probability of missing deadline can be tolerated.

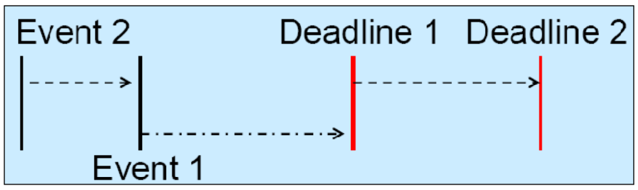
cuu duong than cong . com

Real-Time System?

- Correct behaviour is defined by:
  - Correctness of results, AND
  - *the time the result was delivered*
    - Not too late, sometimes also not *too early*
- Usually *reactive* systems
  - Reacts to an ”external” event
- *Predictability* is the most important criteria!



Real-Time System

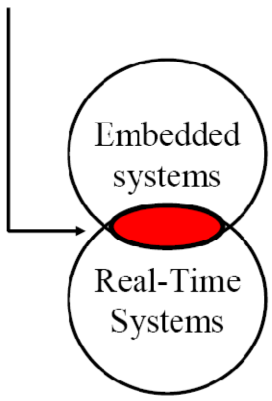


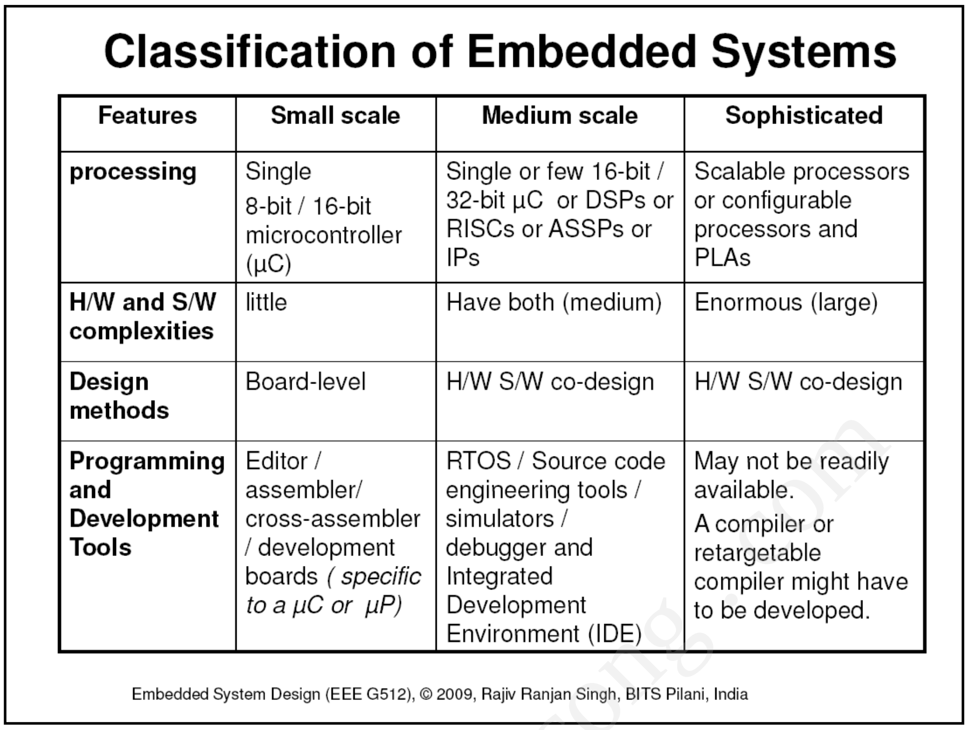
- Hard real-time:
  - **MUST** be in time
    - Otherwise, bad things will happen: money or lives lost, etc..
- Soft real-time
  - **SHOULD** be in time
    - Otherwise uncorrect behavior, but not disastrous
    - e.g. crackling in audio stream

cuu duong than cong . com

Embedded Real-Time Systems

- Many constraints from both domains
  - Predictability
  - Reliability
  - Fault tolerance
  - Low cost
  - Security
  - Restricted performance and memory conditions
  - Power consumption

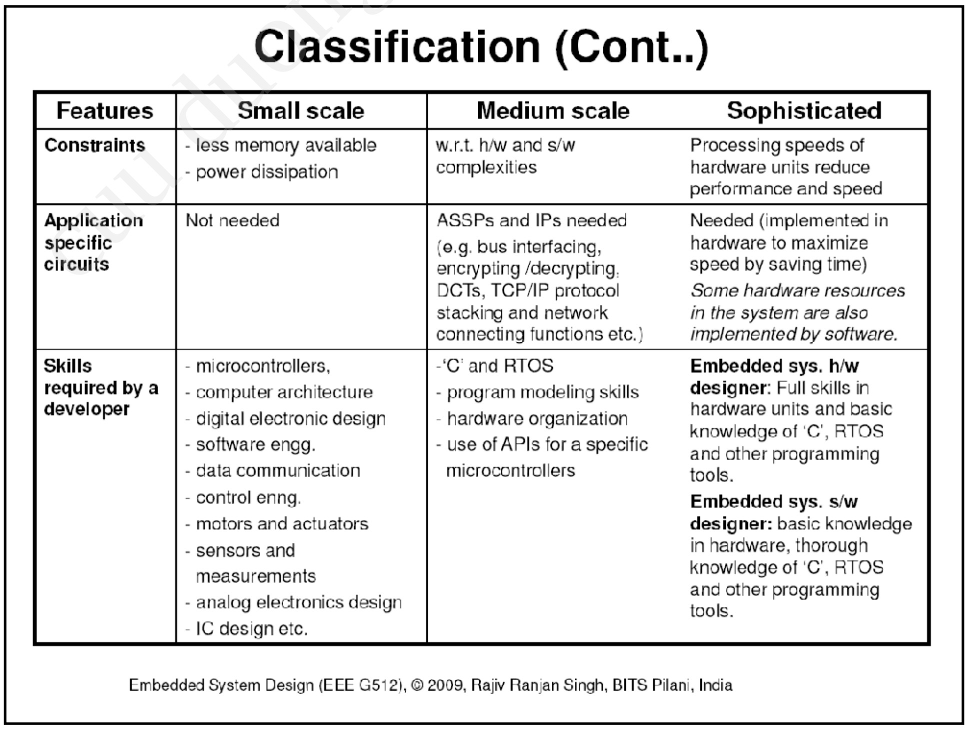




Classification of Embedded Systems

Features	Small scale	Medium scale	Sophisticated
processing	Single 8-bit / 16-bit microcontroller (µC)	Single or few 16-bit / 32-bit µC or DSPs or RISCs or ASSPs or IPs	Scalable processors or configurable processors and PLAs
H/W and S/W complexities	little	Have both (medium)	Enormous (large)
Design methods	Board-level	H/W S/W co-design	H/W S/W co-design
Programming and Development Tools	Editor / assembler/ cross-assembler / development boards *(specific to a µC or µP)*	RTOS / Source code engineering tools / simulators / debugger and Integrated Development Environment (IDE)	May not be readily available. A compiler or retargetable compiler might have to be developed.

Embedded System Design (EEE G512), © 2009, Rajiv Ranjan Singh, BITS Pilani, India



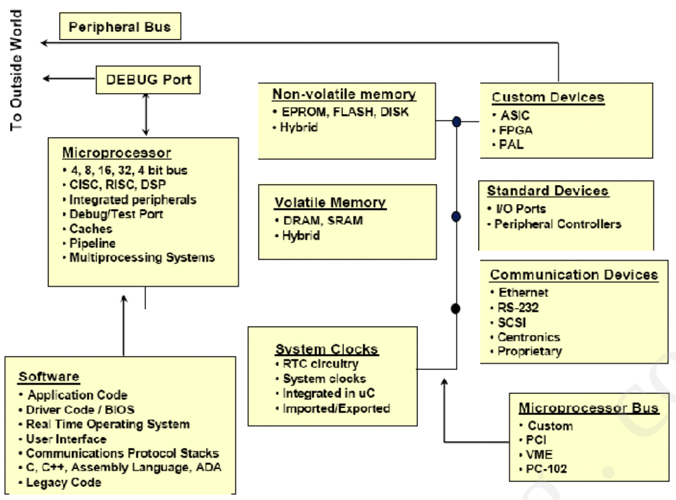
Classification (Cont..)

Features	Small scale	Medium scale	Sophisticated
Constraints	- less memory available - power dissipation	w.r.t. h/w and s/w complexities	Processing speeds of hardware units reduce performance and speed
Application specific circuits	Not needed	ASSPs and IPs needed (e.g. bus interfacing, encrypting /decrypting, DCTs, TCP/IP protocol stacking and network connecting functions etc.)	Needed (implemented in hardware to maximize speed by saving time) *Some hardware resources in the system are also implemented by software.*
Skills required by a developer	- microcontrollers, - computer architecture - digital electronic design - software engg. - data communication - control enng. - motors and actuators - sensors and measurements - analog electronics design - IC design etc.	-'C' and RTOS - program modeling skills - hardware organization - use of APIs for a specific microcontrollers	**Embedded sys. h/w designer**: Full skills in hardware units and basic knowledge of 'C', RTOS and other programming tools. **Embedded sys. s/w designer:** basic knowledge in hardware, thorough knowledge of 'C', RTOS and other programming tools.

Embedded System Design (EEE G512), © 2009, Rajiv Ranjan Singh, BITS Pilani, India

cuu duong than cong . com

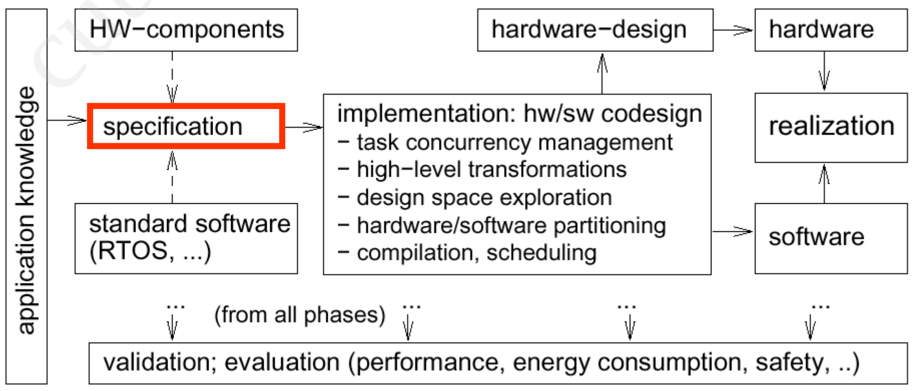
Typical Architecture



ETH Swiss Federal Institute of Technology 1 - 20 Computer Engineering and Networks Laboratory TIK

cuu duong than cong . com

Specifications



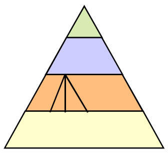
application knowledge
HW−components
specification
standard software (RTOS, ...)
hardware−design
implementation: hw/sw codesign
− task concurrency management
− high−level transformations
− design space exploration
− hardware/software partitioning
− compilation, scheduling
hardware
realization
software
...
(from all phases)
validation; evaluation (performance, energy consumption, safety, ..)

CuuDuongThanCong.com https://fb.com/tailieudientucntt

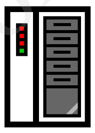
Specification of embedded systems: Requirements for specification techniques (1)

- **Hierarchy**
  Humans not capable to understand systems containing more than ~5 objects.

  Most actual systems require more objects
  ☞ Hierarchy
  – Behavioral hierarchy
    Examples: states, processes, procedures.
  – Structural hierarchy
    Examples: processors, racks, printed circuit boards



proc
  proc
    proc

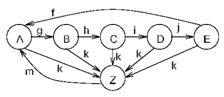


cuu duong than cong . com

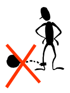
Specification of embedded systems: Requirements for specification techniques (2)

- **Timing behavior.**

- **State-oriented behavior**
  Required for reactive systems; classical automata insufficient.



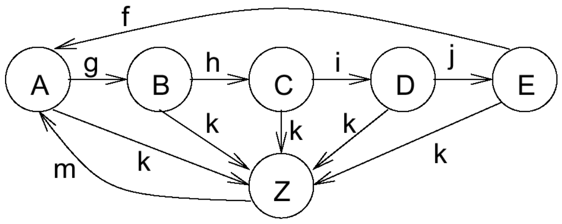
- **Event-handling**
  (external or internal events)
- **No obstacles for efficient implementation**



Requirements for specification techniques (3)

- **Support for the design of dependable systems**
  Unambiguous semantics, ...

- **Exception-oriented behavior**
  Not acceptable to describe exceptions for every state.



We will see, how all the arrows labeled k can be replaced by a single one.

cuu duong than cong . com

Requirements for specification techniques (4)

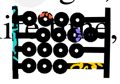
- **Concurrency**
  Real-life systems are concurrent
- **Synchronization and communication**
  Components have to communicate!
- **Presence of programming elements**
  For example, arithmetic operations, loops, and function calls should be available
- **Executability** (no algebraic specification)
- **Support for the design of large systems** (☞ OO)
- **Domain-specific support**

Requirements for specification techniques (5)

- **Readability**
- **Portability and flexibility**
- **Termination**
  It should be clear, at which time
  all computations are completed

- **Support for non-standard I/O devices**
  Direct access to switches, displays, ...

- **Non-functional properties**
  fault-tolerance, disposability, EMC-properties, weight, size, user friendliness, extendibility, expected life time, power consumption...



- **Adequate model of computation**

cuu duong than cong . com

Design challenge – optimizing design metrics

- Obvious design goal:
  - Construct an implementation with desired functionality
- Key design challenge:
  - Simultaneously optimize numerous design metrics
- Design metric
  - A measurable feature of a system’s implementation
  - Optimizing design metrics is a key challenge

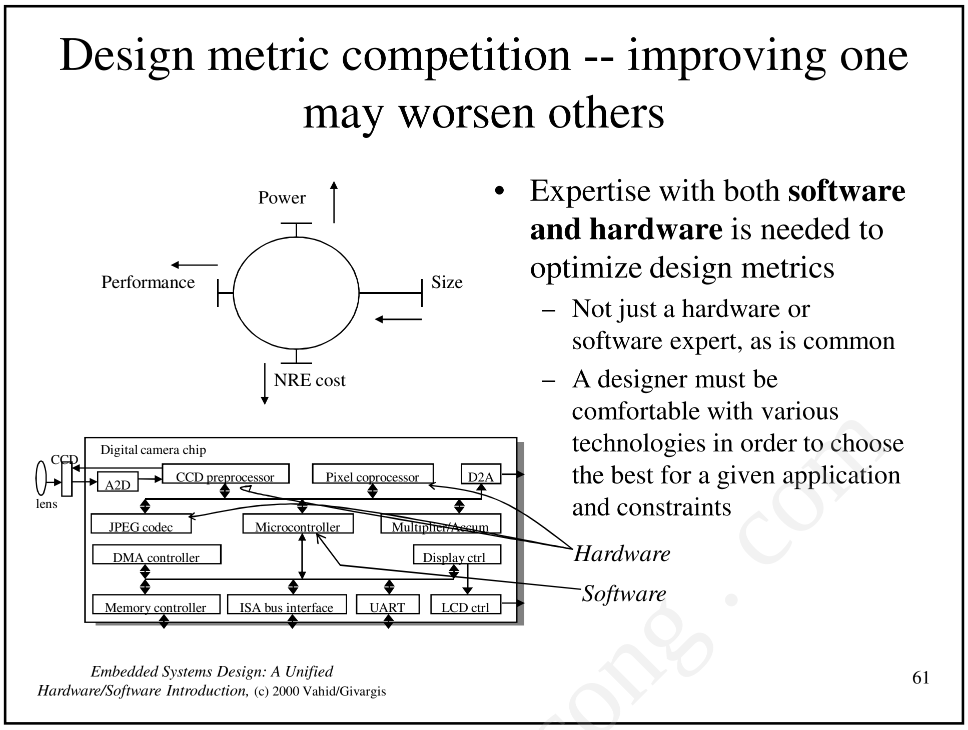
Design challenge – optimizing design metrics

- Common metrics
  - Unit cost: the monetary cost of manufacturing each copy of the system, excluding NRE cost
  - NRE cost (Non-Recurring Engineering cost): The one-time monetary cost of designing the system
  - Size: the physical space required by the system
  - Performance: the execution time or throughput of the system
  - Power: the amount of power consumed by the system
  - Flexibility: the ability to change the functionality of the system without incurring heavy NRE cost

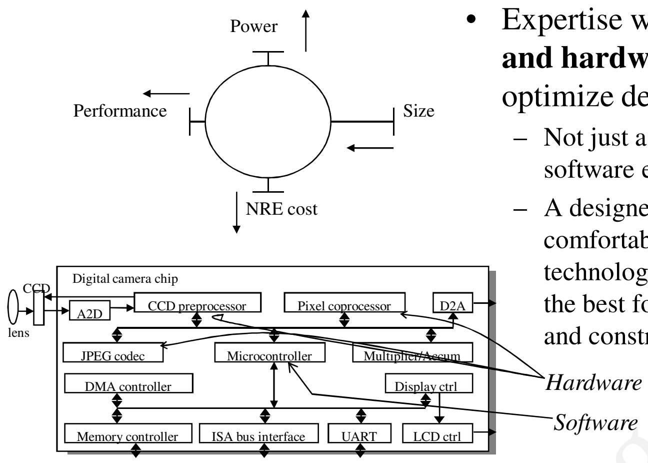
cuu duong than cong . com

Design challenge – optimizing design metrics

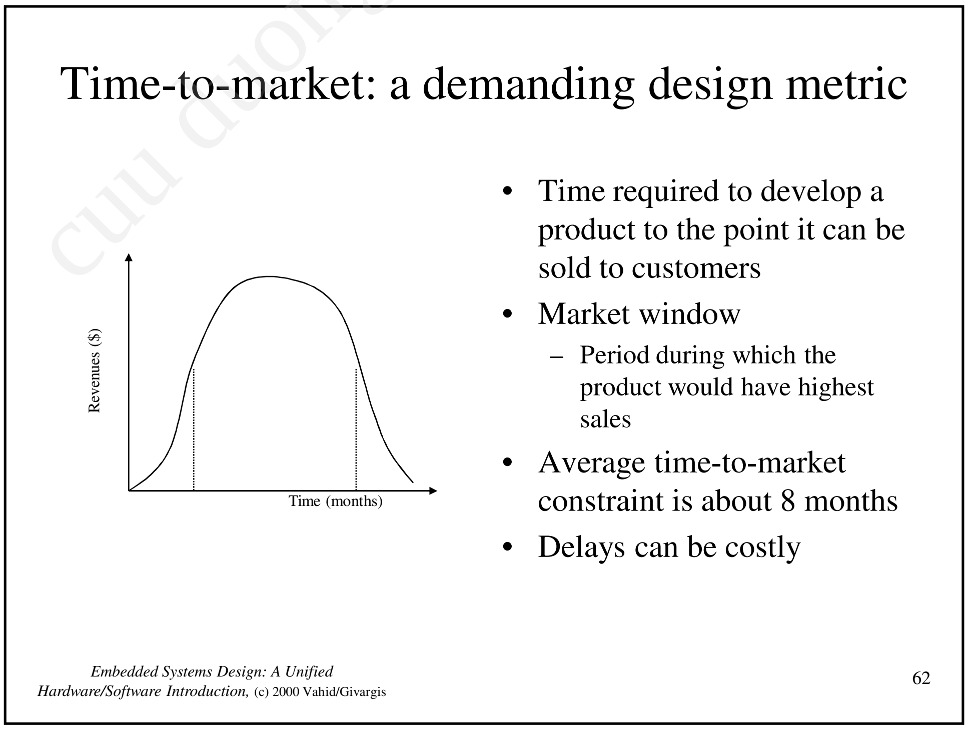
- Common metrics (continued)
  - Time-to-prototype: the time needed to build a working version of the system
  - Time-to-market: the time required to develop a system to the point that it can be released and sold to customers
  - Maintainability: the ability to modify the system after its initial release
  - Correctness, safety, many more



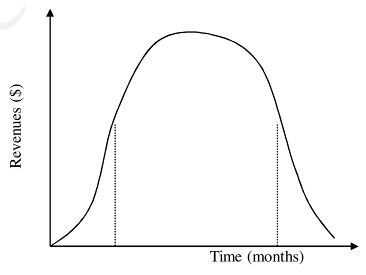
Design metric competition -- improving one may worsen others



- Expertise with both **software and hardware** is needed to optimize design metrics
  - Not just a hardware or software expert, as is common
  - A designer must be comfortable with various technologies in order to choose the best for a given application and constraints



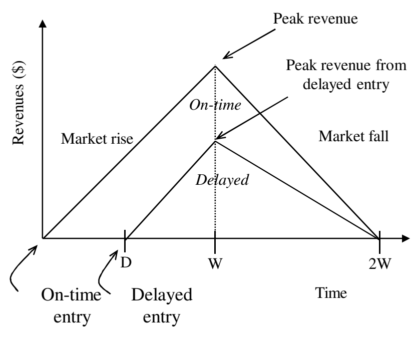
Time-to-market: a demanding design metric



- Time required to develop a product to the point it can be sold to customers
- Market window
  - Period during which the product would have highest sales
- Average time-to-market constraint is about 8 months
- Delays can be costly

cuu duong than cong . com

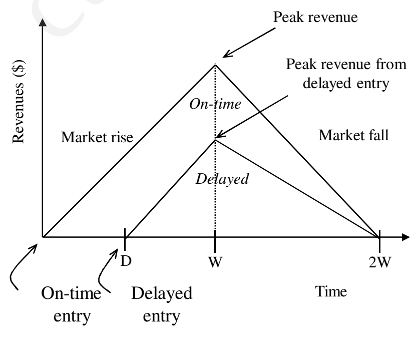
Losses due to delayed market entry



- Simplified revenue model
  - Product life = 2W, peak at W
  - Time of market entry defines a triangle, representing market penetration
  - Triangle area equals revenue
- Loss
  - The difference between the on-time and delayed triangle areas

cuu duong than cong . com

Losses due to delayed market entry (cont.)



- Area = 1/2 * base * height
  - On-time = 1/2 * 2W * W
  - Delayed = 1/2 * (W-D+W)*(W-D)
- Percentage revenue loss = $(D(3W\text{-}D)/2W^2)*100\%$
- Try some examples
  - Lifetime 2W=52 wks, delay D=4 wks
  - (4*(3*26 –4)/2*26^2) = 22%
  - Lifetime 2W=52 wks, delay D=10 wks
  - (10*(3*26 –10)/2*26^2) = 50%
  - Delays are costly!

NRE and unit cost metrics

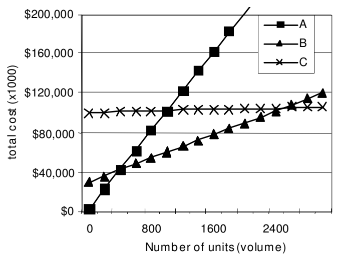
- Costs:
  - Unit cost: the monetary cost of manufacturing each copy of the system, excluding NRE cost
  - NRE cost (Non-Recurring Engineering cost): The one-time monetary cost of designing the system
  - *total cost = NRE cost + unit cost * # of units*
  - *per-product cost = total cost / # of units*
    = (NRE cost / # of units) + unit cost
- Example
  - NRE=$2000, unit=$100
  - For 10 units
    - total cost = $2000 + 10*$100 = $3000
    - per-product cost = $\underbrace{\$2000/10}$ + $100 = $300

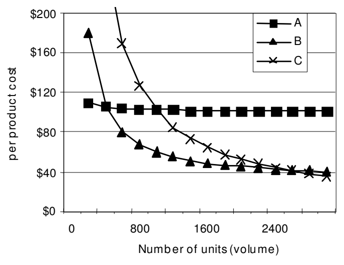
Amortizing NRE cost over the units results in an additional $200 per unit

cuu duong than cong . com

NRE and unit cost metrics

- Compare technologies by costs -- best depends on quantity
  - Technology A: NRE=$2,000, unit=$100
  - Technology B: NRE=$30,000, unit=$30
  - Technology C: NRE=$100,000, unit=$2





- But, must also consider time-to-market

The performance design metric

- Widely-used measure of system, widely-abused
  - Clock frequency, instructions per second – not good measures
  - Digital camera example – a user cares about how fast it processes images, not clock speed or instructions per second
- Latency (response time)
  - Time between task start and end
  - e.g., Camera's A and B process images in 0.25 seconds
- Throughput
  - Tasks per second, e.g. Camera A processes 4 images per second
  - Throughput can be more than latency seems to imply due to concurrency, e.g. Camera B may process 8 images per second (by capturing a new image while previous image is being stored).
- *Speedup* of B over S = B's performance / A's performance
  - Throughput speedup = 8/4 = 2

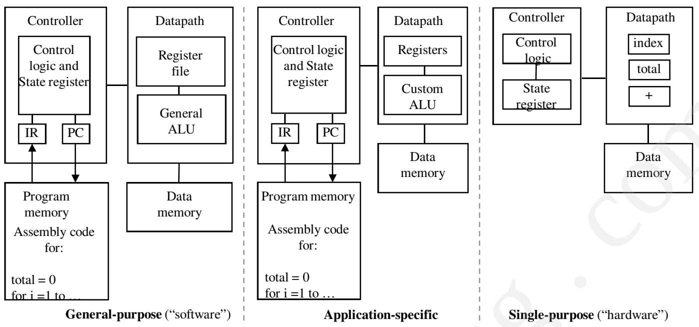
cuu duong than cong . com

Three key embedded system technologies

- Technology
  - A manner of accomplishing a task, especially using technical processes, methods, or knowledge
- Three key technologies for embedded systems
  - Processor technology
  - IC technology
  - Design technology

Processor technology

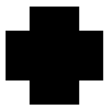
- The architecture of the computation engine used to implement a system's desired functionality
- Processor does not have to be programmable
  - "Processor" *not* equal to general-purpose processor



cuu duong than cong . com

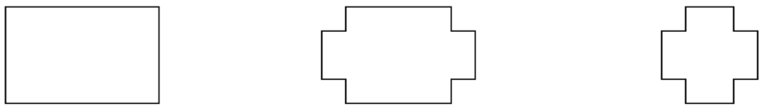
Processor technology

- Processors vary in their customization for the problem at hand



Desired functionality

```
total = 0
for i = 1 to N  loop
   total += M[i]
end loop
```



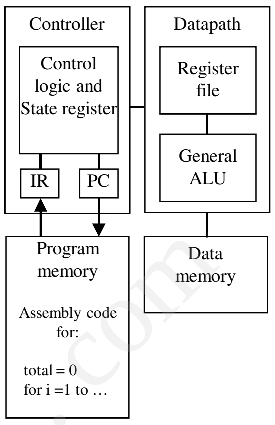
General-purpose processor

Application-specific processor

Single-purpose processor

General-purpose processors

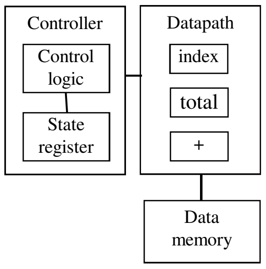
- Programmable device used in a variety of applications
  - Also known as "microprocessor"
- Features
  - Program memory
  - General datapath with large register file and general ALU
- User benefits
  - Low time-to-market and NRE costs
  - High flexibility
- "Pentium" the most well-known, but there are hundreds of others



cuu duong than cong . com

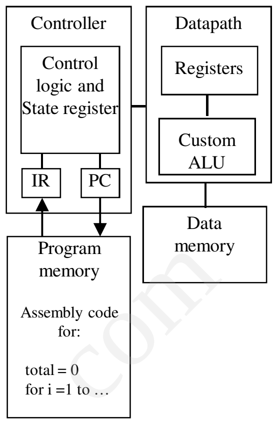
Single-purpose processors

- Digital circuit designed to execute exactly one program
  - a.k.a. coprocessor, accelerator or peripheral
- Features
  - Contains only the components needed to execute a single program
  - No program memory
- Benefits
  - Fast
  - Low power
  - Small size



Application-specific processors

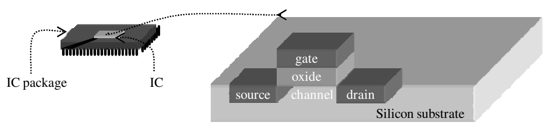
- Programmable processor optimized for a particular class of applications having common characteristics
  - Compromise between general-purpose and single-purpose processors
- Features
  - Program memory
  - Optimized datapath
  - Special functional units
- Benefits
  - Some flexibility, good performance, size and power



cuu duong than cong . com

IC technology

- The manner in which a digital (gate-level) implementation is mapped onto an IC
  - IC: Integrated circuit, or "chip"
  - IC technologies differ in their customization to a design
  - IC's consist of numerous layers (perhaps 10 or more)
    - IC technologies differ with respect to who builds each layer and when



IC technology

- Three types of IC technologies
  - Full-custom/VLSI
  - Semi-custom ASIC (gate array and standard cell)
  - PLD (Programmable Logic Device)

cuu duong than cong . com

Full-custom/VLSI

- All layers are optimized for an embedded system's particular digital implementation
  - Placing transistors
  - Sizing transistors
  - Routing wires
- Benefits
  - Excellent performance, small size, low power
- Drawbacks
  - High NRE cost (e.g., $300k), long time-to-market

Semi-custom

- Lower layers are fully or partially built
  - Designers are left with routing of wires and maybe placing some blocks
- Benefits
  - Good performance, good size, less NRE cost than a full-custom implementation (perhaps $10k to $100k)
- Drawbacks
  - Still require weeks to months to develop

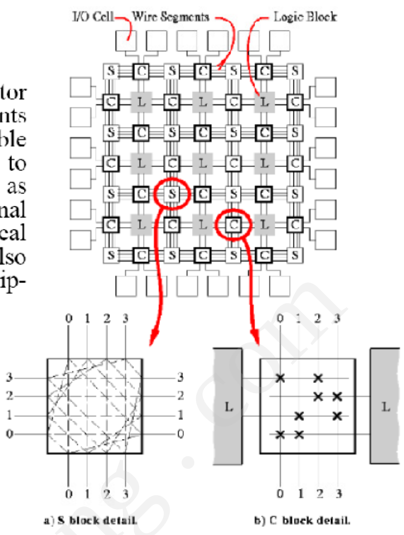
cuu duong than cong . com

PLD (Programmable Logic Device)

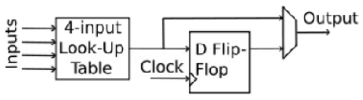
- All layers already exist
  - Designers can purchase an IC
  - Connections on the IC are either created or destroyed to implement desired functionality
  - Field-Programmable Gate Array (FPGA) very popular
- Benefits
  - Low NRE costs, almost instant IC availability
- Drawbacks
  - Bigger, expensive (perhaps $30 per unit), power hungry, slower

FPGA Design

A field-programmable gate array is a semiconductor device containing programmable logic components called "logic blocks", and programmable interconnects. Logic blocks can be programmed to perform the function of basic logic gates such as AND, and XOR, or more complex combinational functions such as decoders or simple mathematical functions. In most FPGAs, the logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory.



A classic FPGA logic block consists of a 4-input lookup table (LUT), and a flip-flop:



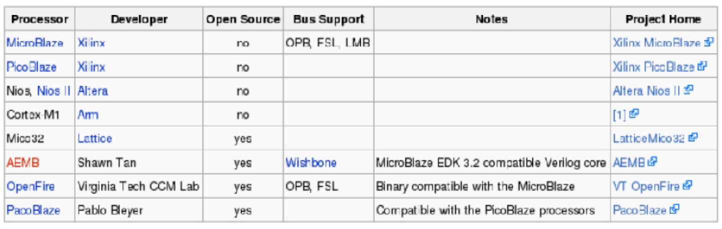
cuu duong than cong . com

Soft Processors

A soft microprocessor (also called softcore microprocessor or a soft processor) is a microprocessor core that can be wholly implemented using logic synthesis. It can be implemented via different semiconductor devices containing programmable logic (e.g., FPGA, CPLD).

Notable soft microprocessors include:

- MicroBlaze
- Nios II

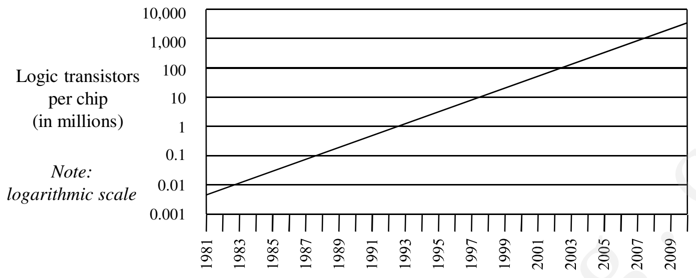


Processor	Developer	Open Source	Bus Support	Notes	Project Home
MicroBlaze	Xilinx	no	OPB, FSL, LMB		Xilinx MicroBlaze
PicoBlaze	Xilinx	no			Xilinx PicoBlaze
Nios, Nios II	Altera	no			Altera Nios II
Cortex-M1	Arm	no			[1]
Mico32	Lattice	yes			LatticeMico32
AEMB	Shawn Tan	yes	Wishbone	MicroBlaze EDK 3.2 compatible Verilog core	AEMB
OpenFire	Virginia Tech CCM Lab	yes	OPB, FSL	Binary compatible with the MicroBlaze	VT OpenFire
PacoBlaze	Pablo Bleyer	yes		Compatible with the PicoBlaze processors	PacoBlaze

Moore's law

- The most important trend in embedded systems
  - Predicted in 1965 by Intel co-founder Gordon Moore

IC transistor capacity has doubled roughly every 18 months for the past several decades

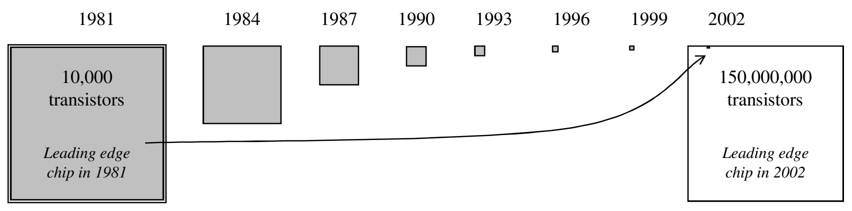


Moore's law

- Wow
  - This growth rate is hard to imagine, most people underestimate
  - How many ancestors do you have from 20 generations ago
    - i.e., roughly how many people alive in the 1500's did it take to make you?
    - 2^{20} = more than *1 million people*
  - *(This underestimation is the key to pyramid schemes!)*

cuu duong than cong . com

Graphical illustration of Moore's law

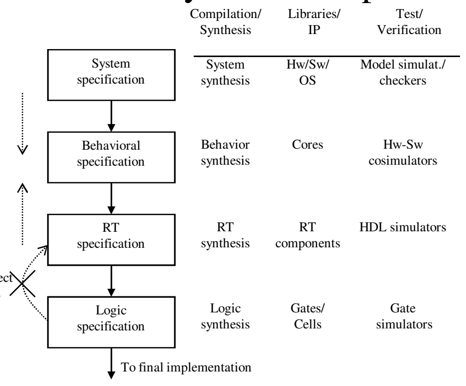


- Something that doubles frequently grows more quickly than most people realize!
  - A 2002 chip can hold about 15,000 1981 chips inside itself

cuu duong than cong . com

Design Technology

- The manner in which we convert our concept of desired system functionality into an implementation

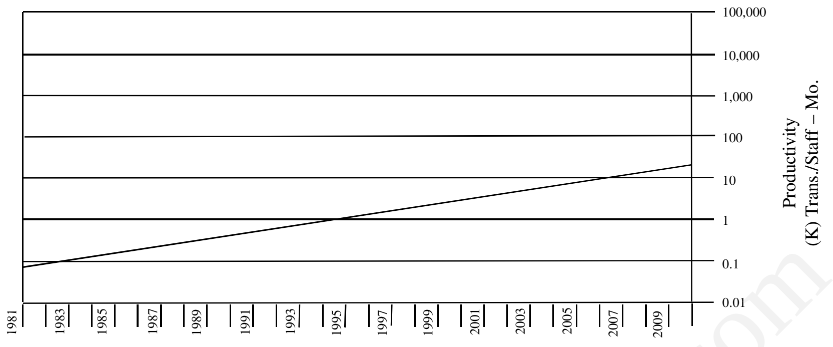


Compilation/Synthesis: Automates exploration and insertion of implementation details for lower level.

Libraries/IP: Incorporates pre-designed implementation from lower abstraction level into higher level.

Test/Verification: Ensures correct functionality at each level, thus reducing costly iterations between levels.

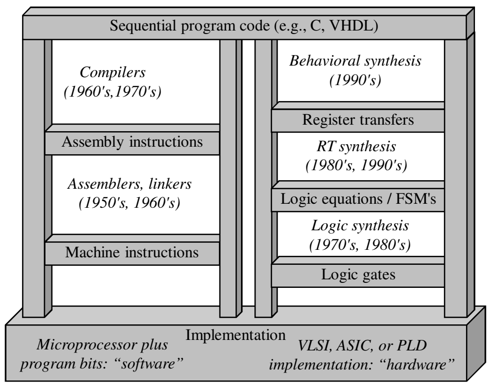
Design productivity exponential increase



- Exponential increase over the past few decades

The co-design ladder

- In the past:
  - Hardware and software design technologies were very different
  - Recent maturation of synthesis enables a unified view of hardware and software
- Hardware/software "codesign"



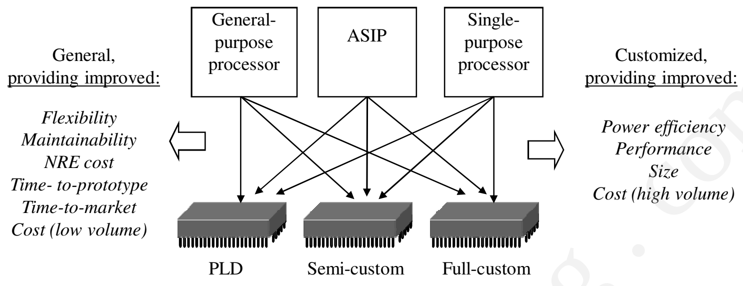
Sequential program code (e.g., C, VHDL)
Compilers (1960's,1970's)
Assembly instructions
Assemblers, linkers (1950's, 1960's)
Machine instructions
Behavioral synthesis (1990's)
Register transfers
RT synthesis (1980's, 1990's)
Logic equations / FSM's
Logic synthesis (1970's, 1980's)
Logic gates
Implementation
Microprocessor plus program bits: "software"
VLSI, ASIC, or PLD implementation: "hardware"

The choice of hardware versus software for a particular function is simply a tradeoff among various design metrics, like performance, power, size, NRE cost, and especially flexibility; there is no fundamental difference between what hardware or software can implement.

cuu duong than cong . com

Independence of processor and IC technologies

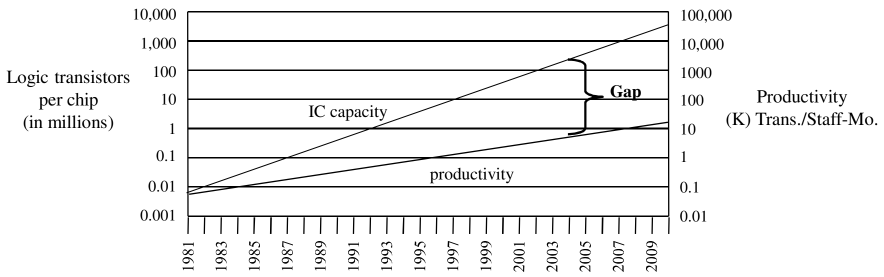
- Basic tradeoff
  - General vs. custom
  - With respect to processor technology or IC technology
  - The two technologies are independent



cuu duong than cong . com

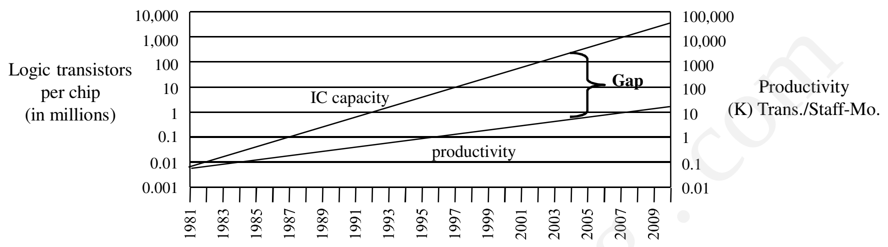
Design productivity gap

- While designer productivity has grown at an impressive rate over the past decades, the rate of improvement has not kept pace with chip capacity



Design productivity gap

- 1981 leading edge chip required 100 designer months
  - 10,000 transistors / 100 transistors/month
- 2002 leading edge chip requires 30,000 designer months
  - 150,000,000 / 5000 transistors/month
- Designer cost increase from $1M to $300M

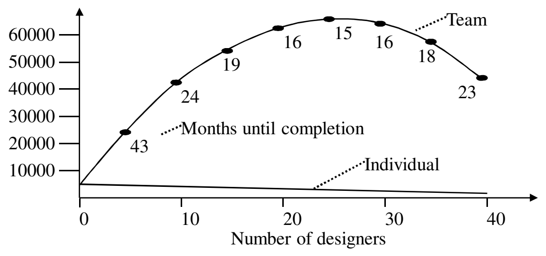


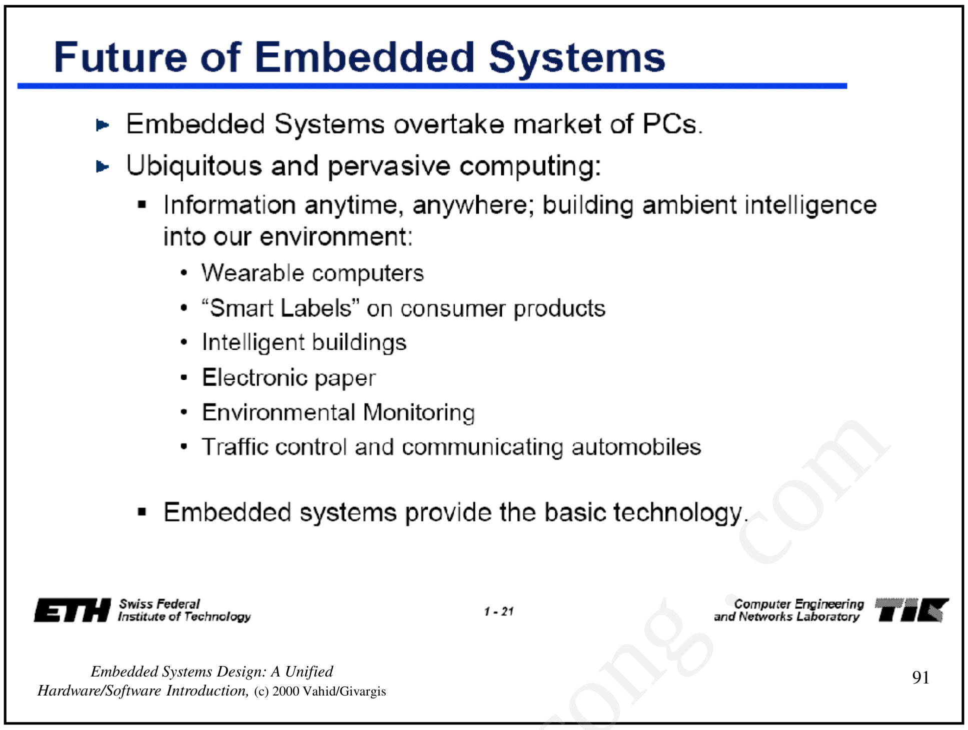
cuu duong than cong . com

The mythical man-month

- The situation is even worse than the productivity gap indicates
- In theory, adding designers to team reduces project completion time
- In reality, productivity per designer decreases due to complexities of team management and communication
- In the software community, known as "the mythical man-month" (Brooks 1975)
- At some point, can actually lengthen project completion time! ("Too many cooks")

- 1M transistors, 1 designer=5000 trans/month
- Each additional designer reduces for 100 trans/month
- So 2 designers produce 4900 trans/month each



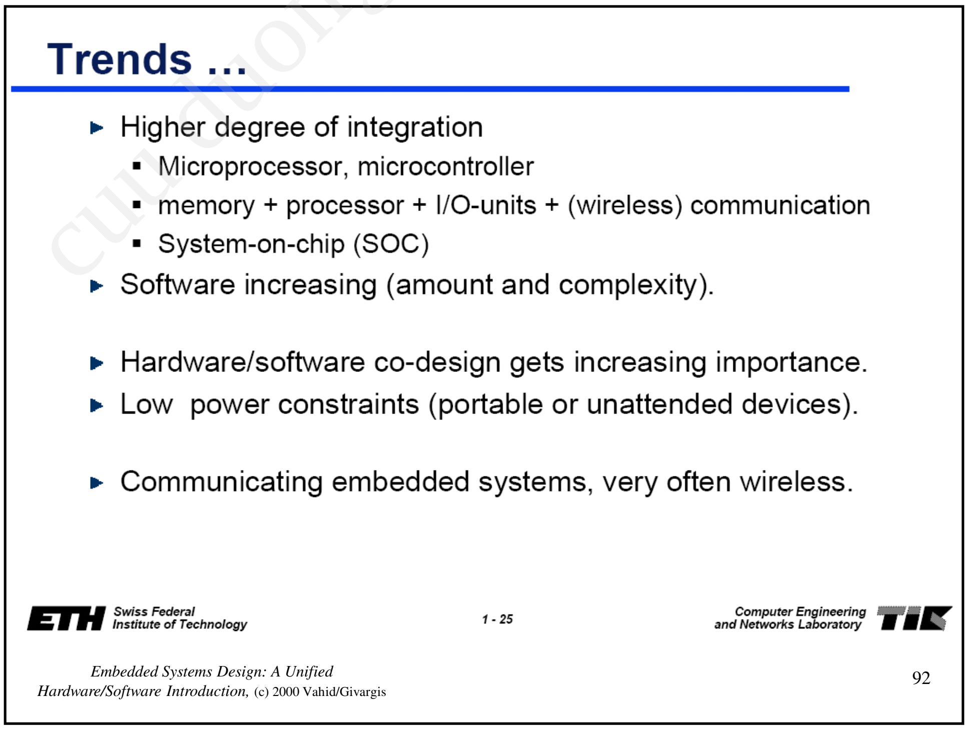


Future of Embedded Systems

- Embedded Systems overtake market of PCs.
- Ubiquitous and pervasive computing:
  - Information anytime, anywhere; building ambient intelligence into our environment:
    - Wearable computers
    - “Smart Labels” on consumer products
    - Intelligent buildings
    - Electronic paper
    - Environmental Monitoring
    - Traffic control and communicating automobiles
  - Embedded systems provide the basic technology.

ETH Swiss Federal Institute of Technology 1 - 21 Computer Engineering and Networks Laboratory TIK

Embedded Systems Design: A Unified Hardware/Software Introduction, (c) 2000 Vahid/Givargis 91



Trends …

- Higher degree of integration
  - Microprocessor, microcontroller
  - memory + processor + I/O-units + (wireless) communication
  - System-on-chip (SOC)
- Software increasing (amount and complexity).
- Hardware/software co-design gets increasing importance.
- Low power constraints (portable or unattended devices).
- Communicating embedded systems, very often wireless.

ETH Swiss Federal Institute of Technology 1 - 25 Computer Engineering and Networks Laboratory TIK

Embedded Systems Design: A Unified Hardware/Software Introduction, (c) 2000 Vahid/Givargis 92

cuu duong than cong . com

Summary

- Embedded systems are everywhere
- Key challenge: optimization of design metrics
  - Design metrics compete with one another
- A unified view of hardware and software is necessary to improve productivity
- Three key technologies
  - Processor: general-purpose, application-specific, single-purpose
  - IC: Full-custom, semi-custom, PLD
  - Design: Compilation/synthesis, libraries/IP, test/verification

cuu duong than cong . com